EFFICIENT-LVSM: FASTER, CHEAPER, AND BETTER LARGE VIEW SYNTHESIS MODEL VIA DECOUPLED CO-REFINEMENT ATTENTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Feedforward models for novel view synthesis (NVS) have recently advanced by transformer-based methods like LVSM, using attention among all input and target views. In this work, we argue that its full self-attention design is suboptimal, suffering from quadratic complexity with respect to the number of input views and rigid parameter sharing among heterogeneous tokens. We propose **Efficient-LVSM**, a dual-stream architecture that avoids these issues with a decoupled co-refinement mechanism. It applies intra-view self-attention for input views and self-then-cross attention for target views, eliminating unnecessary computation. Efficient-LVSM achieves 30.6 dB PSNR on RealEstate10K with 2 input views, surpassing LVSM by 0.9 dB, with 2× faster training convergence and 4.4× faster inference speed. Efficient-LVSM achieves state-of-the-art performance on multiple benchmarks, exhibits strong zero-shot generalization to unseen view counts, and enables incremental inference with KV-cache, thanks to its decoupled designs.

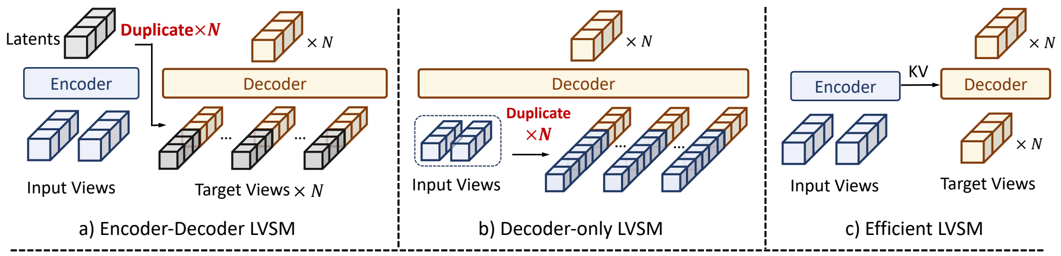
1 INTRODUCTION

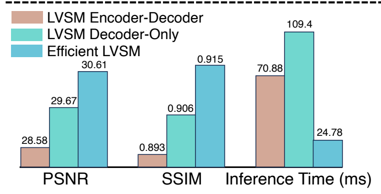
Reconstructing 3D scenes from a collection of 2D images remains a cornerstone challenge in computer vision. The field has witnessed a remarkable evolution, moving from classical photogrammetry systems to per-scene optimized neural representations like NeRF (Mildenhall et al., 2020) and 3DGS (Kerbl et al., 2023), which achieve high-quality reconstruction, but require dense inputs and costly optimization for each new scene. A significant advance came from Large Reconstruction Models (LRMs) (Hong et al., 2024; Wei et al., 2024; Zhang et al., 2024), which learn generalizable 3D priors from vast datasets. A recent paradigm shift, pioneered by models like LVSM (Jin et al., 2025), has further advanced the field by minimizing hand-crafted inductive biases, where it directly synthesizes novel views from posed images. It eliminates the need for predefined 3D structures or rendering equations and achieves surprisingly good rendering quality with flexibility.

Despite the success, its monolithic self-attention mechanism, where all input and target tokens are concatenated into a single sequence, leads to two primary drawbacks: (1) Low efficiency: full self-attention leads to quadratic complexity with regard to the number of input views. Furthermore, when generating multiple target views with the same input views, input representation can not be re-used. (2) Limited performance: full self-attention enforces parameter sharing for heterogeneous tokens - content-rich input views and pose-only target queries. It hinders the model's ability to to learn specialized representations for their distinct tasks, i.e., understanding the semantics & 3D structure of the scene for input tokens and rendering the novel view for target tokens.

In this work, we systematically analyze these trade-offs and derive **Efficient-LVSM**, a Transformer-based architecture designed to resolve these limitations. The key insight is to **decouple** the process of input view encoding from target view generation. To achieve this, Efficient-LVSM is composed of two specialized pathways: an **Input Encoder** that independently processes input views, and a **Target Decoder** that synthesizes novel views by querying the encoded input features at each layer. This dual-stream architecture endows our model with four key properties:

- **Specialized Attention Pathways.** Our architecture utilizes distinct modules for input and target tokens. In the input encoder, only input view is processed. In the target decoder, target tokens act as queries and input tokens serve as keys and values in cross-attention, avoiding the use of shared parameters for heterogeneous information.





Attribute	LVSM Encoder-Decoder	LVSM Decoder-Only	Efficient-LVSM **(Ours)**
Information Integrity	✗	✓	✓
Specialization Pathways	✓	✗	✓
Rendering Quality	✗	✓	✓✓
Efficiency (Speed/Memory)	✓	✗	✓✓
Incremental Inference	✗	✗	✓

Figure 1: **Latent Novel View Synthesis Paradigms Comparison.** The proposed decoupled architecture disentangles the input and target streams. It maintains the integrity and specialization with high efficiency, obtaining better rendering quality and faster inference speed.

- **Robustness to Variable View Counts.** The input self-attention processes each view separately, making the transformation of one view independent of others. This per-view processing strategy allows the model to generalize better than LVSM to a variable number of input views at test time.
- **Computational and Memory Efficiency.** The input encoder processes each input view separately and the target decoder adopts cross-attention, both reducing the computational complexity with respect to the number of input views from **quadratic** $O(N_{in}^2)$ to **linear** $O(N_{in})$.
- **Incremental Inference via KV-Cache.** The decoupled structure enables KV-cache of input view features. When a new input view is provided, only that view needs to be processed. When a new target view is required, the KV-cache could be directly re-used. In summary, the cost of adding new input views and target views is nearly constant and thus enables incremental inference.

We conduct comprehensive evaluations for Efficient-LVSM. It sets a new state-of-the-art, **outperforming LVSM by 0.9dB PSNR and GS-LRM by 2.5dB PSNR** on the RealEstate10K benchmark with 50% training time and achieves $2-4$ times speed acceleration in terms of both training iteration and inference. It exhibits strong zero-shot generalization to unseen numbers of input views.

2 Method

In this section, we present a step-by-step analysis that derives the design of Efficient-LVSM.

2.1 Preliminary

Task Definition: Given N input images with known camera poses and M target view camera poses, novel view synthesis (NVS) aims to render M corresponding target images. Specifically, the input is $\{(\mathbf{I}_i, \mathbf{E}_i, \mathbf{K}_i) | i = 1, 2, ..., N\}$ and $\{(\mathbf{E}_i, \mathbf{K}_i) | i = 1, 2, ..., M\}$, where $\mathbf{I} \in \mathbb{R}^{H \times W \times 3}$ is the input RGB image, H and W are the height and width, $\mathbf{E}, \mathbf{K} \in \mathbb{R}^{4 \times 4}$ are camera extrinsic and intrinsic. The output is rendered target images, denoted as $\{\hat{\mathbf{I}}_i | i = 1, 2, ..., M, \hat{\mathbf{I}} \in \mathbb{R}^{H \times W \times 3}\}$

Feedforward NVS Framework: we adopt LVSM (Jin et al., 2025) end-to-end paradigm. For the input $\{(\mathbf{I}_i, \mathbf{E}_i, \mathbf{K}_i)\}_{i=1}^N$ and $\{(\mathbf{E}_i, \mathbf{K}_i)\}_{i=1}^M$, all camera poses are encoded using Plücker ray embedding (Plucker, 1865) while input images are patchified as in ViT (Dosovitskiy et al., 2020). We obtain the input tokens $\{S_i\}_{i=1}^N$ by concatenating its RGB patches and Plücker ray patches in the hidden dimension and passing through an MLP. We obtain the target tokens $\{T_i\}_{i=1}^M$ by feeding its Plücker ray patches into another MLP.

Next, input and target tokens pass through a set of transformer blocks to extract features, which is the key component of the framework: $\{R_i\}_{i=1}^M = \Phi(\{S_i\}_{i=1}^N, \{T_i\}_{i=1}^M)$ where Φ represents the transformer blocks and $\{R_i\}_{i=1}^M$ is the final features of target views.

The output layer transforms the final features of target views $\{R_i\}_{i=1}^M$ into RGB value by a linear layer followed by a sigmoid function. These RGB patches are then unpatchified to target images:

$$\hat{\mathbf{I}}_i^t = \text{unpatchify}(\text{Sigmoid}(\text{Linear}_{\text{render}}(\mathbf{R}_i)) \in \mathbb{R}^{H \times W \times 3}, \quad (1)$$

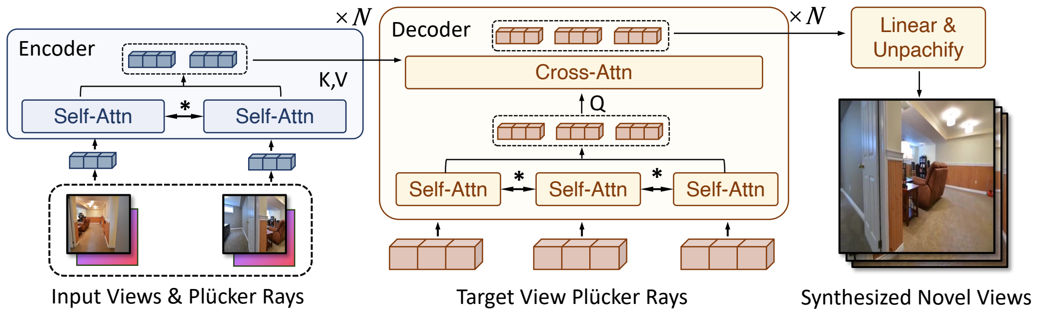


Figure 2: **Efficient-LVSM Model Structure.** Efficient-LVSM patchifies posed input images and target Plücker into tokens. Tokens of each input view separately pass through an encoder to extract contextual information. Target tokens cross-attend to input tokens and render new views.

2.2 Analysis of LVSM's Full Self-Attention Paradigm

LVSM deocder-only model employs full self-attention on all input and target tokens, which introduces the following two limitations:

Entangled Representation. From the content perspective, input tokens contain both semantic and geometric information, while target tokens only have geometric information. From the system perspective, they bear distinct tasks: input tokens are to understand the semantics & 3D structure of the scene and target tokens are to render the novel view. However, shared self-attention parameters do not distinguish the difference, hampering the generalization ability, as evidenced experiments in Table 2 and visualizations in Fig. 10.

Computation and Memory Costs. Consider a sample (S_i, T_i) with the shapes of $Np_n \times d$ and $Mp_n \times d$, where N and M are the numbers of input and target views, and $p_n = HW/p^2$ is the number of patches. LVSM decoder-only model constructs M separate sequences for M target views, where each sequence is a concatenation of the entire set of input tokens and the tokens of a single target view. These sequences are processed by full self-attention:

$$V_i^l = \text{concat}(S_1^l, S_2^l, ..., S_N^l, T_i^l); \quad V_i^l = V_i^{l-1} + \text{Self-Attn}_{\text{full}}^l(V_i^{l-1}) \tag{2}$$

The shape of V_i is $M \times (Np_n + p_n) \times d$. LVSM repeats the computation of one target view for M times. Thus, the temporal complexity of LVSM decoder-only model is $M \cdot O(N^2 p_n^2) = O(N^2 M)$, as shown in Fig. 3 (a) and Table 1. The quadratic complexity with regard to the number of input views hampers the efficiency and the repetition of tokens introduces severe computational cost.

LVSM encoder-decoder structure avoids the repetition issue by using an encoder to compress all input views into one latent vector first. However, this design introduces **loss of information**, significantly limiting the reconstruction quality, which is acknowledged in LVSM paper Jin et al. (2025).

Structure	Overall Complexity	Component	Complexity
LVSM Encoder-Decoder	$O(N^2 + M)$	Encoder Decoder	$O(N^2)$ $O(M)$
LVSM Decoder-Only	$O(M(N+1)^2)$	Decoder	$O(M(N+1)^2)$
Efficient-LVSM (**Ours**)	$O(NM + N)$	Encoder Decoder	$O(N)$ $O(NM)$

Table 1: **Comparison of Model Structure Complexity.** The proposed Efficient-LVSM obtains lower complexity than LVSM and thus achieves significant speed up, as evidenced in Sec. 3.4.

2.3 Dual-Stream Paradigm

Based on the observation above, we propose a dual-stream structure, where distinct modules are applied on input and target tokens to decouple the information flow, as in Fig. 2.

Input Encoder: To maintain the independency of different input views and improve efficiency, we limit the scope of self-attention to patches within the same input view. Each input view is processed separately, which enables efficient inference when a new input view is provided (incremental inference). Instead of constructing a single, prohibitively long attention sequence containing tokens from all N input views, we propose to process N shorter sequences. Specifically, let S_i represent the tokens of the i^{th} input view. They are updated by an intra-view self-attention block at layer l:

$$S_i^l = S_i^{l-1} + \text{Self-Attn}_{\text{input}}^l(S_i^{l-1}); \quad S_i^l = S_i^l + \text{FFN}_{\text{input}}^l(S_i^l) \tag{3}$$

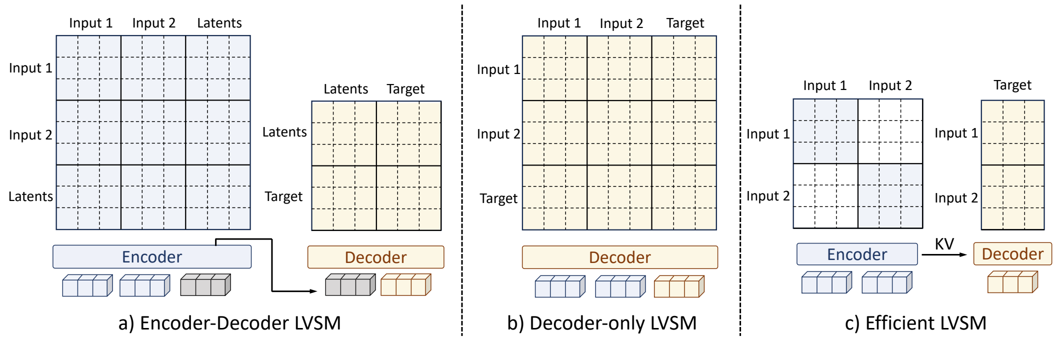


Figure 3: **Comparison of Attention Paradigms.** The shaded areas indicate the token pairs for which attention scores are computed. EfficientLVSM adopts intra-view self-attention for inputs and allows the target decoder to cross-attend to the full set of uncompressed input features.

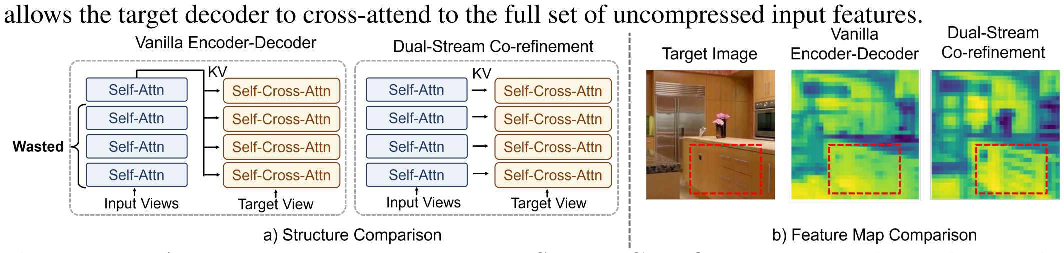


Figure 4: **Vanilla Encoder-Decoder vs. Dual-Stream Co-refinement.** (a) Hidden features in middle layers in vanilla encoder-decoder are wasted while the dual-stream co-refinement structure utilizes these features to extract more information. (b) Feature maps indicate that co-refinement structure catches more details of the target view.

Target Decoder: To allow efficient KV-cache for features of input views, target decoder employs cross-attention, letting output tokens T_i^l attend to input tokens S_i^L from the last layer of encoder:

$$T_i^l = T_i^l + \text{Cross-Attn}_{\text{target}}^l(T_i^l, S_1^L, S_2^L, ..., S_N^L); \quad T_i^l = T_i^l + \text{FFN}_{\text{input}}^l(T_i^l) \tag{4}$$

This design decouples the parameters for input and output tokens and allows rendering multiple target views with the same input KV-cache. Assuming the hidden dimension and the number of patches per image are constants, the temporal complexity of the Target Decoder are $O(NM)$, while complexity of LVSM decoder-only is $O(M(N+1)^2)$, as in Table 1.

2.4 Intra-View Attention of Target Tokens in Decoder

The aforementioned cross-attention only decoder design introduces a drawback: **each target token has to store the information of the whole scene by their own** since there is no scene-level interaction in input encoder, limiting the capacity. To this end, we propose to add intra-view self-attention in target decoder alternatively with the original cross attention :

$$\begin{aligned} T_i^l &= T_i^{l-1} + \text{Self-Attn}_{\text{target}}^l(T_i^{l-1}) \\ T_i^l &= T_i^l + \text{Cross-Attn}_{\text{target}}^l(T_i^l, S_1^L, S_2^L, ..., S_N^L) \\ T_i^l &= T_i^l + \text{FFN}_{\text{input}}^l(T_i^l) \end{aligned} \tag{5}$$

In this way, the intra-view self-attention in decoder allows to integrate scene-level information from other target tokens while still maintaining KV-cache ability. Experiments Table 5 (a) demonstrates 6+6 layers self-then-cross attention performs better than 12 layers cross-attention.

2.5 Co-Refinement of Encoder-Decoder

One widely observed phenomenon for deep neural network is that **different layers of features represent different abstract level of informantion** (Zeiler & Fergus, 2013): early layers capturing fine-grained details such as textures, and later layers encoding high-level semantics. In vanilla encoder-decoder, only last layer features are used, as in Fig. 4 (a), which limits its capacity.

To this end, we propose a dual-stream co-refinement structure, illustrated in Fig. 4 (b), where each layer of the encoder provides information to its corresponding layer in the decoder. At layer l, input tokens S^l are first updated by self-attention, and then the target decoder queries these updated tokens

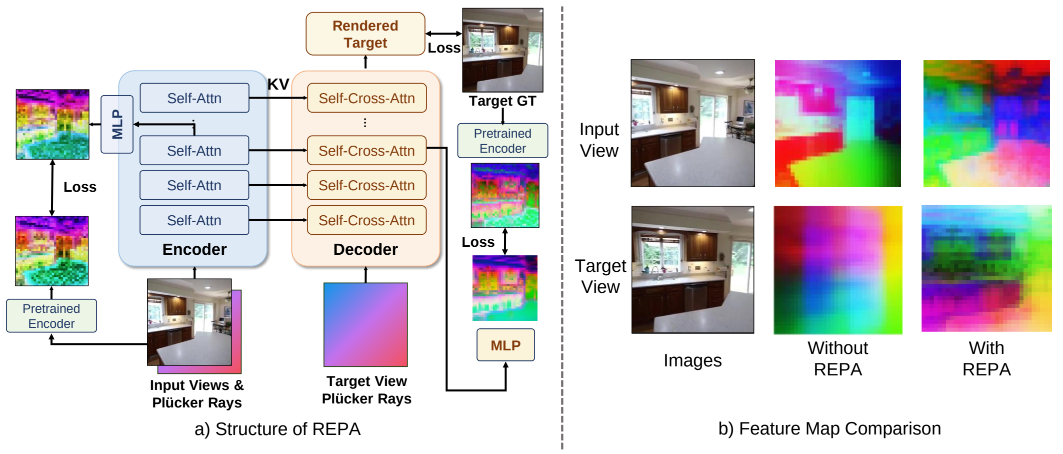


Figure 5: **Applying REPA into Efficient-LVSM.** (a) Pretrained vision encoders and MLP projectors are discarded in inference. (b) Feature maps indicate that REPA helps the model extract semantics.

to refine its own representation T^l:

$$\begin{aligned} T_i^l &= T_i^{l-1} + \text{Self-Attn}^l_{\text{target}}(T_i^{l-1}) \\ T_i^l &= T_i^l + \text{Cross-Attn}^l_{\text{target}}(T_i^l, S_1^l, S_2^l, ..., S_N^l) \\ T_i^l &= T_i^l + \text{FFN}^l_{\text{input}}(T_i^l) \end{aligned} \tag{6}$$

By querying the encoder's representations in the middle layers, the decoder can synthesize its own features using both the fine-grained details from early layers and the rich semantic context from later ones. Fig. 4 (b) demonstrates that the co-refinement model generate more detailed and high-quality features compared to vanilla encoder-decoder structure.

2.6 Distillation with REPA

With the decoupled attention for different views, a natural thought is to utilize those powerful pre-trained vision encoder. To utilize visual features without sacrificing inference speed, we employ REPA (Yu et al., 2025) to distill visual features from DINOv3 (Siméoni et al., 2025). Formally, consider a clean image $\mathbf{I}$ and $h_\phi(X^k)$ is the projection of hidden features of layer k, where h_ϕ is a trainable projector and X^k represents the input tokens or target tokens of layer k: $X^k = S^k$ or $X^k = T^k$. Let f represent the pretrained encoder such as DINOv3. The goal is to align the projection of layer output $h_\phi(X^k)$ with encoded images $f(\mathbf{I})$ by maximizing the patch-wise similarities:

$$\mathcal{L}_{REPA} = \frac{1}{N} \sum_{i=1}^{N} \text{sim}(f(\mathbf{I}), h_\phi(X^k)) \tag{7}$$

We find that improvement with REPA is conditional. Experiment in Table 5 show that LVSM benefits much less compared to Efficient LVSM's dual-stream co-refinement design structure, possibly due to its full self-attention design entangles feature maps of different views.

2.7 KV-cache & Incremental Inference

A key advantage of the decoupled dual stream design is its natural compatibility with KV caching during inference as illustrated in Fig. 6. The key and values of all input views, $\{\hat{S}_i\}_{i=1}^N$, can be computed once and stored. When a new target view is required, the decoder could directly utilize the stored cache $\{\hat{S}_i\}_{i=1}^N$ for rendering. When a new input view $\mathbf{I}_{N+1}$ is introduced, only this new view needs to be processed and appended into the cache. As a result, it enables efficient incremental inference, which could be used in interactive application scenarios.

3 Experiments

3.1 Datasets

Scene-level Datasets. We use the widely used RealEstate10K dataset (Zhou et al., 2018). It contains 80K video clips curated from 10K YouTube videos, including both indoor and outdoor scenes. We follow the training/testing split applied in LVSM (Jin et al., 2025).

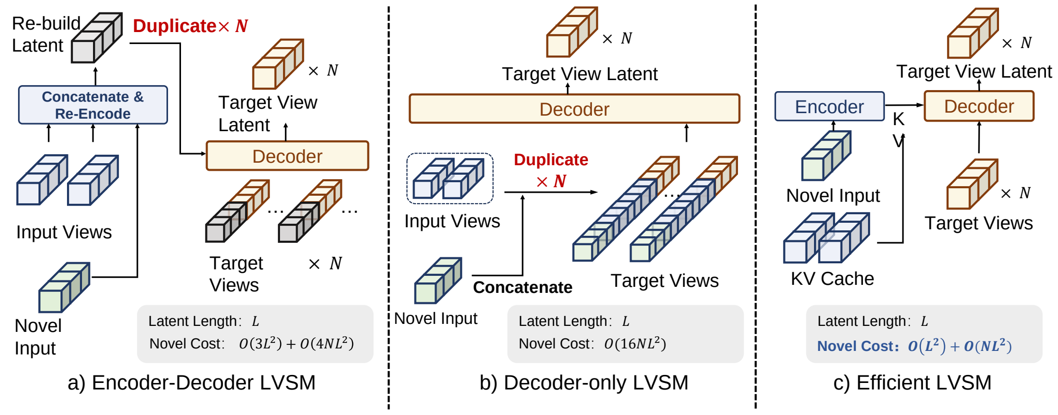


Figure 6: **Efficient Incremental Inference with KV-Cache.** Efficient LVSM saves computation when provided with novel inputs or targets by caching the key and value for previous input views.

Table 2: **Scene-level View Synthesis Quality.** We test on the same validation set proposed in pixelSplat.

	RealEstate10k (Zhou et al., 2018)		
	PSNR ↑	SSIM ↑	LPIPS ↓
pixelNeRF	20.43	0.589	0.550
GPNR	24.11	0.793	0.255
Du et al.	24.78	0.820	0.213
pixelSplat	26.09	0.863	0.136
MVSplat	26.39	0.869	0.128
GS-LRM	28.10	0.892	0.114
LVSM Enc-Dec	28.58	0.893	0.114
LVSM Dec-Only	29.67	0.906	0.098
Ours	30.61	0.915	0.087

Table 3: **Object-level View Synthesis Quality.** We test at 512 and 256 resolution on both input and rendering. "Enc" means encoder and "Dec" means decoder.

	ABO (Collins et al., 2022)			GSO (Downs et al., 2022)		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Triplane-LRM (Res-512)	27.50	0.896	0.093	26.54	0.893	0.064
GS-LRM (Res-512)	29.09	0.925	0.085	30.52	0.952	0.050
LVSM Enc-Dec (Res-512)	29.81	0.913	0.065	29.32	0.933	0.052
LVSM Dec-Only (Res-512)	32.10	0.938	0.045	32.36	0.962	0.028
Ours (Res-512)	32.65	0.951	0.042	32.92	0.973	0.021
LGM (Res-256)	20.79	0.813	0.158	21.44	0.832	0.122
GS-LRM (Res-256)	28.98	0.926	0.074	29.59	0.944	0.051
LVSM Enc-Dec (Res-256)	30.35	0.923	0.052	29.19	0.932	0.046
LVSM Dec-Only (Res-256)	32.47	0.944	0.037	31.71	0.957	0.027
Ours (Res-256)	33.13	0.960	0.035	32.73	0.969	0.022

Object-level Dataset. We use the Objaverse dataset (Deitke et al., 2023) to train our model. Following the rendering settings in GS-LRM (Zhang et al., 2024), we render 730K objects, and each object contains 32 random views. We test our object-level model on Google Scanned Objects (Downs et al., 2022) (GSO) and Amazon Berkeley Objects (Collins et al., 2022) (ABO), containing 1099 and 1000 objects respectively. Following Instant-3D (Li et al., 2023) and LVSM (Jin et al., 2025), we render 4 structured input views and 10 random target views for testing.

3.2 Implementaion Details

Model Details. Following LVSM (Jin et al., 2025), we use a patch size of 8×8 for the image tokenizer with 24 transformer layers (12-layer encoder and 12-layer decoder) and the dimension of hidden feature 1024. Following REPA (Yu et al., 2025), we select a 3-layer MLP as the alignment projection layer.

Protocols. Following the settings in LVSM, we select 4 input views and 8 target views in the object-level dataset. We select 2 input views and 3 target views in scene-level dataset.

3.3 Comparison with Start-of-the-Art Models

Scene-Level Comparison. We compare on scene-level inputs with pixelNeRF (Yu et al., 2021), GPNR (Suhail et al., 2022), Du et al. (Du et al., 2023), pixelSplat (Charatan et al., 2024), MVSplat (Chen et al., 2024), GS-LRM (Zhang et al., 2024), LVSM encoder-decoderand LVSM decoder-only (Jin et al., 2025). As in Table 2, our model establishes a new state-of-the-art on the RealEstate10K benchmark, outperforming the previous leading method, LVSM decoder-only, by a significant margin of 0.9 dB PSNR. This corresponds to **an** 18.7% **reduction in Mean Squared Error (MSE)**, indicating a substantial improvement in reconstruction fidelity. This quantitative leap is supported by our qualitative results in Figure 7, where our model produces noticeably sharper renderings and demonstrates superior geometric accuracy, particularly when synthesizing near-field objects where LVSM often introduces artifacts. Notably, this state-of-the-art performance is achieved with remarkable efficiency. Our model was trained for just 3 days on 64 A100 GPUs, which is **half the training time required by LVSM**. In essence, Efficient-LVSM not only surpasses the previous state-of-the-art in quality but does so while **requiring only** 50% **of the training budget**.

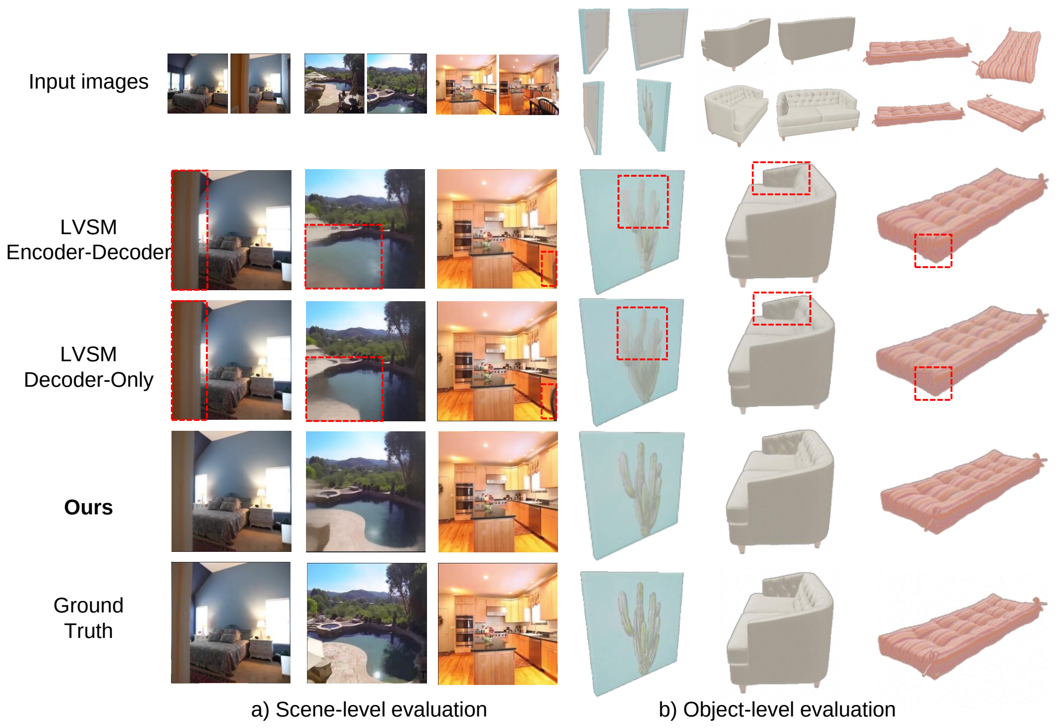


Figure 7: **NVS Visual Comparison.** We compare with LVSM (Jin et al., 2025) in RealEstate10K (Zhou et al., 2018) and Amazon Berkeley Objects (Collins et al., 2022). Images rendered by our model have less blur details.

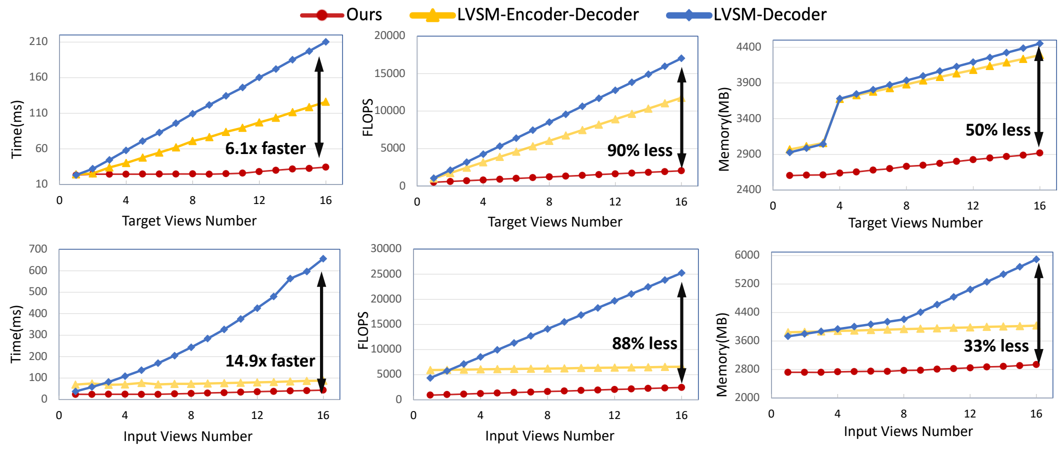


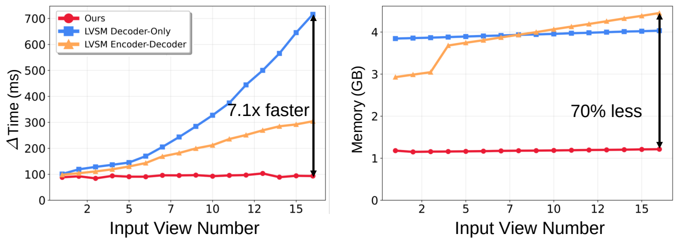
Figure 8: **Inference Speed Comparison.** We compare the inference time (ms) against (a) the number of target views and (b) the number of input views. Our model achieves consistently low latency. The performance of the LVSM baselines, particularly LVSM Decoder-Only, degrades severely as view counts increase. This highlights our model's significant computational efficiency, achieving up to a 14.9x speedup over LVSM Decoder-Only.

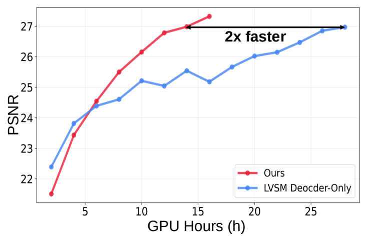
Object-Level Comparison. Similarly, Efficient-LVSM achieves state-of-the-art performance.

3.4 Efficiency Analysis

We evaluate the efficiency from three perspectives: vanilla inference latency, incremental inference latency, and training convergence speed. For fair comparison, we keep the number of layers (12+12) and hidden dimension (1024) the same with LVSM. For the convergence analysis, smaller variants are used for fast verification to save computational resource.

Vanilla Inference Speed. We analyze the inference cost by measuring latency, memory peak, and total GFLOPS as a function of both input and target view counts. As shown in Fig. 8, our model's





(a) **Incremental Inference Experiments.** We compare the inference latency and memory consumption when the input view is fed one by one. We observe that Efficient LVSM achieves near constant latency and memory consumption due to its KV-cache ability.

(b) **Convergence Speed Comparison.** Efficient LVSM achieves 2X faster convergence and achieves higher PNSR in the end.

Figure 9: **Efficiency Comparison**.

Table 5: **Ablation Study.**

(a) **Architectural Components Ablation.**

Arch.	PSNR ↑	SSIM ↑	LPIPS ↓
Cross-Attention Only	24.18	0.7908	0.1982
Self-Cross Attention	24.97	0.8201	0.1628
Co-Refinement	26.25	0.8462	0.1490

(b) **Effect of REPA**

Arch./Variant	PSNR ↑	SSIM ↑	LPIPS ↓
LVSM Dec-Only	25.52	0.8385	0.1541
LVSM Dec-Only w REPA	25.68	0.8410	0.1515
Ours w/o REPA	26.02	0.8483	0.1481
Ours w REPA	26.81	0.8628	0.1296

(c) **Effect of Model Sizes**

Models	Parameters	PSNR ↑	SSIM ↑	LPIPS ↓
Enc(12) + Dec(12)	199M	28.32	0.8892	0.1106
Enc(6) + Dec(6)	101M	27.77	0.8871	0.1149
Enc(3) + Dec(3)	53M	26.43	0.8609	0.1377

(d) **Size and Speed Comparison**

Model	Layers	Parameters	Latency (ms) ↓	GFLOPS ↓	PSNR ↑
GS-LRM	24	307M	88.24	5047	28.10
LVSM Enc-Dec	6+18	177M	70.88	6042	28.58
LVSM Dec-Only	24	177M	109.37	8523	29.67
Ours (inference)	12+12	199M	24.78	1325	30.61

resource consumption exhibits a slow growth, maintaining high efficiency even with many views. In contrast, while the LVSM Encoder-Decoder shows a moderate increase in cost, the LVSM Decoder-Only variant suffers from a severe computational growth. This efficiency gap is substantial: with 16 input views, our model is 14.9x faster and consumes 50% less memory than LVSM Decoder-Only.

Incremental Inference. Fig. 9a indicate that the time and memory required for the incremental coming input views is nearly constant for Efficient-LVSM. Conversely, both LVSM baselines exhibit a clear growth in latency and memory consumption.

Training Convergence Speed. As in Fig. 9 (b), Efficient-LVSM demonstrates a steeper learning curve. It successfully reaches the final performance plateau of the LVSM baseline while consuming only **half the computational budget (GPU hours)**.

Table 4: **Ablation Study of REPA Distillation.**

Category	**Configuration**	**PSNR ↑**	**LPIPS ↓**	**SSIM ↑**
Without REPA Distillation (Baseline)		26.02	0.1481	0.8483
Ablation on REPA Hyperparameters				
Loss Function	Smooth L1	26.81	0.1349	0.8562
	L2	26.39	0.1366	0.8571
	Cosine	26.30	0.1374	0.8542
Distillation Target	Input Tokens Only	26.35	0.1367	0.8569
	Target Tokens Only	26.27	0.1452	0.8536
	Both Input & Target	26.60	0.1256	0.8642
DINOv3 Source Layer	Layer 8	26.60	0.1256	0.8642
	Layer 10	26.28	0.1441	0.8540
	Layer 12	26.11	0.1416	0.8503

3.5 ABLATION STUDIES

All ablation experiments use a smaller 6+6 encoder-decoder configuration to save budget.

Co-refinement of Encoder-Decoder Structure. As in Table 5 (a), self-then-cross attention yields 0.79 dB PSNR improvement compared to cross-attention only in decoder. Further, adopting encoder-decoder co-refinement gives 1.28 dB PSNR gains.

Applicability of REPA Distillation. As in Table 5 (b), applying REPA to Efficient-LVSM brings a substantial gain of 0.8 dB while applying to LSVM only brings 0.16 dB improvement. In Table 4, we study the configuration of REPA. We find that Smooth L1 loss works the best, possibly due to its absolute approximation to DINOv3 features instead of relative approximation as cosine similarity. We confirm that distillation for both input and target are useful. DINOv3's middle layer features instead of the final layers are most helpful, aligning with findings in Siméoni et al. (2025).

Influcne of Model Size. As in Table 5 (c), increasing model size consistently improves reconstruction quality, aligning with Jin et al. (2025), demonstrating the potential of feedforward models.

Size and Speed Comparison. As in Table 5 (d), Efficient LVSM achieves 4x faster inference with 0.94 dB higher PSNR compared to state-of-the-art LVSM decoder-only model.

3.6 Zero-Shot Generalization to the Number of Input Views.

As in Fig. 10, Efficient-LVSM and LVSM both could benefit from more views even not trained under such data, thanks to the set operator - Transformer. Efficient LVSM constantly outperforms LVSM under all view settings while the gap is gradually reduced, since the reconstruction becomes easier with more input views.

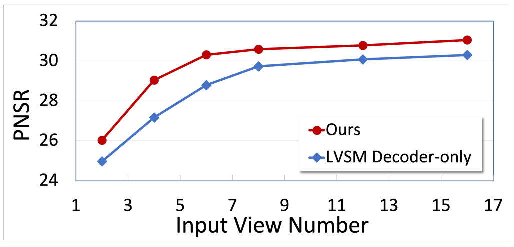


Figure 10: **Zero-Shot Generalization to Input View Count.** We train with 4 input views and test on varying number of views.

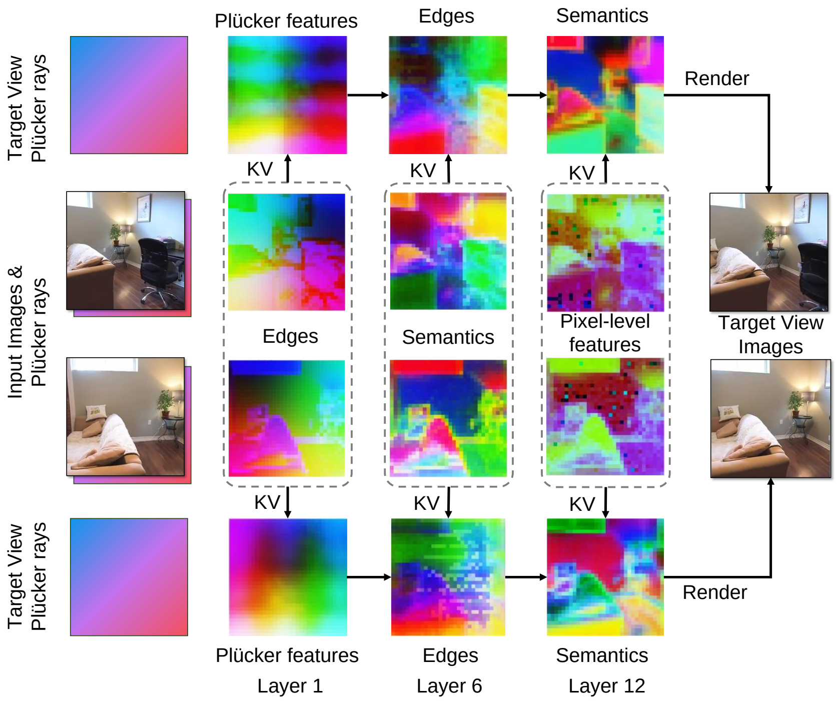


Figure 11: **PCA Visualization of Input and Target Views Features at Different Layers.**

4 Visualization

In Fig. 11, we visualize the features of Efficient-LVSM trained on RealEstate10K. We could observe that from the initial layer (Layer 1) to the middle layer (Layer 6), the features contain more and more semantics. From the middle layer (Layer 6) to the last layer (Layer 12), the features becomes similar to the final output - RGB images. The evolving process demonstrates the effectiveness of the proposed co-refinement structure to extract features from all levels.

5 Conclusion

In this work, we present a systematic analysis for issues of existing Transformer based NVS feedforward model. Based on the analysis, we derive Efficient-LVSM, a decoupled dual-stream architecture. Comprehensive experiments demonstrate that the proposed structure not only performs better but also achieves significant speed up for training convergence and inference latency.

Ethics Statement. Our research aims to advance the field of computer vision and does not present immediate, direct negative social impacts. We believe our work has the potential for a positive impact by improving. The dataset used in this study is publicly available and have been widely adopted by the community for academic research. All data was handled in accordance with their specified licenses and terms of use. We did not use any personally identifiable or sensitive private information. We have focused our evaluation on standard academic benchmarks. We encourage future research building upon our work to consider the specific ethical implications of their target applications.

Reproducibility Statement. To ensure the reproducibility of our research, we provide a comprehensive description of our methodology, implementation details, and experimental setup in the paper. Furthermore, we commit to making our code, pre-trained models, and experiment configurations publicly available upon publication of this paper.

References

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*, pp. 19457–19467, 2024.

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, 2024. URL `https://arxiv.org/abs/2403.14627`.

Inchang Choi, Orazio Gallo, Alejandro Troccoli, Min H Kim, and Jan Kautz. Extreme view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7781–7790, 2019.

Jasmine Collins, Shubham Goel, Kenan Deng, Achleshwar Luthra, Leon Xu, Erhan Gundogdu, Xi Zhang, Tomas F. Yago Vicente, Thomas Dideriksen, Himanshu Arora, Matthieu Guillaumin, and Jitendra Malik. Abo: Dataset and benchmarks for real-world 3d object understanding, 2022. URL `https://arxiv.org/abs/2110.06199`.

Paul E. Debevec, Camillo Jose Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, 1996. URL `https://api.semanticscholar.org/CorpusID:2609415`.

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*, pp. 13142–13153, 2023.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ArXiv*, abs/2010.11929, 2020. URL `https://api.semanticscholar.org/CorpusID:225039882`.

Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas B. McHugh, and Vincent Vanhoucke. Google scanned objects: A high-quality dataset of 3d scanned household items, 2022. URL `https://arxiv.org/abs/2204.11918`.

Yilun Du, Cameron Smith, Ayush Tewari, and Vincent Sitzmann. Learning to render novel views from wide-baseline stereo pairs, 2023. URL `https://arxiv.org/abs/2304.08463`.

Will Gao, Dilin Wang, Yuchen Fan, Aljaz Bozic, Tuur Stuyck, Zhengqin Li, Zhao Dong, Rakesh Ranjan, and Nikolaos Sarafianos. 3d mesh editing using masked lrms. *arXiv preprint arXiv:2412.08641*, 2024.

Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.

Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996. URL `https://api.semanticscholar.org/CorpusID:2036193`.

Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (ToG)*, 37(6):1–15, 2018.

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, 2024. URL `https://arxiv.org/abs/2311.04400`.

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. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=QQBPWtvtcn`.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023. URL `https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/`.

Jiahao Li, Hao Tan, Kai Zhang, Zexiang Xu, Fujun Luan, Yinghao Xu, Yicong Hong, Kalyan Sunkavalli, Greg Shakhnarovich, and Sai Bi. Instant3d: Fast text-to-3d with sparse-view generation and large reconstruction model, 2023. URL `https://arxiv.org/abs/2311.06214`.

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, 2020. URL `https://arxiv.org/abs/2003.08934`.

Julius Plucker. Xvii. on a new geometry of space. *Philosophical Transactions of the Royal Society of London*, pp. 725–791, 1865.

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*, pp. 6229–6238, 2022.

Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, Francisco Massa, Daniel Haziza, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Hervé Jégou, Patrick Labatut, and Piotr Bojanowski. DINOv3, 2025. URL `https://arxiv.org/abs/2508.10104`.

Mohammed Suhail, Carlos Esteves, Leonid Sigal, and Ameesh Makadia. Generalizable patch-based neural rendering. In *European Conference on Computer Vision*, pp. 156–174. Springer, 2022.

Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. URL `https://api.semanticscholar.org/CorpusID:13756489`.

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

Junqi You, Chieh Hubert Lin, Weijie Lyu, Zhengbo Zhang, and Ming-Hsuan Yang. Instainpaint: Instant 3d-scene inpainting with masked large reconstruction model. *NeauIPS*, 2025.

Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images, 2021. URL `https://arxiv.org/abs/2012.02190`.

Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. In *International Conference on Learning Representations*, 2025.

Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *ArXiv*, abs/1311.2901, 2013. URL `https://api.semanticscholar.org/CorpusID:3960646`.

Kai Zhang, Sai Bi, Hao Tan, Yuanbo Xiangli, Nanxuan Zhao, Kalyan Sunkavalli, and Zexiang Xu. Gs-lrm: Large reconstruction model for 3d gaussian splatting, 2024. URL `https://arxiv.org/abs/2404.19702`.

Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. In *SIGGRAPH*, 2018.

A USE OF LARGE LANGUAGE MODELS (LLMS) STATEMENT

During the preparation of this manuscript, we utilized Large Language Models (LLMs), as a writing assistance tool. The use of LLMs was limited to improving the grammar, clarity, and readability of the text. This includes tasks such as rephrasing sentences for better flow, correcting spelling and grammatical errors, and ensuring stylistic consistency. The core scientific ideas, experimental design, results, and conclusions presented in this paper are entirely our own. LLMs were not used to generate any of the primary scientific content or interpre- tations. The final version of the manuscript was thoroughly reviewed and edited by all authors, who take full responsibility for its content and originality.

B RELATED WORKS

Generalizable Novel View Synthesis. The ability to synthesize novel views from a sparse set of images is a long-standing goal in computer vision. Pioneering approaches such as image-based rendering (IBR) blend reference images based on proxy geometries (Debevec et al., 1996; Gortler et al., 1996). Early deep learning based methods predict blending weights or depth maps (Hedman et al., 2018; Choi et al., 2019). Generalizable neural radiance fields models like PixelNeRF (Yu et al., 2021) and MVSNeRF (Chen et al., 2024) pioneered the use of 3D-specific inductive biases.

Transformer-based Large Reconstruction Models. Recently, the field has gravitated towards leveraging the scalability and power of the Transformer architecture (Vaswani et al., 2017) to create Large Reconstruction Models (LRMs)(Hong et al., 2024; Wei et al., 2024; Li et al., 2023; Gao et al., 2024; You et al., 2025). These models are trained on vast datasets to learn generic 3D priors. For instance, Triplane-LRM(Li et al., 2023) and GS-LRM (Zhang et al., 2024) learn to map sparse input images to explicit 3D representations like triplane NeRFs or 3D Gaussian Splatting primitives.

View Synthesis without Explicit 3D Representations. A compelling line of research explores the possibility of performing novel view synthesis in a purely “geometry-free” manner. Early attempts such as Scene Representation Transformers (SRT) (Sajjadi et al., 2022), introduced the idea of using a Transformer to learn a latent scene representation. Large View Synthesis Model (LVSM) (Jin et al., 2025) employs a single, monolithic Transformer to process all input and target tokens.

C TRAINING AND IMPLEMENTATION DETAILS

Training Setup. We train Efficient-LVSM with a constant learning rate schedule with a warmup of 2500 iterations. Following LVSM (Jin et al., 2025), we use AdamW optimizer and the β_1 and β_2 are 0.9 and 0.95 respectively. We also employ a weight decay of 0.5 on all parameters of the LayerNorm layers. Unless noted, our models have 12 encoder layers and 12 decoder layers, which is the same as LVSM.

Dataset-Specific Schedules. For object-level dataset, we use 4 input views and 8 target views with 64 A100 80G GPU. We first train with the resolution of 256 for 3 days. Then we finetune the model with the resolution of 512 for 2 days with a learning rate of $4e-5$. For the scene-level dataset, we train with 2 input views and 3 target views. We first train with the resolution of 256 with 2 days and then finetune it with the resolution of 512 for 1 day.

REPA Distillation Details. We use the DINOv3-ViT-B/16 model (Siméoni et al., 2025) as the pre-trained teacher. We use the output features from the 8th transformer layer of DINOv3 as the distillation target. These teacher features are aligned with the output of a specific layer in our student model, which varies by its size: for our main 12+12 layer models, we align with the 3rd layer's output, while for the smaller 6+6 layer models used in ablations, we align with the 2nd layer. The alignment is performed via a 3-layer MLP projector and optimized using the Smooth L1 loss (Girshick, 2015).