StreamLRM: Test-Time Trained Memory for Streaming Multi-view 4D Reconstruction

Anonymous authors
Paper under double-blind review

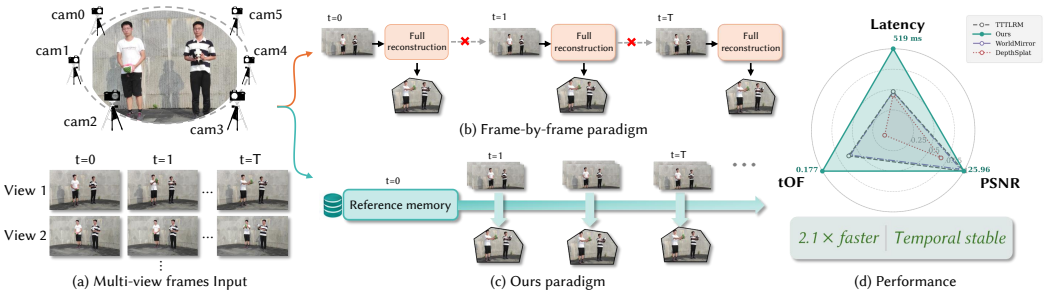


Figure 1: **StreamLRM extends static LRMs to streaming volumetric video reconstruction.** By sharing a reference scene memory across time while preserving current-frame processing, StreamLRM avoids repeated cross-view adaptation and achieves a 2.1× inference speedup with negligible loss in reconstruction quality.

Abstract

Large Reconstruction Models (LRMs) have recently shown remarkable performance in feed-forward static 3D reconstruction. In this work, we explore how such powerful static models can be extended to volumetric video reconstruction, where synchronized cameras share the same multi-view geometry across time. A straightforward frame-wise extension treats each timestamp as an independent static scene, overlooking this temporal structure and repeatedly performing expensive cross-view adaptation. We revisit TTT-based LRMs and reveal a key temporal asymmetry: scene content evolves with current observations, whereas fast-weight adaptation for cross-view interaction exhibits substantial temporal redundancy. Based on this observation, we propose StreamLRM, a simple training-free framework that temporally decouples TTT Update and Apply. We perform Update only at a reference timestamp and share the resulting fast weights across time, while each timestamp retains its own current-frame processing to capture evolving scene content. In this way, StreamLRM effectively extends pretrained static TTT-based LRMs to volumetric video without additional temporal modules or retraining. Surprisingly, this static-to-dynamic extension requires *less* computation rather than more, achieving multi-fold inference acceleration with negligible reconstruction quality degradation on diverse multi-view video benchmarks. Code will be released upon acceptance.

1 Introduction

Volumetric video reconstruction aims to recover temporally evolving 3D scenes from synchronized multi-view videos, enabling applications such as immersive telepresence, sports broadcasting, free-viewpoint video, and virtual reality Rauschnabel et al. (2022); Burdea & Coiffet (2003); Speicher et al. (2019); Jin et al. (2023). Unlike monocular dynamic reconstruction, volumetric capture systems typically employ synchronized cameras placed at fixed viewpoints, providing strong multi-view constraints for high-fidelity geometry reconstruction and novel-view synthesis. As a result, while the captured scene continuously evolves over time, the underlying camera geometry and cross-view configuration are naturally shared across timestamps. Efficiently exploiting this property, however,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

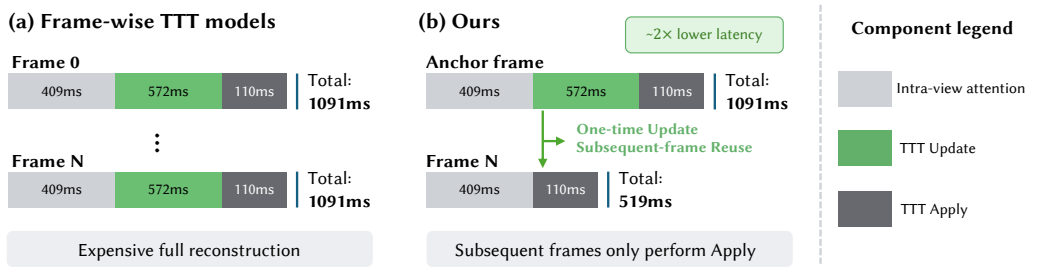


Figure 2: **Runtime breakdown of frame-wise reconstruction and StreamLRM.** TTT Update constitutes a major portion of per-frame inference, while StreamLRM performs it only once and shares the resulting scene memory across time, substantially reducing subsequent-frame latency.

remains challenging as modern capture systems scale to high-resolution inputs, many camera views, and long temporal sequences.

Recent Large Reconstruction Models have demonstrated impressive feed-forward 3D reconstruction capabilities, offering an attractive alternative to per-scene optimization. In particular, long-context models such as TTT-LRM Wang et al. (2026a), TTT-LVSM Zhang et al. (2025) and ZipMap Jin et al. (2026) introduce Test-Time Training (TTT) layers to efficiently aggregate large numbers of multi-view observations. Their computation broadly involves two complementary forms of interaction: intra-view processing extracts scene-dependent information from current observations, while TTT layers exchange information across views through an Update–Apply procedure. Specifically, multi-view features are written into fast weights through Update and subsequently retrieved by queries through Apply. Despite their strong performance on static scenes, extending these models to volumetric video remains non-trivial.



Figure 3: **Query-memory cross-swap on *Cook Spinach*.** (1) Horizontally, changing memory with queries fixed preserves the query-time content, suggesting that queries carry current-frame information. (2) Vertically, changing queries with memory fixed changes the reconstructed content accordingly.

As shown in Fig. 1 (b), a straightforward solution is to apply the static model independently at each timestamp. However, this frame-wise formulation provides no explicit mechanism for maintaining temporal consistency. each frame reconstructs its own scene state without sharing context across time, potentially introducing flickers unrelated to the underlying scene motion (Fig. 8 and Tab. 1). Alternatively, Nopo4D Balice et al. (2026) jointly reconstructs up to 10 frames per chunk to promote temporal consistency, but its growing memory requirements limit the temporal span that can be jointly processed.

Our key observation is that although scene content evolves over time, the multi-view camera geometry remains fixed across timestamps. Consequently, not every component of a static reconstruction model necessarily needs to evolve with the scene. This issue is particularly interesting for TTT-based LRMs, where fast weights are repeatedly rebuilt through expensive Update operations at every timestamp. We therefore ask: **can a pretrained static TTT-based LRM be extended to volumetric video by identifying which computation should evolve over time and which can be shared?**

Inspired by the register tokens that store global scene information in VGGT-Ω Wang et al. (2026b), we ask whether fast weights can serve as a shared scene memory. We first examine cross-scene memory transfer. In Fig. 4, combining target-scene queries with source-scene fast weights produces recognizable source-scene remnants, suggesting that fast weights retain scene-specific information. Across timestamps within the same scene, the reconstruction follows the query timestamp even with memory fixed. Together, these observations suggest a simple principle: **current queries capture evolving scene content, while fast weights provide reusable scene context.** Sharing this context across time connects otherwise independent per-frame reconstructions, enabling a pretrained static model to support streaming volumetric video reconstruction.

Based on this insight, we propose StreamLRM, a simple and training-free framework that extends pretrained TTT-based LRMs from static reconstruction to volumetric video. As shown in Fig. 6, instead of independently constructing fast weights at every timestamp, StreamLRM performs the TTT Update only at a reference timestamp and shares the resulting fast-weight state across time. Each subsequent timestamp retains its current-frame processing and queries to capture the evolving scene content, while directly applying these queries to the shared fast weights. We thus temporally decouple Update and Apply, yielding a simple inference paradigm: **Update Once, Apply Across Time**. Remarkably, this static-to-dynamic extension requires neither additional temporal modules nor video-specific training. Instead, by eliminating redundant cross-view adaptation, it is substantially more efficient than independently applying the original static model to every timestamp.

Extensive experiments on diverse volumetric video benchmarks validate both the effectiveness and efficiency of our static-to-dynamic formulation. The shared fast weights remain effective across different timestamps and substantial scene motion, enabling high-quality dynamic reconstruction while eliminating a major source of repeated computation. Consequently, StreamLRM preserves the strong reconstruction quality of pretrained TTT-based LRMs while achieving multi-fold inference acceleration over frame-wise reconstruction. Our contributions are summarized as follows:

(1) **An empirical finding of temporal redundancy.** We identify substantial functional redundancy in repeated fast-weight adaptation within TTT-based LRMs: current queries capture evolving scene content, while fast weights remain reusable across timestamps.

(2) **A plug-and-play strategy.** We introduce StreamLRM: *Update Once, Apply Across Time*. Reusing reference fast weights establishes a shared temporal context within pretrained static TTT-based LRM, enabling streaming dynamic reconstruction without new modules or retraining.

(3) **Validation across datasets and backbones.** Experiments on four multi-view video benchmarks and two TTT-based backbones demonstrate approximately $2\times$ faster inference while preserving reconstruction quality and improving temporal consistency.

2 Related Work

2.1 Feed-Forward 3D Reconstruction

Feed-forward 3D reconstruction has rapidly evolved from pairwise geometry estimation to large-scale scene modeling. Geometry-oriented methods such as DUSt3R Wang et al. (2024) and VGGT Wang et al. (2025a) recover dense 3D structure and camera information directly from images, while follow-up works including DepthAnything Lin et al. (2025), π^3 Wang et al. (2025d), MASt3R Leroy et al. (2024), Fast3R Yang et al. (2025), Stream3R Lan et al. (2025), TTT3R Chen et al. (2025b), and AnySplat Jiang et al. (2025) further improve scalability and efficiency. In parallel, Large Reconstruction Models for novel-view synthesis, including LRM Hong et al. (2023), Depthsplat Xu et al. (2025a), GS-LRM Zhang et al. (2024b), PixelSplat Charatan et al. (2024), MVSplat Chen et al. (2024), LVSM Jin et al. (2024), Long-LRM Ziwen et al. (2025), WorldMirror Liu et al. (2025), TTT-LRM Wang et al. (2026a), and ZipMap Jin et al. (2026), directly infer renderable scene representations from sparse multi-view observations. Despite their impressive reconstruction capability, these models are primarily designed for static inputs and typically treat each input set independently. Directly extending them to volumetric video therefore requires repeating the full reconstruction for every timestamp, without exploiting the shared capture configuration across time.

2.2 Monocular and Multi-view 4D Reconstruction

Recent feed-forward 4D methods extend pretrained 3D priors to dynamic scenes. Monocular approaches such as MonST3R Zhang et al. (2024a), CUT3R Wang et al. (2025b), PAGE-4D Zhou et al. (2025), VGGT-4D Hu et al. (2025), 4DGT Xu et al. (2025b), DVGGT Chen et al. (2025a), and UFO-4D Hur et al. (2026) mainly focus on recovering geometry and motion from temporally varying monocular observations.

Multi-view volumetric video is typically captured by synchronized fixed cameras, providing strong geometric constraints and consistent viewpoints over time. Early methods rely on geometric priors, such as shape-from-silhouette Ahmed et al. (2008), deformable templates Carranza et al. (2003),

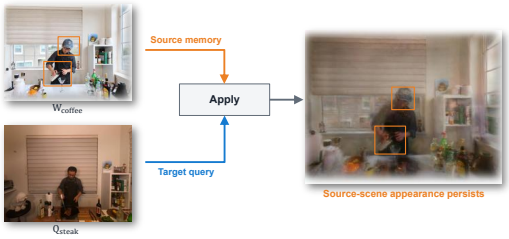


Figure 4: **Cross-scene fast-weight transfer.** Fast weights from Coffee Martini are transferred to Flame Steak with current queries, revealing that fast weights retain scene-specific information.



Figure 5: **Functional redundancy of TTT fast weights.** Given the same query Q_t, applying the current fast weights W_t and reference fast weights W_0 produces nearly identical feature responses.

and depth fusion Newcombe et al. (2015), while later neural methods adopt learned scene representations Lombardi et al. (2019); Bozic et al. (2020). Recent approaches, including HexPlane Cao & Johnson (2023), K-Planes Fridovich-Keil et al. (2023), Deformable-3DGS Yang et al. (2024), SCGS Huang et al. (2023), 4DGaussian Wu et al. (2024), and 4DGS Yang et al. (2023), model scene dynamics with neural or Gaussian representations Kerbl et al. (2023). SpaceTimeGS Li et al. (2024), Swift4D Wu et al. (2025), LongVolCap Xu et al. (2024), FreeTimeGS Wang et al. (2025c) and ClipGStream Liang et al. (2026) further target synchronized multi-view reconstruction. However, these methods generally require scene-specific optimization, motivating direct feed-forward reconstruction.

Existing feed-forward methods either process each timestamp independently, or introduce temporal modules, e.g., Nopo4D Balice et al. (2026) and SplatVoxel Wang et al. (2025e). In contrast, we explore sharing computation in a pretrained static reconstruction model across timestamps without explicit temporal modeling.

2.3 TTT-based Feed-Forward Reconstruction

Test-Time Training (TTT) Sun et al. (2024) represents input context through fast weights updated during inference. Given key-value pairs (k, v), TTT adapts a parametric memory f_W through an Update step and retrieves information using query q through Apply:

$$W \leftarrow W - \eta \nabla_W \mathcal{L}(f_W(k), v), \qquad o = f_W(q). \tag{1}$$

This formulation compresses input context into a fixed-size parametric memory rather than maintaining an explicit KV cache. TTT-LVSM Zhang et al. (2025) further improves its efficiency through large-chunk updates, making TTT suitable for long-context reconstruction.

Recent models such as TTT3R Chen et al. (2025b), TTT-LRM Wang et al. (2026a), and ZipMap Jin et al. (2026) incorporate this mechanism into feed-forward 3D reconstruction. Their backbones combine intra-view processing of image patches with cross-view interaction through fast-weight Update and Apply. When directly extended to volumetric video, however, the fast weights are reconstructed independently at every timestamp, despite all observations sharing the same multi-view capture geometry. Different from prior work, we reveal that this repeated fast-weight adaptation exhibits substantial temporal redundancy and can be amortized across timestamps without retraining.

3 Method

3.1 From Static LRM to Volumetric Video

We aim to extend a pretrained static Large Reconstruction Model (LRM) to efficient volumetric video reconstruction without introducing additional temporal modules or retraining. Given a synchronized multi-view video $\mathcal{V} = \{\mathcal{I}_t\}_{t=0}^{T-1}$ captured by N fixed cameras, where $\mathcal{I}_t = \{I_t^n\}_{n=1}^{N}$, a straightforward solution is to independently apply the static model to every timestamp. For a TTT-based LRM, the current observations are encoded into features $H_t = \mathcal{F}(\mathcal{I}_t)$, from which queries, keys, and values (Q_t, K_t, V_t) are obtained. The TTT layer constructs timestamp-specific fast weights $W_t = \mathrm{Update}(K_t, V_t)$ and reconstructs the scene as

$$\mathcal{G}_t = \mathcal{D}(\mathrm{Apply}(Q_t, \mathrm{Update}(K_t, V_t))), \tag{2}$$

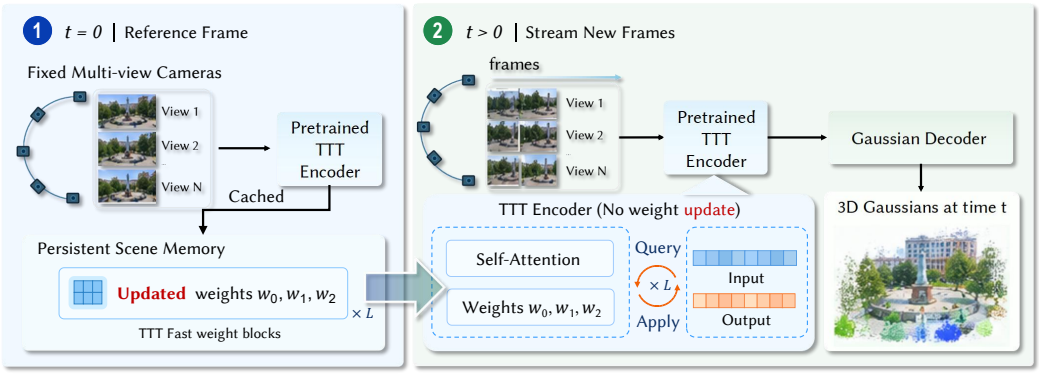


Figure 6: **Overview of StreamLRM.** Given a streaming multi-view sequence, our pipeline operates in two stages. At the reference timestamp, we perform the full TTT Update–Apply procedure and preserve the adapted fast weights as a shared cross-view state. For subsequent timestamps, we retain current-frame processing to capture evolving scene content, while skipping repeated Updates and directly applying current queries to the shared fast weights.

where $\mathcal{D}$ denotes the original reconstruction head.

While this frame-wise strategy produces a dynamic reconstruction sequence, it treats each timestamp as an independent static scene and repeatedly rebuilds the costly cross-view adaptation. This is potentially redundant in volumetric video: scene content evolves over time, whereas all timestamps are captured by the same fixed multi-view camera system. We therefore ask whether the adapted fast weights must be reconstructed at every timestamp.

Our empirical analysis reveals strong functional reusability of TTT fast weights. Fig. 5 shows that, given the same current query Q_t, the current fast weights W_t and reference fast weights W_0 produce nearly identical feature responses:

$$\mathrm{Apply}(Q_t, W_t) \approx \mathrm{Apply}(Q_t, W_0). \tag{3}$$

Importantly, this does not imply that W_t and W_0 are numerically identical; rather, they are functionally interchangeable under the current query. These observations reveal a key temporal asymmetry: current-frame features must evolve with the scene, whereas repeatedly rebuilding the fast-weight adaptation introduces only limited functional changes. In other words, **TTT Update is essential, but repeated Update across timestamps is largely redundant.**

3.2 Temporally Shared Fast-Weight Adaptation

Motivated by this observation, we reinterpret the adapted fast weights as a *temporally shared cross-view state*, rather than a disposable timestamp-specific intermediate. In Fig.6, given a reference timestamp r, we perform TTT Update once to construct

$$W_{\mathrm{shared}} = \mathrm{Update}(K_r, V_r). \tag{4}$$

This state is then preserved across the video. For each timestamp t, the current observations are still processed normally to produce the timestamp-specific query Q_t, but the query directly interacts with the shared fast weights:

$$\mathcal{G}_t = \mathcal{D}(\mathrm{Apply}(Q_t, W_{\mathrm{shared}})), \qquad t = 0, \ldots, T-1. \tag{5}$$

This formulation decouples two temporal behaviors of the original static LRM: scene-dependent computation evolves with each incoming observation, whereas cross-view adaptation is shared across time. The evolving Q_t preserves timestamp-specific geometry and appearance cues, while W_{shared} provides a common cross-view state across timestamps. As a result, otherwise independent per-frame reconstructions become connected through a persistent cross-view context, without introducing temporal attention, feature propagation, or motion estimation.

Dataset	TTTLRM			**Ours** (TTT backbone)			WorldMirror			DepthSplat		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
SelfVolCap	20.72	0.715	0.270	20.61	0.711	0.274	19.17	0.603	0.300	17.07	0.562	0.323
MeetRoom	30.14	0.932	0.187	30.09	0.932	0.186	26.91	0.906	0.137	20.04	0.719	0.345
ENeRF	23.99	0.528	0.325	24.08	0.525	0.331	25.11	0.660	0.272	10.79	0.162	0.637
N3DV	29.23	0.896	0.157	29.05	0.894	0.161	27.81	0.873	0.153	22.46	0.730	0.323
Average	26.02	0.768	0.235	25.96	0.766	0.238	24.75	0.760	0.216	17.59	0.543	0.407
Efficiency and Temporal Consistency												
Temporal Consistency (tOF)↓	0.282			**0.177**			0.292			1.471		
Time (ms/frame)↓	1091.4			**519.8**			526.1			534.4		
Resolution	960×720			960×720			518×392			512×394		

Table 1: **Quantitative comparison across different datasets.** We report PSNR (↑), SSIM (↑), LPIPS (↓), Time (↓), and tOF (↓). For all methods, six uniformly sampled camera views are used as input, and the metrics are averaged over all frames. For a fair comparison, WorldMirror and DepthSplat use low-resolution rendering.

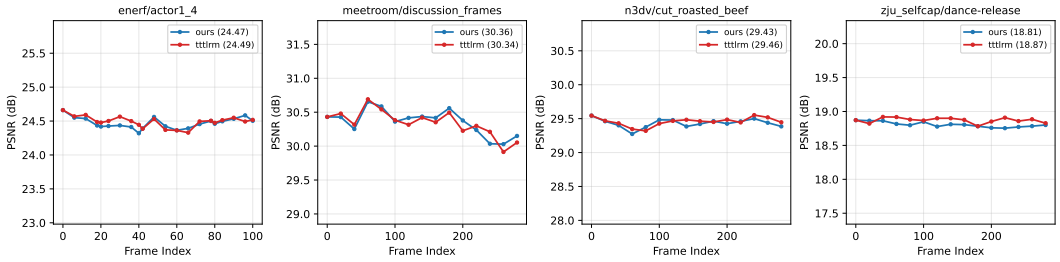


Figure 7: Temporal PSNR curves on four dynamic scenes. Our method maintains stable reconstruction quality over long video sequences without temporal degradation.

3.3 Efficient Dynamic Reconstruction

The resulting inference paradigm is simple: *Update Once, Apply Across Time*. At the reference timestamp, StreamLRM performs the original TTT Update to construct the shared fast weights. For subsequent timestamps, current-frame processing remains unchanged, while the repeated TTT Update is skipped and current queries directly Apply to the shared state. Thus, extending the static model to volumetric video requires no additional temporal computation; instead, it removes a major source of repeated computation.

Let C_{intra}, C_{U}, and C_{A} denote the costs of current-frame processing, TTT Update, and Apply, respectively. Conventional frame-wise inference requires $T(C_{\text{intra}} + C_{\text{U}} + C_{\text{A}})$, whereas StreamLRM requires only $T(C_{\text{intra}} + C_{\text{A}}) + C_{\text{U}}$. The amortized Update cost therefore decreases from C_{U} to C_{U}/T per timestamp and becomes negligible for long sequences.

Rather than introducing additional temporal processing, StreamLRM turns the temporal structure of volumetric video into a computational advantage: the shared multi-view capture geometry allows cross-view adaptation to be amortized across time. The resulting framework is training-free, introduces no additional parameters or temporal modules, and can be directly applied to pretrained TTT-based LRMs Wang et al. (2026a); Zhang et al. (2025); Jin et al. (2026).

4 Experiments

4.1 Experimental Setup

We evaluate StreamLRM on four widely used volumetric video datasets: N3DV Li et al. (2022b), MeetRoom Li et al. (2022a), ENeRF Lin et al. (2022), and SelfVolCap Xu et al. (2023). Unless otherwise specified, all experiments use six input views at a resolution of 960×720. Our experiments are designed to answer four key questions: (1) Why Can Fast Weights Be Reused? (2) What Are the Benefits of Reference-Based Persistent Memory? (3) Does Reference Fast-Weight Reuse Degrade Quality over Time? (4) Does Shared Fast-Weight State Improve Temporal Consistency? All experiments are conducted on a single NVIDIA A100 GPU with 80 GB of VRAM. **See the appendix and supplementary material for more experiments and rendering videos.** Code will be released upon acceptance.



(a) Input renderings (b) Temporal residual (ours) (c) Temporal residual (baseline)

Figure 8: **Temporal residual in static background regions.** Our method exhibits smaller temporal residuals in the highlighted background regions than frame-wise TTT-LRM.

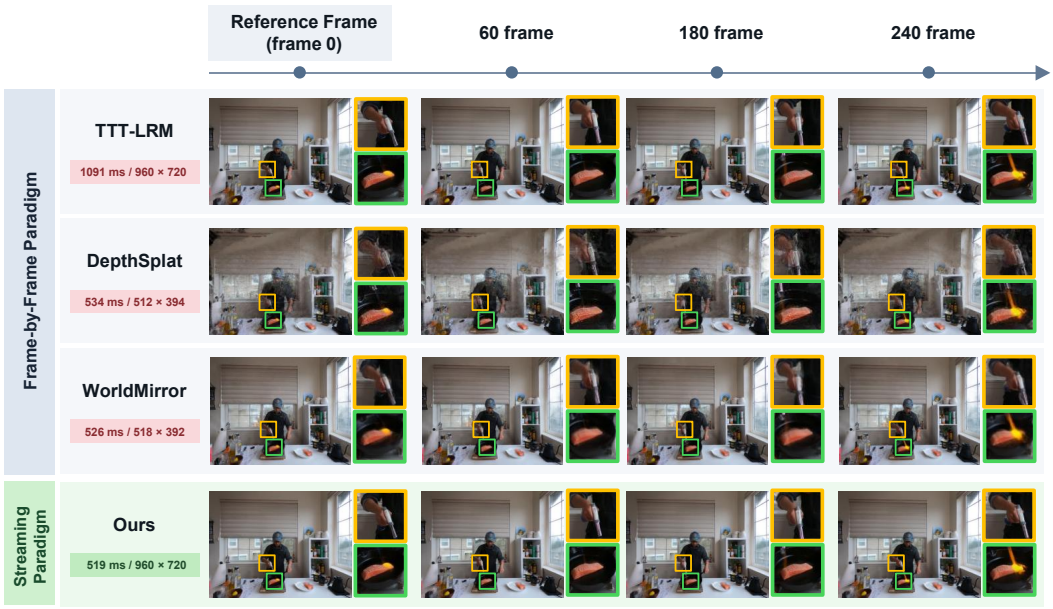


Figure 9: Qualitative comparison on long-term dynamic novel view synthesis. Starting from the same reference frame, we evaluate different methods at increasing temporal intervals. The inference latency and input resolution are reported on the left.

Q1: Why Can Fast Weights Be Reused?

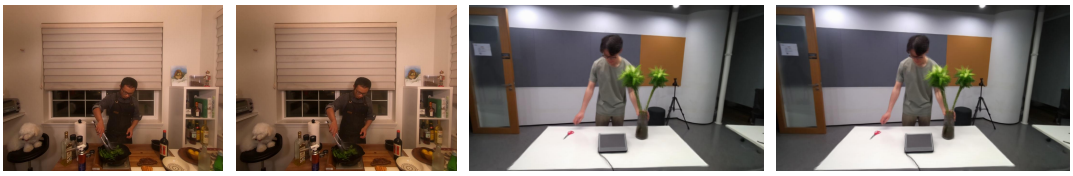
A natural question is why fast weights constructed at the reference timestamp remain effective for subsequent dynamic frames. We hypothesize that this reusability arises from the temporal asymmetry between evolving scene observations and the shared multi-view capture configuration.

To verify this property, we fix the query tokens Q_t from a later timestamp and compare the Apply outputs using the timestamp-specific fast weights W_t and the reference fast weights W_0. In Tab. 4, the cosine similarity between the two outputs remains above 98% even at a temporal distance of 300 frames. Importantly, this does not imply that W_t and W_0 are numerically identical; rather, their differences introduce only limited functional changes when interacting with the same current query. Fig.5 provides a qualitative view of this behavior. Given the same Q_t, replacing W_t with W_0 produces nearly indistinguishable feature responses.

Fig. 3 further shows that swapping memories with queries fixed largely preserves scene content, whereas changing queries with memory fixed changes the reconstructed results. This explains why a reference fast-weight state can be shared across time while current queries continue to capture the evolving scene content.

Q2: What Are the Benefits of Reference-Based Persistent Memory?

Reference-based persistent memory amortizes the expensive TTT Update across time. We perform Update once at the reference timestamp and preserve the resulting fast-weight state, while subsequent frames retain their current-frame processing and directly Apply current queries to the shared state. Consequently, the persistent state has a fixed size independent of sequence length, making the formulation naturally suited to causal streaming inference.



(a) TTTLVSM (b) TTTLVSM + Ours (c) TTTLRM (d) TTTLRM + Ours

Figure 10: **Model sweep.** Visualization of applying our method to different feed-forward reconstruction models, demonstrating its generality across existing approaches.

As shown in Tab. 1, this temporal sharing preserves nearly identical reconstruction quality to frame-by-frame TTT-LRM (mean PSNR 25.96 vs. 26.02) while reducing per-frame latency from 1091.4 ms to 519.8 ms at 960×720, corresponding to a 2.1× speedup. Qualitatively, Fig. 9 shows competitive reconstruction results, while Fig. 8 shows reduced temporal variation in the static background regions.

Q3: Does Reference Fast-Weight Reuse Degrade Quality over Time?

A natural concern is whether continuously reusing the reference fast weights causes reconstruction quality to degrade as the temporal distance increases. We therefore evaluate per-frame reconstruction quality over a 300-frame sequence while keeping the reference fast weights fixed throughout.

As shown in Fig. 7, the per-frame PSNR remains stable throughout the sequence, with no noticeable downward trend as the temporal distance from the reference frame increases. This suggests that reference fast-weight reuse does not accumulate reconstruction errors over the evaluated temporal horizon, further supporting its long-term functional reusability.

Q4: Does Shared Fast-Weight State Improve Temporal Consistency?

We evaluate the temporal consistency of the reconstructed videos using temporal optical flow error (tOF), following TecoGAN Chu et al. (2020). Let $\hat{I}_t$ and I_t denote the rendered and ground-truth images at timestamp t for the same evaluation viewpoint. Given an optical flow estimator $\mathcal{F}$, we define

$$\text{tOF} = \frac{1}{|\mathcal{P}|\,|\Omega|} \sum_{t\in\mathcal{P}} \sum_{\mathbf{x}\in\Omega} \left\| \mathcal{F}(\hat{I}_{t-1}, \hat{I}_t)(\mathbf{x}) - \mathcal{F}(I_{t-1}, I_t)(\mathbf{x}) \right\|_2, \tag{6}$$

where $\mathcal{P}$ denotes the evaluated consecutive-frame pairs and Ω is the evaluated pixel domain. Lower tOF indicates better agreement with the ground-truth temporal motion.

As shown in Tab. 1, StreamLRM achieves substantially lower tOF than TTT-LRM, demonstrating improved temporal consistency while preserving comparable per-frame reconstruction quality. Unlike independent frame-wise inference, which reconstructs a new fast-weight state at every timestamp, StreamLRM connects all timestamps through the same cross-view state. This shared reference provides a consistent cross-view context throughout the sequence, reducing temporal variation introduced by independent per-frame adaptation and producing more coherent motion across frames.

4.2 Ablation Study

Model Sweep. We evaluate StreamLRM on different TTT-based reconstruction backbones. As shown in Tab.2 and Fig.10, our method maintains comparable reconstruction quality while improving temporal consistency across both evaluated models, demonstrating that the proposed temporal fast-weight sharing is not specific to a particular backbone.

Table 2: **Model sweep.** our method achieves comparable performance across different reconstruction models. Experiments are conducted on N3DV.

Method	PSNR	SSIM	LPIPS	Time(ms/frame)	tOF
TTT-LVSM	29.70	0.911	0.132	985.9	0.212
TTT-LVSM + Ours	29.56	0.908	0.135	481.7	0.169
TTT-LRM	29.23	0.896	0.157	1091.4	0.264
TTT-LRM + Ours	29.05	0.894	0.161	519.8	0.180

Is TTT Update Necessary at Every Timestamp? We first investigate the role of TTT Update and whether it needs to be repeated for every timestamp. As shown in table 3, directly using the pretrained initialization W_{init} without test-time adaptation leads to severe performance degradation across all datasets. On average, performing Update improves PSNR from 12.09 to 26.02 dB. This confirms that TTT Update is essential for constructing effective fast weights and cannot simply be removed.

Table 3: Effect of Update. W_{init}:pretrained initialization; W_{update}:adaptation using current observations.

Dataset	Fast weights	PSNR ↑	SSIM ↑	LPIPS ↓
N3DV	W_{init}	13.07	0.562	0.720
	W_{update}	**29.23**	**0.896**	**0.157**
MeetRoom	W_{init}	11.70	0.675	0.787
	W_{update}	**30.14**	**0.932**	**0.187**
ENeRF	W_{init}	10.60	0.200	0.664
	W_{update}	**23.99**	**0.528**	**0.325**
SelfVolCap	W_{init}	13.00	0.490	0.738
	W_{update}	**20.72**	**0.715**	**0.270**
Mean	W_{init}	12.09	0.482	0.727
	W_{update}	**26.02**	**0.768**	**0.235**

Table 4: Functional similarity of TTT Apply outputs using current and reference fast weights. Q_t is fixed, while W_t is replaced by W_0. ΔPSNR denotes reference-weight reuse minus frame-wise inference at the same timestamp.

Dataset	Scene	t	Apply cosine similarity (%) ↑				ΔPSNR (dB)
			Layer 0	Layer 7	Layer 14	Layer 21	
SelfVolCap	Dance-release	100	99.97	99.90	98.95	99.95	−0.09
		300	99.98	99.75	98.90	99.95	−0.12
MeetRoom	Discussion	100	99.97	99.92	99.31	99.94	−0.09
		300	99.97	99.81	98.94	99.94	−0.11
ENeRF	Actor 1_4	50	99.96	99.76	98.79	99.92	+0.08
		100	99.96	99.81	98.56	99.93	+0.12
N3DV	Flame Salmon	100	99.98	99.96	99.76	99.94	−0.11
		300	99.90	99.97	99.67	99.94	−0.13

However, an important question is whether this costly adaptation must be repeated at every timestamp. To isolate the effect of fast weights, we fix the current query Q_t and compare the Apply responses produced by the timestamp-specific fast weights W_t and the reference fast weights W_0. As shown in Table 4, cosine similarity remains above 98.5%, while the absolute PSNR difference from frame-wise inference is at most 0.13 dB. Therefore, although adaptation is critical, repeatedly reconstructing the fast weights introduces only limited functional changes across time.

We further evaluate different fast weight update intervals by periodically refreshing the reference timestamp. As shown in Tab.5, using a single reference update for the entire 300-frame sequence only reduces PSNR by 0.12 dB while achieving over 2× speedup, demonstrating that frequent fast weight updates are unnecessary.

Together, these results reveal a key property of TTT-based reconstruction: **TTT Update is essential, but repeated Update is largely functionally redundant.** This observation directly motivates our strategy of performing Update once at a reference timestamp and reusing the resulting fast weights for subsequent frames.

Table 5: Effect of fast weight update frequency on streaming reconstruction over a 300-frame sequence.

Update interval	Reconstruction quality			Runtime	
	PSNR ↑	SSIM ↑	LPIPS ↓	Total (s) ↓	Avg. (s/frame) ↓
1 (baseline)	30.69	0.917	0.144	327.42	1.09
2	30.67	0.917	0.144	241.95	0.81
5	30.68	0.917	0.145	190.54	0.64
10	30.68	0.916	0.145	173.28	0.58
20	30.66	0.916	0.145	165.54	0.55
100	30.62	0.914	0.146	159.19	0.53
300 (ours)	30.57	0.914	0.147	155.28	0.52

5 Discussion & Limitation

Our simple inference reformulation extends static TTT-based LRMs to efficient volumetric video reconstruction without additional temporal modules, parameters, or video-specific training. This suggests that extending static LRMs to dynamic scenes may rely on identifying which computation should evolve or be shared over time, rather than complex temporal modeling. Perhaps surprisingly, turning a static LRM into a volumetric video reconstructor can be achieved not by performing more computation, but by eliminating redundant computation.

Limitation. StreamLRM remains constrained by its pretrained static backbone. Its fixed reference memory may become less effective under substantial scene changes, such as the arrival of new objects or major appearance changes. Adaptive memory refresh is a promising direction for maintaining quality while preserving efficiency.

6 Conclusion

In this work, We reveal substantial temporal redundancy in TTT-based fast-weight adaptation: scene content evolves over time, while multi-view geometry remains shared across timestamps. Based on this observation, we propose StreamLRM, a training-free framework that performs the costly TTT Update only once and reuses the resulting fast weights for subsequent frames. Extensive experiments demonstrate multi-fold inference acceleration with negligible reconstruction quality degradation. Our findings show that not all computation needs to evolve with scene observations, enabling efficient feed-forward volumetric video reconstruction.

AI USE STATEMENT

In this work, we used generative AI tools to assist with translation. We did not use generative AI tools for any other tasks requiring disclosure, including developing the research methodology, designing experiments, implementing the proposed method, or interpreting results; the remaining required disclosure tasks are not applicable to this work.

Additionally, we used generative AI tools to assist with drafting and editing parts of the manuscript, improving its readability, and writing a small number of utility functions. The core method and implementation were developed by the authors. All AI-assisted text and code were reviewed and verified by the authors. We take full responsibility for the final content of this work, including text and code produced with the aid of generative AI.

REFERENCES

Naveed Ahmed, Christian Theobalt, Christian Rossl, Sebastian Thrun, and Hans-Peter Seidel. Dense correspondence finding for parametrization-free animation reconstruction from video. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8. IEEE, 2008.

Matteo Balice, Yanik Kunzi, Chenyangguang Zhang, Matteo Matteucci, Marc Pollefeys, and Sunghwan Hong. No pose, no problem in 4d: Feed-forward dynamic gaussians from unposed multi-view videos, 2026. URL https://arxiv.org/abs/2605.22190.

Aljaz Bozic, Michael Zollhofer, Christian Theobalt, and Matthias Nießner. Deepdeform: Learning non-rigid rgb-d reconstruction with semi-supervised data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7002–7012, 2020.

Grigore C Burdea and Philippe Coiffet. *Virtual reality technology*. John Wiley & Sons, 2003.

Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 130–141, 2023.

Joel Carranza, Christian Theobalt, Marcus A Magnor, and Hans-Peter Seidel. Free-viewpoint video of human actors. *ACM transactions on graphics (TOG)*, 22(3):569–577, 2003.

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.

Xiaoxue Chen, Ziyi Xiong, Yuantao Chen, Gen Li, Nan Wang, Hongcheng Luo, Long Chen, Haiyang Sun, BING WANG, Guang Chen, et al. Feedforward 4d reconstruction for dynamic driving scenes using unposed images. 2025a.

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

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*, pp. 370–386. Springer, 2024.

Mengyu Chu, You Xie, Jonas Mayer, Laura Leal-Taixé, and Nils Thuerey. Learning temporal coherence via self-supervision for gan-based video generation. *ACM Transactions on Graphics (TOG)*, 39(4):75–1, 2020.

Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12479–12488, 2023.

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.

Yu Hu, Chong Cheng, Sicheng Yu, Xiaoyang Guo, and Hao Wang. Vggt4d: Mining motion cues in visual geometry transformers for 4d scene reconstruction, 2025. URL https://arxiv.org/abs/2511.19971.

Yi-Hua Huang, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. Sc-gs: Sparse-controlled gaussian splatting for editable dynamic scenes. *arXiv preprint arXiv:2312.14937*, 2023.

Junhwa Hur, Charles Herrmann, Songyou Peng, Philipp Henzler, Zeyu Ma, Todd Zickler, and Deqing Sun. Ufo-4d: Unposed feedforward 4d reconstruction from two images. *arXiv preprint arXiv:2602.24290*, 2026.

Lihan Jiang, Yucheng Mao, Linning Xu, Tao Lu, Kerui Ren, Yichen Jin, Xudong Xu, Mulin Yu, Jiangmiao Pang, Feng Zhao, et al. Anysplat: Feed-forward 3d gaussian splatting from unconstrained views. *ACM Transactions on Graphics*, 44(6):1–16, 2025.

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.

Haian Jin, Rundi Wu, Tianyuan Zhang, Ruiqi Gao, Jonathan T. Barron, Noah Snavely, and Aleksander Holynski. Zipmap: Linear-time stateful 3d reconstruction via test-time training, 2026. URL https://arxiv.org/abs/2603.04385.

Yili Jin, Kaiyuan Hu, Junhua Liu, Fangxin Wang, and Xue Liu. From capture to display: A survey on volumetric video. *arXiv preprint arXiv:2309.05658*, 2023.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4):139–1, 2023.

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.

Vincent Leroy, Yohann Cabon, and Jérôme Revaud. Grounding image matching in 3d with mast3r. In *Proc. of ECCV*, pp. 71–91. Springer, 2024.

Lingzhi Li, Zhen Shen, Zhongshu Wang, Li Shen, and Ping Tan. Streaming radiance fields for 3d video synthesis. *Advances in Neural Information Processing Systems*, 35:13485–13498, 2022a.

Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, et al. Neural 3d video synthesis from multi-view video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5521–5531, 2022b.

Zhan Li, Zhang Chen, Zhong Li, and Yi Xu. Spacetime gaussian feature splatting for real-time dynamic view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8508–8520, 2024.

Jie Liang, Jiahao Wu, Chao Wang, Jiayu Yang, Xiaoyun Zheng, Kaiqiang Xiong, Zhanke Wang, Jinbo Yan, Feng Gao, and Ronggang Wang. Clipgstream: Clip-stream gaussian splatting for any length and any motion multi-view dynamic scene reconstruction. *arXiv preprint arXiv:2604.13746*, 2026.

Haotong Lin, Sida Peng, Zhen Xu, Yunzhi Yan, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Efficient neural radiance fields for interactive free-viewpoint video. In *SIGGRAPH Asia Conference Proceedings*, 2022.

Haotong Lin, Sili Chen, Jun Hao Liew, Donny Y. Chen, Zhenyu Li, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth anything 3: Recovering the visual space from any views. *arXiv preprint arXiv:2511.10647*, 2025.

Yifan Liu, Zhiyuan Min, Zhenwei Wang, Junta Wu, Tengfei Wang, Yixuan Yuan, Yawei Luo, and Chunchao Guo. Worldmirror: Universal 3d world reconstruction with any-prior prompting. *arXiv preprint arXiv:2510.10726*, 2025.

Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *arXiv preprint arXiv:1906.07751*, 2019.

Richard A Newcombe, Dieter Fox, and Steven M Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 343–352, 2015.

Philipp A Rauschnabel, Reto Felix, Chris Hinsch, Hamza Shahab, and Florian Alt. What is xr? towards a framework for augmented and virtual reality. *Computers in human behavior*, 133:107289, 2022.

Maximilian Speicher, Brian D Hall, and Michael Nebeling. What is mixed reality? In *Proceedings of the 2019 CHI conference on human factors in computing systems*, pp. 1–15, 2019.

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.

Chen Wang, Hao Tan, Wang Yifan, Zhiqin Chen, Yuheng Liu, Kalyan Sunkavalli, Sai Bi, Lingjie Liu, and Yiwei Hu. tttlrm: Test-time training for long context and autoregressive 3d reconstruction, 2026a. URL https://arxiv.org/abs/2602.20160.

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proc. of CVPR*, 2025a.

Jianyuan Wang, Minghao Chen, Shangzhan Zhang, Nikita Karaev, Johannes Schönberger, Patrick Labatut, Piotr Bojanowski, David Novotny, Andrea Vedaldi, and Christian Rupprecht. Vggt-Ω. In *Proc. of CVPR*, 2026b.

Qianqian Wang, Yifei Zhang, Aleksander Holynski, Alexei A Efros, and Angjoo Kanazawa. Continuous 3d perception model with persistent state. In *Proc. of CVPR*, pp. 10510–10522, 2025b.

Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *Proc. of CVPR*, pp. 20697–20709, 2024.

Yifan Wang, Peishan Yang, Zhen Xu, Jiaming Sun, Zhanhua Zhang, Yong Chen, Hujun Bao, Sida Peng, and Xiaowei Zhou. Freetimegs: Free gaussian primitives at anytime anywhere for dynamic scene reconstruction. In *CVPR*, 2025c. URL https://zju3dv.github.io/freetimegs.

Yifan Wang, Jianjun Zhou, Haoyi Zhu, Wenzheng Chang, Yang Zhou, Zizun Li, Junyi Chen, Jiangmiao Pang, Chunhua Shen, and Tong He. π^3: Scalable permutation-equivariant visual geometry learning. *arXiv preprint arXiv:2507.13347*, 2025d.

Yiming Wang, Lucy Chai, Xuan Luo, Michael Niemeyer, Manuel Lagunas, Stephen Lombardi, Siyu Tang, and Tiancheng Sun. Splatvoxel: History-aware novel view streaming without temporal training, 2025e. URL https://arxiv.org/abs/2503.14698.

Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20310–20320, 2024.

Jiahao Wu, Rui Peng, Zhiyan Wang, Lu Xiao, Luyang Tang, Jinbo Yan, Kaiqiang Xiong, and Ronggang Wang. Swift4d: Adaptive divide-and-conquer gaussian splatting for compact and efficient reconstruction of dynamic scene. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=c1RhJVTPwT.

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*, pp. 16453–16463, 2025a.

Zhen Xu, Sida Peng, Haotong Lin, Guangzhao He, Jiaming Sun, Yujun Shen, Hujun Bao, and Xiaowei Zhou. 4k4d: Real-time 4d view synthesis at 4k resolution. 2023.

Zhen Xu, Yinghao Xu, Zhiyuan Yu, Sida Peng, Jiaming Sun, Hujun Bao, and Xiaowei Zhou. Representing long volumetric video with temporal gaussian hierarchy. *ACM Transactions on Graphics*, 43(6), November 2024. URL https://zju3dv.github.io/longvolcap.

Zhen Xu, Zhengqin Li, Zhao Dong, Xiaowei Zhou, Richard Newcombe, and Zhaoyang Lv. 4dgt: Learning a 4d gaussian transformer using real-world monocular videos. *Advances in neural information processing systems*, 2025b.

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*, pp. 21924–21935, 2025.

Zeyu Yang, Hongye Yang, Zijie Pan, Xiatian Zhu, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. *arXiv preprint arXiv:2310.10642*, 2023.

Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20331–20341, 2024.

Junyi Zhang, Charles Herrmann, Junhwa Hur, Varun Jampani, Trevor Darrell, Forrester Cole, Deqing Sun, and Ming-Hsuan Yang. Monst3r: A simple approach for estimating geometry in the presence of motion. *arXiv preprint arxiv:2410.03825*, 2024a.

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*, pp. 1–19. Springer, 2024b.

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, 2025. URL https://arxiv.org/abs/2505.23884.

Kaichen Zhou, Yuhan Wang, Grace Chen, Gaspard Beaudouin, Fangneng Zhan, Paul Pu Liang, and Mengyu Wang. Page-4d: Vggt-4d perception via disentangled pose and geometry estimation. In *The Fourteenth International Conference on Learning Representations*, 2025.

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*, 2025.

A Appendix

A.1 Detailed Inference Procedure

StreamLRM extends a pretrained TTT-based reconstruction model to synchronized multi-view video by reusing its adapted fast weights across timestamps. The pretrained model parameters remain unchanged. Only the inference schedule is modified: fast-weight Update is performed at the reference timestamp, while subsequent timestamps retain current-frame processing and fast-weight Apply.

Layer-wise Fast-Weight Reuse Let $\mathcal{L}_{\mathrm{TTT}}$ denote the set of TTT layers in the backbone. At the reference timestamp r, we execute the original model and cache the adapted fast weights separately for each layer:

$$\mathcal{W}_r = \left\{ W_r^{(\ell)} \right\}_{\ell \in \mathcal{L}_{\mathrm{TTT}}}. \tag{7}$$

Each cached state is reused only by its corresponding layer; fast weights are not shared between different layers.

For a subsequent timestamp t, the model processes the current multi-view observations through the backbone. At each TTT layer, it computes the current query and applies the cached reference state:

$$O_t^{(\ell)} = \mathrm{Apply}\left(Q_t^{(\ell)}, W_r^{(\ell)} \right). \tag{8}$$

The resulting features continue through the remaining backbone and the original reconstruction head. Thus, current-frame information is processed throughout the network even though fast-weight Update is skipped.

The query at a deeper layer is computed from the hidden features produced by the preceding layers of this inference path. StreamLRM therefore does not require a separate current-frame Update pass to obtain its queries.

Causal Streaming Operation For causal streaming, the first observed timestamp serves as the reference, i.e., $r = 0$. The reference states are constructed from the initial synchronized observations and then held fixed:

$$\mathcal{W}_t = \mathcal{W}_0, \qquad t > 0. \tag{9}$$

At timestamp t, reconstruction uses only the current observations, the cached reference states, and the pretrained model parameters. No future observations are required. The persistent fast-weight state does not grow with sequence length.

A.2 Computational Analysis

Let C_{U} denote the total fast-weight Update cost across the backbone, and let

$$C_{\mathrm{B}} = C_{\mathrm{intra}} + C_{\mathrm{A}} \tag{10}$$

collect the costs of current-frame processing and Apply. Under the cost decomposition used in the main paper, independent frame-wise inference over T timestamps requires

$$C_{\mathrm{frame}} = T(C_{\mathrm{B}} + C_{\mathrm{U}}). \tag{11}$$

StreamLRM performs Update once, giving

$$C_{\mathrm{stream}} = TC_{\mathrm{B}} + C_{\mathrm{U}}. \tag{12}$$

The amortized cost per timestamp is therefore

$$\overline{C}_{\mathrm{stream}} = C_{\mathrm{B}} + \frac{C_{\mathrm{U}}}{T}. \tag{13}$$

The corresponding idealized speedup is

$$S(T) = \frac{C_{\mathrm{B}} + C_{\mathrm{U}}}{C_{\mathrm{B}} + C_{\mathrm{U}}/T}. \tag{14}$$

For $T = 1$, no repeated Update can be removed. As T increases, the initialization cost is amortized and the speedup approaches

$$\lim_{T \to \infty} S(T) = 1 + \frac{C_{\mathrm{U}}}{C_{\mathrm{B}}}. \tag{15}$$

This expression also makes clear that the achievable acceleration depends on the fraction of the original runtime spent in Update.

Table 6: Evaluation sequences and camera splits. All listed input views are used at inference time, and camera 0 is held out for evaluation.

Dataset	Sequences	Frames/seq.	Input cameras	Test camera
N3DV	coffee_martini, cook_spinach, cut_roasted_beef, flame_salmon, flame_steak, sear_steak	300	1, 5, 6, 10, 15, 17	0
ENeRF	actor1_4, actor2_3, actor5_6	100	1, 4, 5, 7, 9, 11	0
MeetRoom	discussion, trimming	300	1, 4, 5, 7, 9, 11	0
SelfVolCap	corgi, dance, hair, yoga	*300*	1, 5, 6, 10, 15, 17	0

A.3 Datasets and evaluation protocol.

We evaluate StreamLRM on four multi-view video datasets: N3DV, ENeRF, MeetRoom, and SelfCap. We use all six N3DV sequences with 300 frames per sequence, all three ENeRF sequences with 100 frames per sequence, two MeetRoom sequences with 300 frames per sequence, and four SelfCap sequences with 300 frames per sequence. Table 6 summarizes the evaluated sequences and camera splits. For each dataset, we use a fixed set of six input cameras throughout each sequence and reserve camera 0 for novel-view evaluation. The input and evaluation camera sets are strictly disjoint. These camera splits define inference inputs and held-out evaluation views; no additional training is performed.

A.4 More experiments

Efficiency over Long Sequences. We further evaluate sequences of increasing length to assess the scalability of StreamLRM. Unlike frame-by-frame inference that performs full reconstruction at every timestamp, StreamLRM only incurs this cost during reference initialization and reuses computation for subsequent frames. As shown in Fig. 11, the initialization overhead is progressively amortized as the sequence length increases, leading to substantially lower per-frame computation and greater efficiency gains for long volumetric videos.

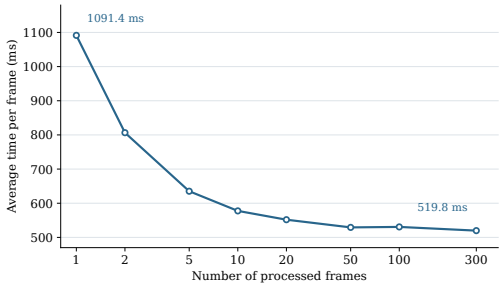


Figure 11: **Amortized inference cost.** Processing more frames amortizes the initial Update cost, reducing average latency toward 519.8 ms/frame. The horizontal axis uses a logarithmic scale.

GPU Memory Usage. We further evaluate the GPU memory consumption of StreamLRM during streaming inference. Despite processing high-resolution multi-view inputs, the peak memory usage typically remains below 24 GB, enabling StreamLRM to run on widely available consumer-grade GPUs and reducing the hardware requirements for feed-forward volumetric video reconstruction.