TACC: TARGET-AWARE CARRIER COMPRESSION FOR DENSE-CONTEXT NOVEL VIEW SYNTHESIS

Anonymous authors
Paper under double-blind review

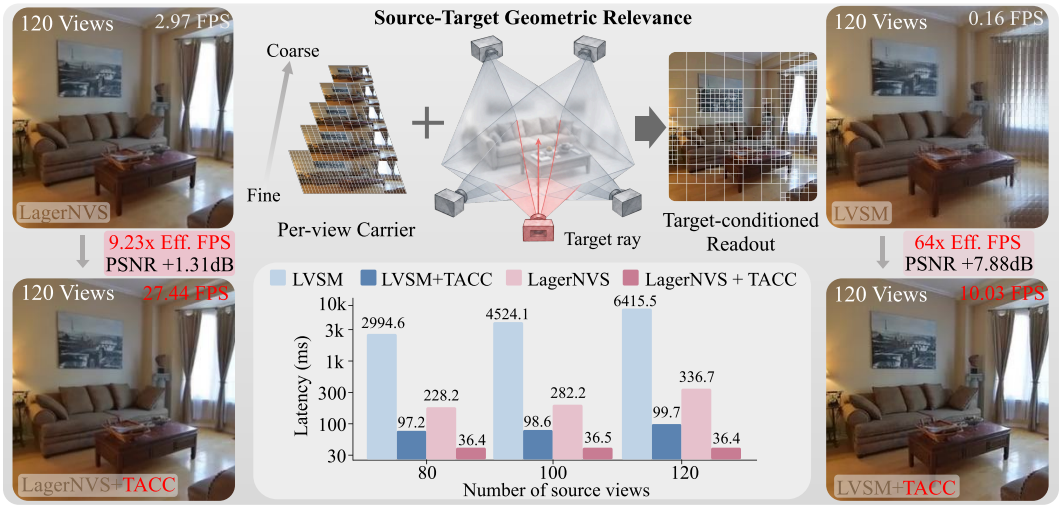


Figure 1: TACC organizes source representations into reusable per-view Carrier hierarchies for target-conditioned readout. On RE10K with 120 source views, TACC achieves $9.23\times$ and $64\times$ higher Effective FPS than Full Context on LagerNVS and LVSM, respectively.

ABSTRACT

Recent feed-forward approaches mark a paradigm shift in novel view synthesis by eliminating the reliance on per-scene optimization, directly inferring 3D representations from input views. However, their reliance on global scene-token context incurs substantial memory and latency overhead, constraining their ability to handle increasingly dense source-view contexts. Compressing source representations offers an intuitive way to limit this token budget. Prior general-purpose compression methods are not tailored for novel view synthesis, as their fixed, target-agnostic nature fails to adapt evidence allocation to individual target views. To overcome this, we introduce TACC, a target-aware compression module tailored for feed-forward latent NVS. Specifically, TACC organizes each source view into a progressive carrier hierarchy. Driven by source-target geometric relevance, it adaptively retrieves tokens to dynamically allocate exact computational quotas. This selective retrieval directly yields a compact, target-conditioned representation. Extensive experiments on the RE10K demonstrate that under dense input settings, TACC accelerates inference by $9.23\times$ and $64\times$ over the LagerNVS and LVSM. Project page at TACC.

1 INTRODUCTION

Novel view synthesis (NVS) is a fundamental task in computer vision and graphics that has long been dominated by handcrafted 3D representations and slow, per-scene optimization (Mildenhall et al., 2020; Kerbl et al., 2023). Recently, feed-forward latent methods (Jiang et al., 2025; Jin et al., 2025) have driven a paradigm shift by completely bypassing explicitly handcrafted 3D reconstructions. These purely transformer-based designs leverage scaled geometry transformers to output new

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.

views from tokens directly. By minimizing 3D inductive biases, they transform NVS into a fully data-driven problem. However, this reliance on a global scene-token context creates a fundamental bottleneck in dense source-view settings. Because the rendering pathway must process all source tokens, adapting to dense inputs inevitably inflates the token budget, making the synthesis process computationally prohibitive and causing severe memory and latency overheads (Jin et al., 2025; Jia et al., 2026).

Compressing the source token space offers the most direct strategy to mitigate this computational bottleneck. However, dedicated compression schemes for latent feed-forward NVS architectures remain largely unexplored. Adapting context reduction strategies from feed-forward 3D reconstruction or Gaussian splatting frameworks is fundamentally inadequate (Shen et al., 2026; Chen et al., 2026; Wang et al., 2025b). While these approaches successfully achieve global token reduction, they fail to address how a limited token budget should be allocated across source views. Since varying target cameras rely on different source-view overlaps and visibilities, uniform compression introduces an unavoidable tension between maintaining broad multi-view coverage and preserving high-frequency visual evidence from specific, target-relevant views. Fully unlocking the potential of feed-forward latent NVS requires a compression paradigm that is inherently target-aware.

We fulfill this requirement by introducing **TACC** (**T**arget-**A**ware **C**arrier **C**ompression), a novel compression module explicitly designed for feed-forward latent NVS. Specifically, it pre-organizes each source-view representation into a progressive, reusable carrier hierarchy. When queried with a target camera, TACC evaluates source-target geometric relevance to dynamically allocate an exact computational quota across the source views. Guided by these, it retrieves the corresponding token subsets from the hierarchy to directly construct a highly compact, target-conditioned representation. Crucially, this selective retrieval process updates only the multi-granularity Carrier module, leaving the pretrained backbone frozen. Extensive evaluations across three prominent latent NVS backbones (Szymanowicz et al., 2026; Jin et al., 2025; Wang et al., 2025c) on the DL3DV (Ling et al., 2024) and RealEstate10K (Zhou et al., 2018) datasets demonstrate that TACC consistently delivers superior reconstruction quality compared to matched-budget compression baselines. As shown in Fig. 1, TACC accelerates inference by up to $9.23\times$ and $64\times$ over vanilla LagerNVS and LVSM on RealEstate10K demonstrating its exceptional scalability and real-time potential. Our core contributions are summarized as follows:

- We introduce TACC, a target-aware compression module for dense-context feed-forward latent NVS that dynamically controls the decoder-visible token budget while keeping pretrained rendering pathways completely unchanged.
- We propose a reusable, progressive per-view carrier hierarchy. This structure partitions the source token grid into reading frontiers at every valid positive quota, enabling exact, target-conditioned allocation and readout without redundant re-encoding.
- We demonstrate that TACC consistently improves reconstruction quality over matched-budget baselines across three NVS backbones and two datasets, maintaining highly stable target-time throughput even as the source context grows significantly.

2 Related Work

2.1 Generalizable Novel View Synthesis

Generalizable novel view synthesis has evolved from learned image-based and radiance-field rendering (Wang et al., 2021; Chen et al., 2021; Johari et al., 2022) to recent feed-forward models that infer scene representations directly from source observations (Charatan et al., 2024; Chen et al., 2024; Zhang et al., 2024; Xu et al., 2024). Transformer-based methods further encode multi-view observations into learned latent representations or process source and target tokens within learned rendering architectures (Sajjadi et al., 2022; Kulhánek et al., 2022; Jin et al., 2025; Jiang et al., 2025; Szymanowicz et al., 2026). As source-view contexts grow, however, the computation required to process these representations also increases. Efficient-LVSM reduces this cost by decoupling source and target processing (Jia et al., 2026), whereas TACC compresses the source representation while retaining the pretrained rendering pathway.

2.2 Visual Token Reduction and Structure Preservation

As visual token sequences grow longer, their redundancy becomes increasingly pronounced, motivating pruning, sampling, merging, and learned token condensation across vision and multimodal Transformers (Rao et al., 2021; Liang et al., 2022; Ryoo et al., 2021; Yin et al., 2022; Fayyaz et al., 2022; Bolya et al., 2023; Alvar et al., 2025). These approaches reduce token computation through selection, pruning, merging, or condensation while preserving task-relevant visual information. Feed-forward visual geometry Transformers such as VGGT jointly process multiple views to infer camera and dense 3D attributes (Wang et al., 2025a). Recent work therefore extends token reduction to this setting, where preserving spatial and multi-view structure becomes increasingly important (Shen et al., 2026; Zheng et al., 2026). Co-Me, for example, uses confidence-guided token merging to accelerate visual geometry Transformers while preserving spatial coverage (Chen et al., 2026). These methods primarily reduce token computation within visual geometry backbones; TACC instead controls the source representation exposed to the target-conditioned rendering pathway under a fixed decoder-visible budget.

2.3 Compact Multi-View Scene Representations for Neural Rendering

More directly related to our setting, ZPressor compresses support-view information into anchor-view features to form a compact latent state for feed-forward 3D Gaussian Splatting (Wang et al., 2025b). CLiFT constructs compressed light-field tokens and retrieves a target-dependent subset according to the available rendering budget (Wang et al., 2025c). These methods establish compact scene representations for efficient neural rendering, but they do not provide a reusable per-view hierarchy that supports exact target-conditioned budget allocation across source views. TACC instead retains a reusable progressive Carrier hierarchy for each source view and allocates exact integer quotas across views according to target relevance before reading the corresponding frontiers under a fixed decoder-visible budget.

3 Method

3.1 Overview

Given V source views, TACC operates on a backbone-specific source representation $X = \{X_i\}_{i=1}^{V}$, where $X_i = \{x_{i,p}\}_{p=1}^{P} \in \mathbb{R}^{P\times C}$ denotes the per-view token representation at the TACC insertion point, with P tokens and feature dimension C. The exact construction of X_i depends on the backbone and is detailed in Appendix B.3. Under a fixed decoder-visible token budget $B = A \cdot P$, TACC compresses X into a compact target-conditioned representation Z_t containing B tokens, where A denotes the equivalent view count, i.e., the budget measured in units of one complete P-token source representation, with $1 < A \leq V$. Let R_s and R_t denote the source-view and target-view camera-ray inputs, respectively. Figure 2 illustrates the overall TACC pipeline, which is formulated as

$$Z_t = \mathrm{TACC}(X, R_s, R_t, B), \qquad I_t = \mathcal{D}(Z_t, R_t), \tag{1}$$

where D is the backbone's unchanged target-conditioned rendering pathway.

TACC follows an *encode once, allocate on demand* design. Before the target view is known, each source representation X_i is organized into a progressive Carrier hierarchy $H_i = G_\theta(X_i)$. Since the hierarchy depends only on the fixed source representation, it is constructed once and reused across target views. Once a target view t becomes available, TACC allocates the fixed budget B across source views according to their geometric relevance to the target, reads the corresponding frontier from each hierarchy, and concatenates the selected Carriers to form Z_t. Progressive Carrier Encoding provides reusable multi-granularity source representations, while Target-Conditioned Evidence Allocation determines the per-view quotas and corresponding frontier readout for each target.

3.2 Progressive Carrier Encoding

For each per-view source representation X_i, TACC constructs a progressive Carrier hierarchy that supports readout under different per-view token budgets.

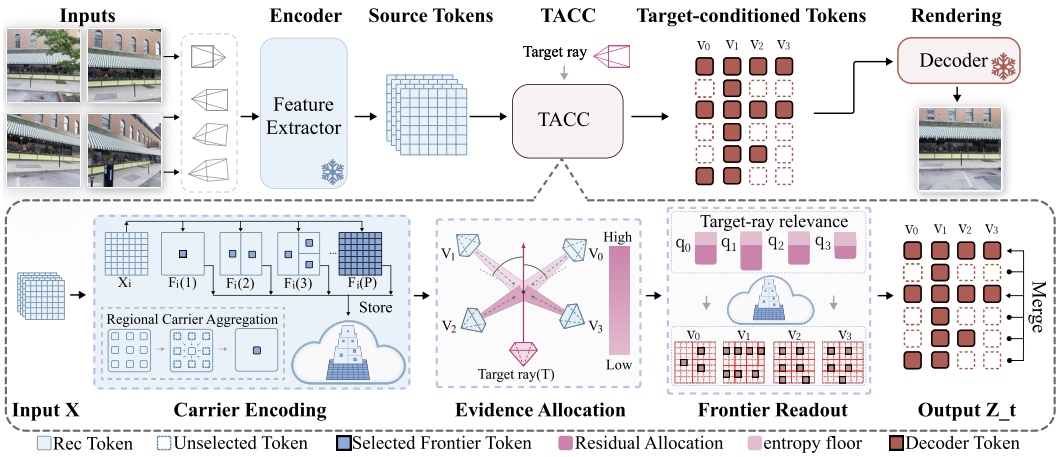


Figure 2: **Overview of TACC for feed-forward latent NVS.** Source representations are organized into reusable progressive Carrier hierarchies, from which target-aware allocation reads per-view frontiers under a fixed decoder-visible budget to form Z_t.

Regional Carrier construction. Let $X_i = \{x_{i,p}\}_{p=1}^{P}$ denote the source tokens arranged on an $H \times W$ spatial grid, where $P = HW$. For a rectangular region R, we retain the original token nearest to its geometric center as the anchor a_R; for a single-token region, the anchor is simply its unique token. We use a learned residual to encode complementary regional information.

We compute the regional mean $\mu_R = |R|^{-1} \sum_{p \in R} x_{i,p}$ and the channel-normalized variance $v_R = (|R|C)^{-1} \sum_{p \in R} \|x_{i,p} - \mu_R\|_2^2$, where C denotes the feature dimension. For region R, let h_R and w_R denote its height and width on the $H \times W$ token grid, and define the normalized four-dimensional regional attributes as

$$s_R = \left[\frac{\log |R|}{\kappa_a}, \frac{\min(\sqrt{v_R}, \kappa_v)}{\kappa_v}, \frac{h_R}{H}, \frac{w_R}{W} \right]. \tag{2}$$

Here, κ_a and κ_v are the area-normalization and variance clipping-and-normalization hyperparameters, respectively. With the anchor-to-region difference $d_R = \mathrm{LN}(\mu_R - a_R)$, the regional Carrier is defined as

$$c_R = \begin{cases} a_R, & |R| = 1, \\ a_R + \sigma(g) W_o \, \mathrm{SiLU}(\mathrm{LN}(W_d d_R + \phi_\theta(s_R))), & |R| > 1. \end{cases} \tag{3}$$

Here, W_d and W_o are learnable linear projections, ϕ_θ is a learnable MLP embedding of s_R, g is a learnable scalar gate, and σ is the sigmoid function. The same Carrier parameters are shared across all regions, hierarchy levels, and source views within a backbone. Every Carrier is computed directly from the frozen source tokens within its region rather than recursively from child Carriers.

Progressive hierarchy and frontier readout. TACC organizes each per-view source representation into a binary spatial hierarchy T_i. Starting from the complete spatial domain Ω_i, each rectangular region is bisected at the midpoint of its longer spatial dimension until each leaf contains a single token. To determine the refinement order, we measure the within-region scatter as $D(R) = \sum_{p \in R} \|x_{i,p} - \mu_R\|_2^2$ and use $\Delta(R) = D(R) - D(R_L) - D(R_R)$ as the marginal refinement gain of splitting R into R_L and R_R. Regions are refined in decreasing order of area and, among equal-area regions, decreasing $\Delta(R)$, with exact ties resolved by stable ordering.

Starting from $\mathcal{F}_i(1) = \{\Omega_i\}$, each refinement replaces one active region with its two children, increasing the frontier size by exactly one while preserving complete spatial coverage. Hence, every integer quota $q \in \{1, \ldots, P\}$ corresponds to a valid frontier. Let $H_i = (T_i, \{c_{i,R}\}, \pi_i)$ denote the resulting hierarchy and its refinement order. For $q \in \{1, \ldots, P\}$, TACC reads the Carriers on the corresponding active frontier as $Z_i(q) = \mathrm{Read}(H_i, q)$. We additionally define $Z_i(0) = \mathrm{Read}(H_i, 0) = \varnothing$, so that a zero quota simply omits source view i from the target-time readout. At the full per-view quota $q = P$, the frontier contains only single-token leaves and therefore exactly recovers the original source representation, i.e., $Z_i(P) = X_i$.

Carrier training. Only the Carrier module is optimized, while all pretrained backbone parameters remain frozen. We use teacher–student distillation (Hinton et al., 2015) with a view-subsampled but spatially dense teacher context $X^T \subseteq X$ to obtain the teacher prediction $I_t^T = \mathcal{D}(X^T, R_t)$, while the compressed representation Z_t produces the student prediction $I_t^S = \mathcal{D}(Z_t, R_t)$.

The training objective is

$$\mathcal{L} = \mathcal{L}_{\text{render}} + \lambda_{\text{dist}}\mathcal{L}_{\text{distill}} + \lambda_{\text{carrier}}\mathcal{L}_{\text{carrier}}. \tag{4}$$

Here, $\mathcal{L}_{\text{render}}$ combines ℓ_1, MSE, and image-gradient reconstruction losses; $\mathcal{L}_{\text{distill}}$ performs quality-gated RGB distillation when the teacher is more accurate under the RGB reconstruction criterion; and $\mathcal{L}_{\text{carrier}}$ regularizes active non-leaf Carriers toward their regional feature means. Detailed Carrier architecture, training objectives, teacher construction, backbone-specific protocols, and optimization settings are provided in Appendix B.1.

3.3 Target-Conditioned Evidence Allocation

Given a target view t, TACC distributes the fixed decoder-visible budget B across the source hierarchies through target-dependent per-view quotas $\{q_i^t\}_{i=1}^V$. The allocation stage is training-free and introduces no learnable parameters.

Target relevance and full-view preservation. TACC ranks source views by their geometric relevance to the target. Let d_i and d_t denote the center-pixel ray directions, and ξ_i and ξ_t the backbone-specific geometric descriptors used for distance. We define

$$r_i^t = \left\langle \frac{d_i}{\|d_i\|_2}, \frac{d_t}{\|d_t\|_2} \right\rangle - \lambda_d\, \delta(\xi_i, \xi_t). \tag{5}$$

For LagerNVS, LVSM, and CLiFT, we instantiate ξ as the central-ray Plücker moment, the central-ray origin, and the camera center from the co-normalized c2w pose, respectively. We compute $\tilde{\delta}_i^t = \|\xi_i - \xi_t\|_2$ and min–max normalize it over the source views, with $\lambda_d = 0.1$. The highest-scoring source view $i_t^\star = \arg\max_i r_i^t$ receives the full quota P. The remaining views form the support set $\mathcal{S}_t = \{1, \dots, V\} \setminus \{i_t^\star\}$, with remaining budget $B_{\text{rem}} = B - P$.

Entropy-adaptive evidence breadth. For the support views, let $\mu_{\mathcal{S}_t}$ and $\sigma_{\mathcal{S}_t}$ denote the mean and population standard deviation of their relevance scores. The standardized scores are

$$z_i^t = \begin{cases} 0, & \sigma_{\mathcal{S}_t} \le \epsilon_s, \\ \dfrac{r_i^t - \mu_{\mathcal{S}_t}}{\sigma_{\mathcal{S}_t}}, & \text{otherwise.} \end{cases} \tag{6}$$

Here, ϵ_s is a small numerical threshold. We then obtain $p_i^t = \mathrm{Softmax}(z^t)_i$ and use the normalized entropy of this distribution to determine a shared coverage floor:

$$H^t = -\frac{\sum_{i \in \mathcal{S}_t} p_i^t \log p_i^t}{\log\big(\max(|\mathcal{S}_t|, 2)\big)}, \qquad q_{\text{floor}}^t = \left\lfloor \frac{H^t B_{\text{rem}}}{|\mathcal{S}_t|} \right\rfloor. \tag{7}$$

Higher entropy produces a larger shared floor and broader cross-view coverage, whereas lower entropy leaves more of the budget for selective allocation.

Relevance-weighted residual allocation. After assigning the shared floor, the residual budget is $B_{\text{res}}^t = B_{\text{rem}} - |\mathcal{S}_t| q_{\text{floor}}^t$. We apply a softmax with temperature $\tau = 0.8$ over the support set to obtain $\hat{p}_i^t$, yielding soft residual quotas $a_i^t = B_{\text{res}}^t \hat{p}_i^t$. We then use capacity-aware largest-remainder rounding to convert the soft residual quotas into integer allocations b_i^t under the remaining per-view capacity $P - q_{\text{floor}}^t$, while preserving the residual budget exactly.

The final quotas are $q_{i_t^\star}^t = P$ and $q_i^t = q_{\text{floor}}^t + b_i^t$ for $i \in \mathcal{S}_t$. TACC then reads $Z_i^t = \mathrm{Read}(H_i, q_i^t)$ from each source hierarchy and concatenates the resulting representations to form Z_t, which is passed to the unchanged target-conditioned rendering pathway of the corresponding backbone.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETTINGS

Datasets. We evaluate TACC on DL3DV (Ling et al., 2024) and RealEstate10K (RE10K) (Zhou et al., 2018). On DL3DV, the Carrier modules are trained on 10,139 training scenes and evaluated on the official 140-scene benchmark. On RE10K, the LVSM Carrier is trained using scenes from the RE10K training split. For the large-view RE10K evaluation, we select scenes with more than 200 available views from the 6,474 scenes in our preprocessed test split, yielding 1,633 evaluation scenes that are disjoint from those used for training and validation.

Backbones, baselines, and metrics. We integrate TACC into three latent NVS backbones: LagerNVS (Szymanowicz et al., 2026), LVSM Decoder-Only (Jin et al., 2025), and CLiFT (Wang et al., 2025c). We compare against ToMe (Bolya et al., 2023), DivPrune (Alvar et al., 2025), and ZPressor (Wang et al., 2025b). Within each experimental setting, all compression methods operate under the same decoder-visible token budget and use identical scene IDs and source- and target-view indices. Full Context denotes inference with all available source tokens using the corresponding frozen pretrained backbone, without dense-context adaptation or fine-tuning. For CLiFT, Native denotes its original compression pathway retrained under matched experimental settings, with the pretrained encoder and renderer kept frozen. We evaluate reconstruction quality using PSNR, SSIM (Wang et al., 2004), and LPIPS (Zhang et al., 2018), and measure target-time efficiency using Effective FPS. For a fixed set of source views, Effective FPS includes all target-dependent computation from the arrival of the target camera to the final RGB prediction, but excludes one-time source encoding and, for TACC, one-time Carrier construction. Amortized end-to-end (E2E) FPS includes these one-time costs; see Appendix C.5.

Implementation details. We use the official implementations and pretrained checkpoints of LagerNVS, LVSM Decoder-Only, and CLiFT, and train only the Carrier module while keeping the pretrained backbone parameters frozen. For CLiFT, TACC replaces its native compression pathway rather than introducing an additional compression stage. LagerNVS and LVSM Decoder-Only are evaluated at 256×256, while CLiFT follows its native 256×448 resolution. For each backbone, a single Carrier checkpoint is shared across all evaluated source-view counts and target views under its corresponding decoder-visible budget. Backbone-specific training protocols, optimization settings, and checkpoint-selection criteria are provided in Appendix B.1. All experiments are conducted on NVIDIA A800 GPUs.

4.2 MATCHED-BUDGET COMPARISONS AND EFFICIENCY ANALYSIS

Matched-budget comparisons. We compare TACC with representation-reduction baselines on DL3DV using LagerNVS with source-view counts ranging from 80 to 120. Compressed methods use a fixed $A = 10$ decoder-visible budget, while Full Context uses all source tokens; the target-view batch size is fixed to 4 for all methods.

As shown in Tab. 1, TACC achieves the strongest reconstruction quality among compressed methods while maintaining nearly constant target-time throughput as the source set grows. Although ToMe and DivPrune are slightly faster, they incur substantially larger quality degradation. Notably, TACC also yields higher reconstruction quality than the frozen Full Context checkpoint in all three evaluated dense-view settings. For the frozen pretrained LagerNVS checkpoint evaluated here, Full Context quality decreases as the test-time source context grows, whereas TACC operates under a fixed decoder-visible budget and maintains higher reconstruction quality. Fig. 3 provides consistent qualitative evidence.

Cross-dataset generalization. We directly evaluate the DL3DV-trained LagerNVS and TACC checkpoints on RE10K without fine-tuning, using 80 to 120 source views, a fixed $A = 10$ decoder-visible budget, and 12 target views. Across these settings, TACC consistently outperforms matched-budget compression baselines in reconstruction quality while maintaining nearly constant Effective FPS, whereas frozen Full Context inference becomes progressively slower.

Table 1: **Quantitative comparisons on DL3DV.** Compression methods use a fixed $A = 10$ decoder-visible budget and identical source views within each setting; Full Context uses all LagerNVS source tokens as the uncompressed reference.

Method	80 Source Views				100 Source Views				120 Source Views			
	PSNR ↑	SSIM ↑	LPIPS ↓	Eff. FPS ↑	PSNR ↑	SSIM ↑	LPIPS ↓	Eff. FPS ↑	PSNR ↑	SSIM ↑	LPIPS ↓	Eff. FPS ↑
LagerNVS-based												
Full Context	25.6766	0.8064	0.0963	4.40	25.2932	0.7953	0.1012	3.57	24.9604	0.7851	0.1056	2.99
ToMe	20.0344	0.5442	0.3548	31.43	19.7096	0.5304	0.3737	31.00	19.2548	0.5122	0.3991	31.34
DivPrune	21.7624	0.6560	0.2053	**31.54**	21.5944	0.6479	0.2106	31.07	21.4627	0.6420	0.2147	31.08
ZPressor	24.1568	0.7720	0.1381	31.26	24.0773	0.7684	0.1396	**31.61**	24.5947	0.7872	0.1338	**31.52**
Ours	**27.3415**	**0.8649**	**0.0791**	27.95	**26.8787**	**0.8523**	**0.0833**	28.12	**26.4732**	**0.8416**	**0.0869**	27.96

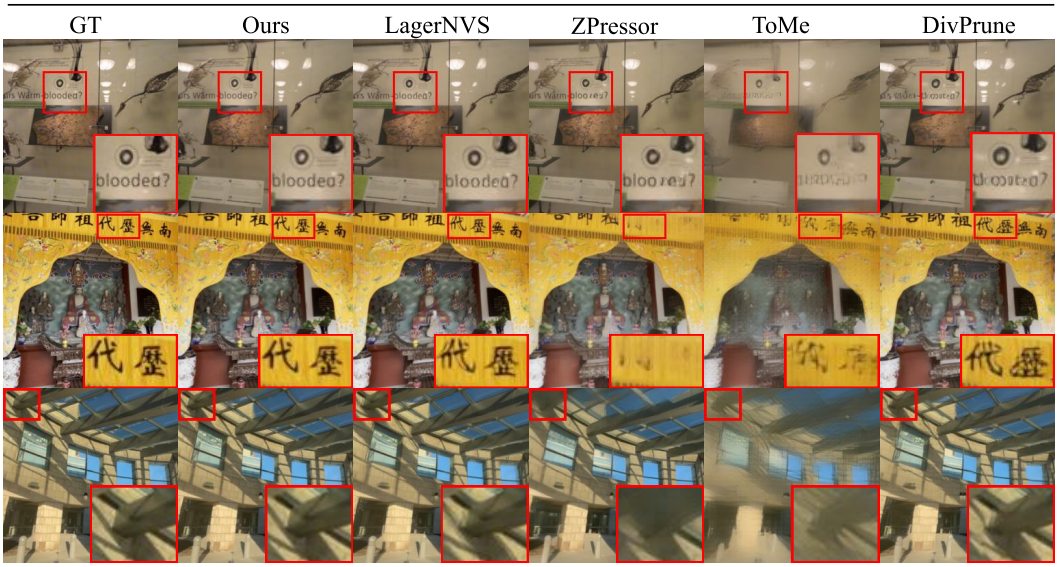
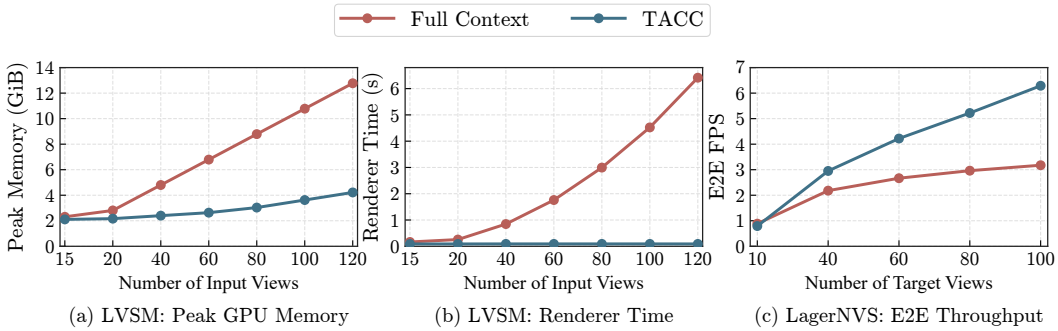


Figure 3: **Qualitative comparison on DL3DV with 100 source views.** Compressed methods use a fixed $A = 10$ decoder-visible budget, while LagerNVS uses Full Context.



(a) LVSM: Peak GPU Memory (b) LVSM: Renderer Time (c) LagerNVS: E2E Throughput

Figure 4: **Efficiency analysis.** We report peak GPU memory and renderer latency for LVSM on RE10K with source-view counts ranging from 15 to 120, and amortized E2E FPS for LagerNVS on DL3DV with 80 source views across increasing target counts.

Cross-backbone evaluation and efficiency. We further evaluate TACC with LVSM Decoder-Only under the same RE10K evaluation protocol. By training only the Carrier module while keeping the pretrained backbone frozen, TACC enables the use of dense source views without long-context retraining or fine-tuning of the backbone. As shown in Tab. 2, with source-view counts ranging from 80 to 120, TACC achieves the highest reconstruction quality among compressed methods and improves both reconstruction quality and target-time throughput over Full Context, while maintaining Effective FPS near 10. Fig. 5 further shows that TACC better preserves fine structures under the matched budget. For LVSM Decoder-Only, Figs. 4(a) and 4(b) show that Full Context incurs increasing renderer latency and peak allocated GPU memory as the source context grows, whereas TACC maintains nearly constant renderer latency and slower memory growth. At 120 source views, TACC reduces peak allocated memory from 12.77 to 4.22 GiB, a 66.96% reduction. The LagerNVS

E2E results in Fig. 4(c) further show that TACC is slightly slower with 10 targets, but surpasses Full Context at 40 targets and achieves a $1.98\times$ speedup at 100 targets. Additional LVSM results with larger source-view counts and runtime details are provided in Appendices C.1 and C.5.

Table 2: **Quantitative comparisons on RE10K.** Results are reported for LagerNVS and LVSM Decoder-Only, with compressed methods using a fixed $A = 10$ decoder-visible budget and Full Context using all source tokens as the uncompressed reference.

Method	80 Source Views				100 Source Views				120 Source Views			
	PSNR ↑	SSIM ↑	LPIPS ↓	Eff. FPS ↑	PSNR ↑	SSIM ↑	LPIPS ↓	Eff. FPS ↑	PSNR ↑	SSIM ↑	LPIPS ↓	Eff. FPS ↑
LagerNVS-based												
Full Context	28.5991	0.9066	0.0681	4.38	28.0454	0.8976	0.0718	3.54	27.5702	0.8890	0.0754	2.97
ToMe	22.1035	0.7030	0.2889	**31.51**	21.8357	0.6915	0.3070	31.05	21.4481	0.6780	0.3276	**31.50**
DivPrune	25.1277	0.8298	0.1215	31.46	24.9600	0.8252	0.1240	**31.42**	24.8335	0.8217	0.1259	31.12
ZPressor	27.9540	0.8999	0.0854	31.28	27.7685	0.8961	0.0868	31.41	27.5760	0.8921	0.0882	31.20
Ours	**30.2044**	**0.9285**	**0.0645**	27.47	**29.4752**	**0.9182**	**0.0681**	27.42	**28.8774**	**0.9090**	**0.0715**	27.44
LVSM Decoder-Only-based												
Full Context	22.7867	0.7489	0.2134	0.33	22.4844	0.7364	0.2267	0.22	22.2624	0.7269	0.2375	0.16
ToMe	19.4181	0.5842	0.3874	**11.25**	19.1861	0.5698	0.4015	11.12	18.8588	0.5585	0.4182	**11.34**
DivPrune	20.3286	0.6110	0.3084	11.21	20.0773	0.5990	0.3233	**11.29**	19.9148	0.5911	0.3338	11.10
ZPressor	27.8870	0.8883	0.1047	11.17	27.8993	0.8885	0.1045	11.28	27.9235	0.8890	0.1041	11.25
Ours	**30.2851**	**0.9260**	**0.0739**	10.28	**30.1978**	**0.9250**	**0.0744**	10.15	**30.1441**	**0.9243**	**0.0750**	10.03

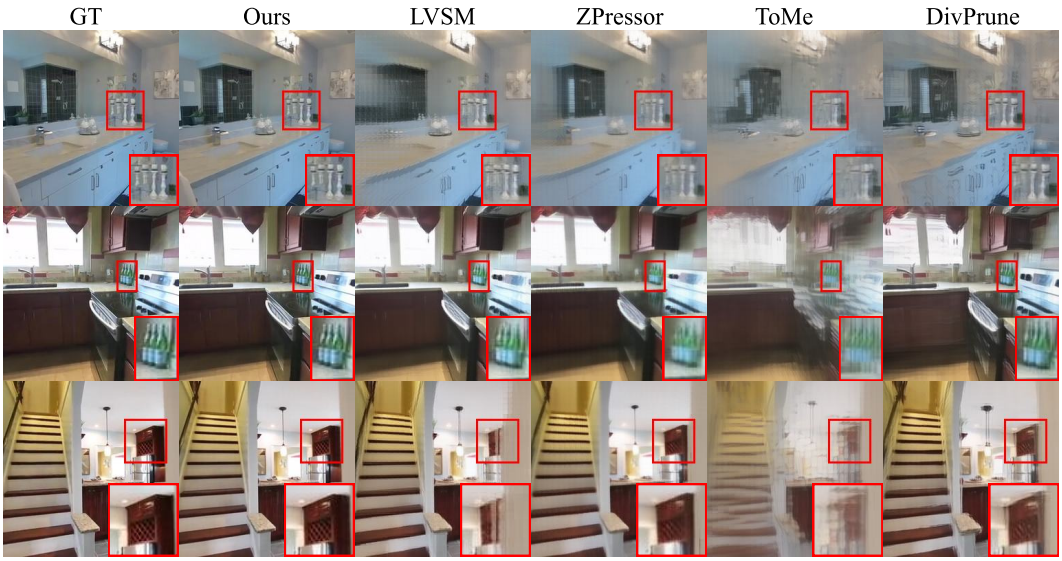


Figure 5: **Qualitative comparison with LVSM Decoder-Only on RE10K.** Compressed methods use a fixed $A = 10$ decoder-visible budget, while LVSM Decoder-Only uses Full Context.

Table 3: **Comparison on CLiFT on DL3DV with increasing source-view counts.** Compressed variants use a fixed $A = 6$ budget, while Full Context uses the complete representation.

Method	24 Source Views				36 Source Views				48 Source Views			
	PSNR ↑	SSIM ↑	LPIPS ↓	Eff. FPS ↑	PSNR ↑	SSIM ↑	LPIPS ↓	Eff. FPS ↑	PSNR ↑	SSIM ↑	LPIPS ↓	Eff. FPS ↑
CLiFT-based												
Full Context	20.1825	0.5962	0.2912	15.43	20.1843	0.5940	0.2992	10.64	20.1703	0.5930	0.3014	8.08
Native	21.8549	0.6574	0.2569	47.29	21.7869	0.6513	0.2630	**47.46**	21.7493	0.6498	0.2647	47.39
ToMe	19.4351	0.5367	0.3850	47.27	19.3258	0.5292	0.3963	47.19	18.7800	0.5031	0.4332	**47.40**
DivPrune	19.4529	0.5515	0.3264	47.21	19.3823	0.5437	0.3371	47.13	19.3468	0.5411	0.3407	47.09
ZPressor	20.9791	0.6344	0.2814	**47.43**	21.0250	0.6348	0.2825	47.43	20.9394	0.6307	0.2855	47.27
Ours	**23.3425**	**0.7397**	**0.2180**	46.37	**23.5100**	**0.7441**	**0.2165**	45.20	**23.5842**	**0.7470**	**0.2149**	43.18

Complementary evaluation on CLiFT. We further evaluate TACC with CLiFT (Wang et al., 2025c) on DL3DV, where the native pathway already constructs a compact light-field token representation and performs target-dependent token retrieval. TACC replaces this compression-and-retrieval pathway with progressive per-view Carrier encoding and target-conditioned frontier read-

out, while retaining the pretrained encoder and renderer. As shown in Tab. 3, TACC achieves the best reconstruction quality among all compressed variants across 24, 36, and 48 source views, outperforming the retrained Native pathway by 1.49–1.83 dB in PSNR with consistent improvements in SSIM and LPIPS. The quality margin further increases as the source-to-budget ratio grows from $4\times$ to $8\times$, with only a modest reduction in target-time throughput. Appendix C.3 provides additional qualitative comparisons under the same matched setting in Fig. 8 and separately reports the official pretrained Native checkpoint for reference in Tab. 8.

4.3 Ablation Study

We evaluate TACC using LagerNVS on DL3DV with 80 source views and a fixed $A = 10$ decoder-visible budget. Tab. 4 shows that all controlled variants reduce reconstruction quality relative to TACC, while the qualitative comparisons in Fig. 6 exhibit consistent trends. Together, these results support the contributions of both learned regional Carriers and target-conditioned evidence allocation, with further controls provided in Appendix C.4.

The *w/o Carrier* variant retains TACC's target-conditioned per-view quotas but replaces Carrier readout with approximately uniform spatial subsampling of the original tokens, while *Uniform Alloc.* instead uses approximately uniform per-view quotas, with its Carrier retrained under the same allocation rule. *Top-A Full Views* retains the A highest-ranked complete source views according to TACC's relevance score.

Table 4: **Component ablations.**

Variant	PSNR ↑	SSIM ↑	LPIPS ↓
Top-A Full Views	21.6315	0.6509	0.1582
Anchor-only Carrier	22.2410	0.6936	0.1316
w/o Carrier	25.1493	0.8047	0.0984
Uniform Alloc.	23.0085	0.7154	0.1945
TACC	**27.3415**	**0.8649**	**0.0791**



Figure 6: **Qualitative analysis of Carrier and evidence allocation.** We compare TACC with w/o Carrier, Uniform Alloc., Anchor-only Carrier, and Top-A Full Views on DL3DV. Red boxes indicate the regions enlarged in the corresponding insets.

Anchor-only Carrier preserves the hierarchy and allocation but replaces learned regional Carriers with their anchors. Top-A Full Views indicates that selecting target-relevant complete views alone is insufficient, while Anchor-only Carrier shows that the hierarchy and allocation without learned regional Carriers cannot recover TACC's performance. The degradation under *w/o Carrier* further supports the contribution of learned regional Carriers, whereas the larger drop under *Uniform Alloc.* highlights the role of target-conditioned evidence allocation.

5 Conclusion

We addressed the challenge of efficiently processing dense source views in feed-forward latent NVS, where growing scene-token context incurs substantial target-time memory and latency overhead. We introduced TACC, a target-aware compression module that organizes each source view into a reusable progressive Carrier hierarchy. Under a fixed decoder-visible token budget, TACC assigns exact per-view quotas according to source-target geometric relevance and reads the corresponding Carrier frontiers to construct a compact target-conditioned representation while keeping the pretrained rendering pathway unchanged. Across three latent NVS backbones and two datasets, TACC consistently improves reconstruction quality over matched-budget compression baselines while maintaining stable target-time throughput under dense source views, demonstrating the effectiveness of target-aware representation-granularity allocation under a fixed rendering budget.

Reproducibility Statement

We provide the main algorithmic details of TACC in Secs. 3.2 and 3.3, including progressive Carrier construction, frontier readout, and target-conditioned evidence allocation. The datasets, pretrained backbones, decoder-visible budgets, image resolutions, evaluation metrics, and hardware settings are described in Sec. 4.1.

The appendix provides additional details required for reproduction, including the training objectives, backbone-specific training protocols, validation and checkpoint-selection criteria, optimization settings, target-time inference procedure, backbone integration, deterministic source/target view construction, and exact-budget baseline implementations. It further reports runtime analysis and additional robustness and source-view-count evaluations.

All experiments use publicly available datasets and official pretrained backbones where applicable. We plan to release the implementation and trained Carrier checkpoints to facilitate reproduction of the reported results.

AI Use Statement

We used GPT 5.6 to assist with manuscript writing and language refinement, including translation, improving clarity and readability, and organizing the presentation of the paper. It was also used to provide feedback on the presentation and interpretation of experimental results. All technical content, experimental results, claims, and AI-assisted text were reviewed and verified by the authors, who take full responsibility for the final content of the paper.

References

Saeed Ranjbar Alvar, Gursimran Singh, Mohammad Akbari, and Yong Zhang. DivPrune: Diversity-based visual token pruning for large multimodal models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9392–9401, June 2025.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your ViT but faster. In *The Eleventh International Conference on Learning Representations*, 2023.

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 (CVPR)*, pp. 19457–19467, June 2024.

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

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 *Computer Vision – ECCV 2024*, pp. 370–386, 2024.

Yutian Chen, Yuheng Qiu, Ruogu Li, Jay Patrikar, and Sebastian Scherer. Co-Me: Confidence guided token merging for visual geometric transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14590–14599, June 2026.

Mohsen Fayyaz, Soroush Abbasi Koohpayegani, Farnoush Rezaei Jafari, Sunando Sengupta, Hamid Reza Vaezi Joze, Eric Sommerlade, Hamed Pirsiavash, and Jürgen Gall. Adaptive token sampling for efficient vision transformers. In *Computer Vision – ECCV 2022*, pp. 396–414, 2022.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Xiaosong Jia, Yihang Sun, Junqi You, Songbur Wong, Zichen Zou, Junchi Yan, Zuxuan Wu, and Yu-Gang Jiang. Efficient-LVSM: Faster, cheaper, and better large view synthesis model via decoupled co-refinement attention. In *The Fourteenth International Conference on Learning Representations*, 2026.

Hanwen Jiang, Hao Tan, Peng Wang, Haian Jin, Yue Zhao, Sai Bi, Kai Zhang, Fujun Luan, Kalyan Sunkavalli, Qixing Huang, and Georgios Pavlakos. RayZer: A self-supervised large view synthesis model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4918–4929, October 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. In *The Thirteenth International Conference on Learning Representations*, 2025.

Mohammad Mahdi Johari, Yann Lepoittevin, and François Fleuret. GeoNeRF: Generalizing NeRF with geometry priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18365–18375, June 2022.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4):139:1–139:14, July 2023.

Jonáš Kulhánek, Erik Derner, Torsten Sattler, and Robert Babuška. ViewFormer: NeRF-free neural rendering from few images using transformers. In *Computer Vision – ECCV 2022*, pp. 198–216, 2022.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set Transformer: A framework for attention-based permutation-invariant neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3744–3753. PMLR, 2019.

Youwei Liang, Chongjian Ge, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. Not all patches are what you need: Expediting vision transformers via token reorganizations. In *International Conference on Learning Representations*, 2022.

Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, Xuanmao Li, Xingpeng Sun, Rohan Ashok, Aniruddha Mukherjee, Hao Kang, Xiangrui Kong, Gang Hua, Tianyi Zhang, Bedrich Benes, and Aniket Bera. DL3DV-10K: A large-scale scene dataset for deep learning-based 3D vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22160–22169, June 2024.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *Computer Vision – ECCV 2020*, pp. 405–421. Springer, 2020.

Jinsoo Park, Donggyu Choi, Ahyun Seo, Minsu Cho, and Jeany Son. Diversity-aware view partitioning for scalable VGGT. In *Computer Vision – ECCV 2026*, pp. 547–564, 2026.

Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. DynamicViT: Efficient vision transformers with dynamic token sparsification. In *Advances in Neural Information Processing Systems*, volume 34, pp. 13937–13949, 2021.

Michael S. Ryoo, A. J. Piergiovanni, Anurag Arnab, Mostafa Dehghani, and Anelia Angelova. TokenLearner: Adaptive space-time tokenization for videos. In *Advances in Neural Information Processing Systems*, volume 34, pp. 12786–12797, 2021.

Mehdi S. M. Sajjadi, Henning Meyer, Etienne Pot, Urs Bergmann, Klaus Greff, Noha Radwan, Suhani Vora, Mario Lučić, Daniel Duckworth, Alexey Dosovitskiy, Jakob Uszkoreit, Thomas Funkhouser, and Andrea Tagliasacchi. 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 (CVPR)*, pp. 6229–6238, June 2022.

You Shen, Zhipeng Zhang, Yansong Qu, Xiawu Zheng, Jiayi Ji, Shengchuan Zhang, and Liujuan Cao. FastVGGT: Fast visual geometry transformer. In *The Fourteenth International Conference on Learning Representations*, 2026.

Stanislaw Szymanowicz, Minghao Chen, Jianyuan Wang, Christian Rupprecht, and Andrea Vedaldi. LagerNVS: Latent geometry for fully neural real-time novel view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15443–15453, June 2026.

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. VGGT: Visual geometry grounded transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5294–5306, June 2025a.

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

Weijie Wang, Donny Y. Chen, Zeyu Zhang, Duochao Shi, Akide Liu, and Bohan Zhuang. ZPressor: Bottleneck-aware compression for scalable feed-forward 3DGS. In *Advances in Neural Information Processing Systems*, volume 38, pp. 125793–125822, 2025b.

Zhengqing Wang, Yuefan Wu, Jiacheng Chen, Fuyang Zhang, and Yasutaka Furukawa. CLiFT: Compressive light-field tokens for compute efficient and adaptive neural rendering. In *Advances in Neural Information Processing Systems*, volume 38, pp. 8759–8787, 2025c.

Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4): 600–612, April 2004.

Yinghao Xu, Zifan Shi, Yifan Wang, Hansheng Chen, Ceyuan Yang, Sida Peng, Yujun Shen, and Gordon Wetzstein. GRM: Large gaussian reconstruction model for efficient 3D reconstruction and generation. In *Computer Vision – ECCV 2024*, pp. 1–20, 2024.

Hongxu Yin, Arash Vahdat, Jose M. Alvarez, Arun Mallya, Jan Kautz, and Pavlo Molchanov. A-ViT: Adaptive tokens for efficient vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10809–10818, June 2022.

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 *Computer Vision – ECCV 2024*, pp. 1–19, 2024.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 586–595, June 2018.

Shuhong Zheng, Michael Oechsle, Erik Sandström, Marie-Julie Rakotosaona, Federico Tombari, and Igor Gilitschenski. Good token hunting: A hitchhiker's guide to token selection for visual geometry transformers. *arXiv preprint arXiv:2605.23892*, 2026.

Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *ACM Transactions on Graphics*, 37(4):65:1–65:12, August 2018.

A Limitations and Future Work

TACC introduces one-time computation and storage costs to construct and cache the per-view Carrier hierarchies. Although these costs are amortized across targets, they are excluded from Effective FPS, and the end-to-end benefit depends on the number of targets rendered from the same source context. The LagerNVS E2E results in Fig. 4(c) and the complementary LVSM evaluation in Appendix C.5 illustrate this trade-off. In the evaluated LagerNVS setting, TACC is slightly slower than

Full Context with 10 targets but achieves higher E2E throughput as more targets are rendered. Since the one-time overhead is less effectively amortized for shorter target sequences, reducing hierarchy construction cost could improve E2E performance in such workloads, while reducing the cache footprint would lower storage requirements.

TACC currently uses target relevance to determine the token budget for each source view, while its intra-view frontier follows a precomputed refinement order. Thus, the selected regional granularity does not explicitly adapt to the content most relevant to the current target. A promising direction is to improve target-conditioned regional refinement within each fixed per-view quota, allowing the same number of tokens to represent target-relevant information more effectively. Such an extension could build on the reusable Carrier hierarchies while preserving the exact decoder-visible budget and limiting additional target-time overhead.

B Additional Implementation Details

B.1 Carrier Architecture and Training Details

Carrier architecture. All three backbones use the same Carrier architecture with token dimension $C = 768$ and hidden dimension $D_h = 192$. For the regional attributes in Eq. (2), we set $\kappa_a = 8$ and $\kappa_v = 10$. The four-dimensional attribute embedding is $\phi_\theta : \mathrm{Linear}(4, 32) \to \mathrm{SiLU} \to \mathrm{Linear}(32, 192)$.

The experimental checkpoints retain a duplicate of the first area attribute: for $s_R = [s_1, s_2, s_3, s_4]$, their input is $\tilde{s}_R = [s_1, s_1, s_2, s_3, s_4]$. This redundant input is exactly equivalent to the four-dimensional formulation. Specifically, the first-layer weights $\widetilde{W}_\phi^{(1)} = [w_1, w_2, w_3, w_4, w_5]$ can be replaced by $W_\phi^{(1)} = [w_1 + w_2, w_3, w_4, w_5]$ without changing the bias or network output.

The projections $W_d : \mathbb{R}^{768} \to \mathbb{R}^{192}$ and $W_o : \mathbb{R}^{192} \to \mathbb{R}^{768}$ map the anchor-to-region difference into and out of the hidden space. The combined hidden representation is normalized by LayerNorm and followed by SiLU. The scalar residual gate g is initialized to -2.0, and the weight and bias of W_o are zero-initialized. Carrier parameters are shared across source views, regions, and hierarchy levels within each backbone; different backbones use independently trained Carrier checkpoints. The spatial token grids are 37×37 for LagerNVS, 32×32 for LVSM Decoder-Only, and 32×56 for CLiFT.

Training objectives. We expand the three training terms in Eq. (4). The rendering objective combines RGB reconstruction and first-order image-gradient consistency:

$$\begin{aligned}\mathcal{L}_{\mathrm{render}} = \lambda_1 \ \mathrm{mean}\big|I^S - I^{GT}\big| + \lambda_{\mathrm{mse}} \ \mathrm{mean}\big[(I^S - I^{GT})^2\big] \\ + \lambda_{\mathrm{grad}}\Big(\mathrm{mean}\big|\nabla_x I^S - \nabla_x I^{GT}\big| + \mathrm{mean}\big|\nabla_y I^S - \nabla_y I^{GT}\big| \Big).\end{aligned} \tag{8}$$

Here, ∇_x and ∇_y denote first-order differences between neighboring pixels. All reductions average over the corresponding batch, target, channel, and spatial dimensions.

RGB distillation is applied independently to each scene–target pair when the teacher has lower reconstruction error than the student. Let $\mathcal{B}$ and $\mathcal{T}$ denote the batch and target index sets, and let N_{rgb} be the number of scalar values in one RGB image. We define $e^S_{b,t} = \|I^S_{b,t} - I^{GT}_{b,t}\|_1 / N_{\mathrm{rgb}}$ and $e^T_{b,t} = \|I^T_{b,t} - I^{GT}_{b,t}\|_1 / N_{\mathrm{rgb}}$. The quality-gated distillation loss is

$$g_{b,t} = \mathbf{1}\big[e^T_{b,t} < e^S_{b,t}\big], \qquad \mathcal{L}_{\mathrm{distill}} = \frac{1}{|\mathcal{B}||\mathcal{T}|} \sum_{b \in \mathcal{B}} \sum_{t \in \mathcal{T}} g_{b,t} \frac{\|I^S_{b,t} - I^T_{b,t}\|_1}{N_{\mathrm{rgb}}}. \tag{9}$$

The student error used to compute $g_{b,t}$ is detached. Inactive terms remain in the full batch–target normalization.

Carrier regularization constrains active non-leaf Carriers toward their corresponding regional feature means. Let $\mathcal{A}_{t,i}$ denote the active non-leaf regions of source view i under the quota for target t, and let $\mathcal{T}_{\mathrm{reg}}$ denote the training targets used for regularization. The valid target–view pairs are

$$\mathcal{J} = \{(t, i) \mid t \in \mathcal{T}_{\mathrm{reg}},\ \mathcal{A}_{t,i} \neq \varnothing\}. \tag{10}$$

The regularization loss is

$$\mathcal{L}_{\text{carrier}} = \frac{1}{|\mathcal{J}|} \sum_{(t,i)\in\mathcal{J}} \frac{1}{|\mathcal{A}_{t,i}|C} \sum_{R\in\mathcal{A}_{t,i}} \sum_{c=1}^{C} \text{SmoothL1}\left(c_{R,c}, \mu_{R,c}\right). \tag{11}$$

SmoothL1 uses $\beta = 1$. Regions are averaged uniformly within each valid target–view pair, without area-based weighting. For LagerNVS and LVSM Decoder-Only, $\mathcal{T}_{\text{reg}}$ contains the first training target; for CLiFT, it contains both training targets.

Teacher context. For all three backbones, the teacher representation $X^T \subseteq X$ uses a fixed context $\mathcal{C}_{24}$ of 24 pose-covered source views selected by the procedure in Appendix B.4. Each selected teacher view retains its complete token grid.

Backbone-specific training and checkpoint selection. Only the Carrier module is trained; all pretrained backbone parameters remain frozen. Source and target views are disjoint, and each training sample uses two distinct target views. The deterministic view-construction procedure and the fixed $\mathcal{T}_{20}$ target pool are defined in Appendix B.4. Each backbone uses its corresponding source-view regime and decoder-visible budget.

LagerNVS. Carrier training uses 10,139 DL3DV scenes, with source-view counts $V \in \{60, 80, 100, 120\}$ and a fixed budget of $A = 10$. Each of the 12 epochs traverses all training scenes using the nested pose-coverage source contexts. For each scene, a deterministic permutation of the fixed $\mathcal{T}_{20}$ pool selects two distinct targets per epoch. The first ten epochs cover all 20 targets once; subsequent epochs deterministically revisit a subset.

Validation uses 200 held-out DL3DV scenes and four fixed targets per source-view setting. We select the Carrier checkpoint by minimizing the weighted ground-truth validation reconstruction loss over the settings with 80, 100, and 120 source views, using weights 0.2, 0.3, and 0.5, respectively. The selected checkpoint is from epoch 9.

LVSM Decoder-Only. We use 2,048 RE10K scenes for Carrier training and 128 for validation. These partitions and the 1,633-scene large-view evaluation set are mutually disjoint. The training scenes are assigned to three fixed groups by a deterministic hash of their scene identifiers, containing 661, 700, and 687 scenes, respectively. The source-view count assigned to each group is rotated among $V \in \{80, 100, 120\}$ across epochs. Let $\text{HashBucket}(s) \in \{0, 1, 2\}$ denote the group of scene s, and let $(v_0, v_1, v_2) = (80, 100, 120)$. At epoch e, the source-view count is

$$V(s, e) = v_{(e-1+\text{HashBucket}(s)) \bmod 3}. \tag{12}$$

Over 12 epochs, each scene is therefore used four times at each source-view count. Training uses the same deterministic pose-coverage and target-sampling procedure as LagerNVS, under a fixed $A = 10$ budget.

For checkpoint selection, each validation scene uses four fixed target indices $\{2, 7, 12, 17\}$ from its 20-view target pool. Evaluation across the three source-view settings yields 1,536 rendered targets per validation round. We select the Carrier checkpoint by validation PSNR; the selected checkpoint is from epoch 12.

CLiFT. Carrier training uses 2,048 DL3DV scenes with source-view counts $V \in \{24, 36, 48\}$ and a fixed budget of $A = 6$. Training runs for 24 epochs, totaling 3,072 optimizer steps. Nested pose-coverage source contexts are constructed within the first 100 available views after reserving the target pool. A deterministic scene-hash and epoch-dependent cycle assign the source-view setting, such that each scene is trained eight times at each source-view count. Two distinct targets are selected from opposite halves of the fixed $\mathcal{T}_{20}$ pool.

Validation uses 60 held-out scenes and all 20 target views for each source-view setting. We select the Carrier checkpoint using equal-weight macro PSNR across the three settings. The selected checkpoint is from epoch 23.

Optimization. We share the loss weights and optimizer hyperparameters across all three backbones: $\lambda_1 = 1$, $\lambda_{\text{mse}} = 0.25$, $\lambda_{\text{grad}} = 0.10$, $\lambda_{\text{dist}} = 0.25$, and $\lambda_{\text{carrier}} = 0.05$. We optimize the Carrier with AdamW using an initial learning rate of 1×10^{-5}, weight decay of 1×10^{-4}, and gradient clipping at 1.0. A 20-step warmup is followed by cosine decay to 1×10^{-6}.

Algorithm 1 Target-Conditioned Allocation and Frontier Readout in TACC.

Input: Carrier hierarchies $\{H_i\}_{i=1}^{V}$, source center-pixel ray directions $\{d_i\}_{i=1}^{V}$, source geometric descriptors $\{\xi_i\}_{i=1}^{V}$, target center-pixel ray direction d_t, target geometric descriptor ξ_t, budget B, and tokens per view P.
Output: Target-conditioned representation Z_t with $|Z_t| = B$.

Target relevance and full-view preservation:
for $i = 1$ to V **do**
  $\tilde{\delta}_i^t \leftarrow \|\xi_i - \xi_t\|_2$
end for
Normalize $\{\tilde{\delta}_i^t\}_{i=1}^{V}$ using Eq. (13)
Compute $\{r_i^t\}_{i=1}^{V}$ using Eq. (5)
$q^t \leftarrow \mathbf{0}$
$i_t^\star \leftarrow \arg\max_i r_i^t$
$q_{i_t^\star}^t \leftarrow P$
$\mathcal{S}_t \leftarrow \{1, \ldots, V\} \setminus \{i_t^\star\}$
$B_{\mathrm{rem}} \leftarrow B - P$

Entropy-adaptive evidence breadth:
Compute $\{z_i^t\}_{i \in \mathcal{S}_t}$ using Eq. (6)
Compute H^t and q_{floor}^t using Eq. (7)
$B_{\mathrm{res}}^t \leftarrow B_{\mathrm{rem}} - |\mathcal{S}_t| q_{\mathrm{floor}}^t$

Relevance-weighted residual allocation:
$\hat{p}^t \leftarrow \mathrm{Softmax}(z^t/\tau)$
for $i \in \mathcal{S}_t$ **do**
  $a_i^t \leftarrow B_{\mathrm{res}}^t \hat{p}_i^t$
end for
$\{b_i^t\}_{i \in \mathcal{S}_t} \leftarrow \mathrm{CALR}\left(\{a_i^t\}, \{P - q_{\mathrm{floor}}^t\}, B_{\mathrm{res}}^t, \{r_i^t\}\right)$
for $i \in \mathcal{S}_t$ **do**
  $q_i^t \leftarrow q_{\mathrm{floor}}^t + b_i^t$
end for

Frontier readout:
$Z_t \leftarrow \mathrm{Concat}_{i=1}^{V} \mathrm{Read}(H_i, q_i^t)$
return Z_t

LagerNVS, LVSM Decoder-Only, and CLiFT use a per-GPU batch size of 4 on four GPUs, giving a global batch size of 16. Gradient accumulation is not used.

B.2 TACC Inference Procedure

Algorithm 1 specifies the target-conditioned allocation and frontier readout described in Sec. 3.3. Each backbone provides the center-pixel ray directions and geometric descriptors required for allocation, as detailed in Appendix B.3. All evaluated settings satisfy $V \geq 2$ and $2P \leq B \leq VP$.

The raw geometric distance is $\tilde{\delta}_i^t = \|\xi_i - \xi_t\|_2$, which is normalized over the source views as

$$\delta_i^t = \frac{\tilde{\delta}_i^t - \min_j \tilde{\delta}_j^t}{\max_j \tilde{\delta}_j^t - \min_j \tilde{\delta}_j^t + \epsilon_\delta}, \qquad \delta_i^t \equiv \delta(\xi_i, \xi_t). \tag{13}$$

For the standardization in Eq. (6), the population standard deviation is computed as $\sigma_{\mathcal{S}_t} = \sqrt{|\mathcal{S}_t|^{-1} \sum_{i \in \mathcal{S}_t} (r_i^t - \mu_{\mathcal{S}_t})^2}$. We use the same λ_d and τ as in Sec. 3.3, and set $\epsilon_\delta = 10^{-6}$ and $\epsilon_s = 10^{-8}$. The normalized entropy is clipped to $[0, 1]$ as a numerical safeguard.

Algorithm 2 Capacity-Aware Largest-Remainder Allocation.

Input: Soft quotas $\{a_i\}_{i\in\mathcal{S}}$, capacities $\{c_i\}_{i\in\mathcal{S}}$, target total B_{res}, and relevance scores $\{r_i\}_{i\in\mathcal{S}}$.
Output: Integer allocations $\{b_i\}_{i\in\mathcal{S}}$ satisfying $\sum_i b_i = B_{\text{res}}$ and $0 \le b_i \le c_i$.
for $i \in \mathcal{S}$ **do**
  $b_i \leftarrow \min(\lfloor a_i \rfloor, c_i)$
end for
$R \leftarrow B_{\text{res}} - \sum_{i\in\mathcal{S}} b_i$
$\bar{r} \leftarrow \text{MinMax}(\{r_i\}_{i\in\mathcal{S}})$
for $i \in \mathcal{S}$ **do**
  $s_i \leftarrow \text{frac}(a_i) + 10^{-6}\bar{r}_i$
end for
$\pi \leftarrow \text{ArgsortDescending}(\{s_i\}_{i\in\mathcal{S}})$
while $R > 0$ **do**
  for $i \in \pi$ **do**
    if $R > 0$ and $b_i < c_i$ **then**
      $b_i \leftarrow b_i + 1$
      $R \leftarrow R - 1$
    end if
  end for
end while
return $\{b_i\}_{i\in\mathcal{S}}$

Algorithm 2 details the capacity-aware largest-remainder allocation used in Algorithm 1. It converts soft residual quotas into integer allocations while respecting per-view capacities and the exact residual budget. Relevance serves only as a small secondary ranking term.

B.3 Backbone Integration Details

Camera geometry for allocation. All three backbone adapters use the center-pixel ray direction for the directional term in Eq. (5). The geometric descriptor ξ used for the distance term is the center-ray Plücker moment for LagerNVS, the center-ray origin for LVSM Decoder-Only, and the camera center from the co-normalized c2w pose for CLiFT. For CLiFT, camera normalization is computed separately for each source-view setting using its source set. The resulting coordinate transform and scale are applied jointly to the source and target poses before extracting their camera centers.

LagerNVS. For LagerNVS, X_i is the final per-view scene representation produced by the Reconstructor after camera-conditioned VGGT processing. Unlike initial image patch embeddings, these tokens already incorporate cross-view interactions. TACC is inserted between the Reconstructor and Renderer. The cached Carrier hierarchies provide Z_t as scene memory for the unchanged Renderer, while target rays follow the original target-embedding pathway.

LVSM Decoder-Only. For LVSM Decoder-Only, TACC is inserted after source-image tokenization and before source and target tokens are concatenated for joint Transformer processing. Each X_i consists of posed-image patch embeddings derived from the source RGB image and its Plücker rays, before joint source–target processing. TACC operates only on the source-token stream; target Plücker rays are independently converted into target tokens. The resulting Z_t is concatenated with the target tokens and passed through the unchanged joint Transformer, after which only the target-token outputs are decoded into the rendered image.

CLiFT. For CLiFT, X_i is the dense per-view source representation produced by the source encoder. Progressive Carrier encoding and target-conditioned frontier readout replace CLiFT's native clustering-and-NeuralCondenser compression pathway, rather than introducing an additional compression stage after its compact representation. For each target, Z_t serves directly as scene memory for the original Transformer decoder, while target Plücker rays follow the native target-embedding pathway. The subsequent normalization and image reconstruction modules remain unchanged. The $A = 6$ budget constrains the source scene-memory tokens visible to each target.

B.4 View Selection Protocol

We define the target-view sets and nested source-view contexts used in training and evaluation. The candidate range is the first 180 valid views for LagerNVS on DL3DV, the first 200 valid views for LagerNVS and LVSM Decoder-Only on RE10K, and the first 100 valid views for CLiFT on DL3DV.

Let $\mathcal{F} = \{j_0, \dots, j_{M-1}\}$ denote the ordered valid frames of a scene. Each stored pose $T_j = T_j^{\mathrm{c2w}}$ is used in camera-to-world form. The camera center and world-space viewing direction are extracted from its translation column and negative local z-axis:

$$\mathbf{c}_j = T_j[:3, 3], \qquad \mathbf{f}_j = \frac{-T_j[:3, 2]}{\|T_j[:3, 2]\|_2}. \tag{14}$$

These quantities determine view indices only. Camera normalization and ray construction are performed after the source and target indices are fixed.

We assign each frame a one-dimensional cumulative pose coordinate. Let α be the median non-zero translation distance between adjacent valid frames. For adjacent frames j_{k-1} and j_k, define

$$\delta_k = \|\mathbf{c}_{j_k} - \mathbf{c}_{j_{k-1}}\|_2 + 0.1\,\alpha\,\frac{\arccos\big(\mathrm{clip}\big(\mathbf{f}_{j_k}^{\mathsf{T}}\mathbf{f}_{j_{k-1}}, -1, 1\big)\big)}{\pi}. \tag{15}$$

The normalized cumulative coordinate is

$$s_{j_0} = 0, \qquad s_{j_k} = \frac{\sum_{\ell=1}^{k}\delta_\ell}{\sum_{\ell=1}^{M-1}\delta_\ell}, \qquad k = 1, \dots, M-1. \tag{16}$$

For a view set $\mathcal{U}$ with pose coordinates $\{s_j\}$, $\mathrm{ProgressiveFPS}(\mathcal{U})$ denotes a deterministic progressive farthest-point ordering. We first select the views with the minimum and maximum pose coordinates. Each subsequent view is selected by

$$j^\star = \arg\max_{j \in \mathcal{U}\setminus\mathcal{S}} \min_{k \in \mathcal{S}} |s_j - s_k|, \tag{17}$$

where $\mathcal{S}$ is the set of previously selected views. Ties are resolved by the smaller frame index.

Target selection. We construct a 40-view target pool using the reference coordinates

$$u_r = \frac{r + 0.5}{40}, \qquad r = 0, \dots, 39. \tag{18}$$

For each u_r, we select the remaining frame that minimizes $|s_j - u_r|$, resolving ties by the smaller frame index. The selected frames form $\mathcal{T}_{40}$.

For a required target count $N \le 40$, the standard target set is

$$\mathcal{T}_N = \mathrm{SortIndex}(\mathrm{Prefix}_N(\mathrm{ProgressiveFPS}(\mathcal{T}_{40}))). \tag{19}$$

Thus, $\mathcal{T}_{20}$ contains the first 20 views in the progressive farthest-point ordering of $\mathcal{T}_{40}$, sorted by frame index.

LagerNVS uses $\mathcal{T}_{20}$ for DL3DV evaluation. The RE10K evaluations of LagerNVS and LVSM Decoder-Only use $\mathcal{T}_{12}$, while CLiFT uses $\mathcal{T}_{20}$. Within each scene and evaluation setting, the target set is fixed across source-view counts.

Source-view selection. We exclude the entire target pool $\mathcal{T}_{40}$ from the source candidates and construct a single progressive ordering:

$$\mathcal{O}_s = \mathrm{ProgressiveFPS}\left(\mathcal{F} \setminus \mathcal{T}_{40}\right). \tag{20}$$

For a required source-view count K, the source context is

$$\mathcal{C}_K = \mathrm{SortIndex}(\mathrm{Prefix}_K(\mathcal{O}_s)). \tag{21}$$

All source contexts are prefixes of the same ordering and are therefore nested:

$$K_1 < K_2 \quad \Longrightarrow \quad \mathcal{C}_{K_1} \subset \mathcal{C}_{K_2}. \tag{22}$$

Since the standard target sets are subsets of $\mathcal{T}_{40}$ and all source views belong to $\mathcal{F} \setminus \mathcal{T}_{40}$, source and target views are strictly disjoint. The target-count experiment in Appendix C.5 uses a separate target-selection protocol defined there rather than Eq. (19).

B.5 BASELINE IMPLEMENTATION

For each backbone, TACC and the adapted compression baselines operate on the same source representation X defined in Sec. 3.1. The decoder-visible budget $B = A \cdot P$ counts source-side tokens only. Target-query, target-ray, target-pose, and backbone-internal register tokens remain unchanged and are excluded from budget matching. ToMe and DivPrune are training-free; the training protocols for CLiFT Native and ZPressor are specified below. Tab. 5 summarizes the exact-budget implementation of the adapted compression baselines.

Table 5: **Baseline implementation and exact-budget matching.** Each compressed baseline exposes exactly $B = A \cdot P$ source-side tokens.

Method	Exact-budget realization
ToMe	Each source view receives $\lfloor B/V \rfloor$ tokens, with one additional token assigned to each of the first $B \bmod V$ views. BSM and mass-weighted merging are applied within each view until its assigned quota is reached.
DivPrune	Greedy max–min diversity selection is applied jointly to all source tokens. Exactly B tokens are retained and restored to their original sequence order before rendering.
ZPressor	$A = B/P$ anchor views are used, each producing a complete P-token grid. The resulting representation contains exactly $AP = B$ source tokens.

CLiFT Native training protocol. For the controlled comparison in Tab. 3, we retrain CLiFT's native compression pathway on the same 2,048 DL3DV scenes as TACC. Both methods use the same source-view schedule ($V \in \{24, 36, 48\}$), target-sampling protocol, and fixed $A = 6$ decoder-visible budget. Training runs for 24 epochs, totaling 3,072 optimizer steps, with a global batch size of 16 across four GPUs.

Both methods share the same pretrained source encoder and renderer, which remain frozen throughout training. Only their respective compression modules are optimized, using method-specific training objectives and optimization settings. Validation uses the same 60 held-out scenes and all 20 target views per source-view setting; checkpoint selection uses equal-weight macro PSNR across the three settings. Unlike CLiFT's official second-stage training, the renderer is not fine-tuned in this controlled comparison.

ZPressor training protocol. For each backbone, ZPressor uses the same training split, source-view schedule, target-sampling protocol, validation split, and batch configuration as TACC. Only the compressor is trained; the pretrained NVS backbone remains frozen. We retain the ZPressor training objective and optimization configuration.

LagerNVS. ZPressor is trained on 10,139 DL3DV scenes with source-view counts $V \in \{60, 80, 100, 120\}$ and a fixed decoder-visible budget of $A = 10$. Validation uses 200 held-out scenes and four fixed targets per source-view setting. Checkpoint selection minimizes the weighted validation reconstruction objective over the settings with 80, 100, and 120 source views, using weights 0.2, 0.3, and 0.5, respectively. The compressor is evaluated every 2,500 optimization steps, and the checkpoint at step 25,000 is selected. For the tighter-budget evaluation in Appendix C.2, the compressor trained with an $A = 10$ budget is evaluated at $A = 6$ without budget-specific retraining.

LVSM Decoder-Only. ZPressor is trained on 2,048 RE10K scenes with source-view counts $V \in \{80, 100, 120\}$ and $A = 10$. Training targets follow the LVSM protocol in Appendix B.1. Validation uses 128 held-out RE10K scenes and the fixed target indices $\{2, 7, 12, 17\}$ under all three source-view settings; the checkpoint is selected by validation PSNR.

CLiFT. ZPressor is trained on the same DL3DV training split as TACC, with source-view counts $V \in \{24, 36, 48\}$ and $A = 6$. Training targets follow the CLiFT protocol in Appendix B.1. Validation uses 60 held-out scenes and all 20 target views per source-view setting; the checkpoint is selected by equal-weight macro PSNR across the three settings.

All ZPressor compressors are trained for 25,000 optimization steps using AdamW with an initial learning rate of 1×10^{-4}, $\beta_1 = 0.9$, $\beta_2 = 0.95$, and weight decay 0.05. A 1,000-step linear warmup is followed by cosine decay to 1×10^{-5}.

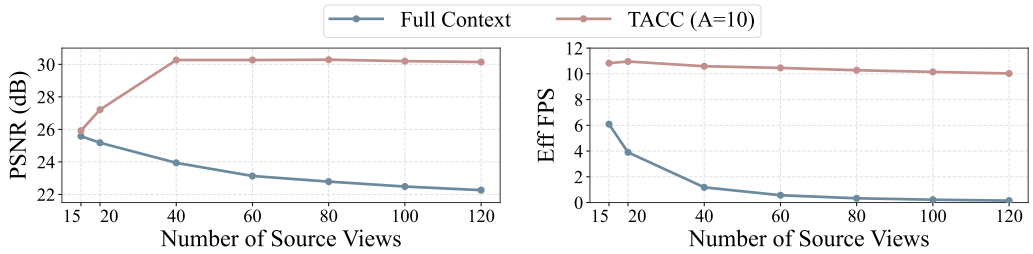


Figure 7: **LVSM Decoder-Only with increasing source views.** PSNR (left) and Effective FPS (right) on RE10K across 15–120 source views. TACC uses a fixed $A = 10$ decoder-visible budget, whereas Full Context processes all selected source tokens.

LagerNVS and LVSM Decoder-Only use a per-GPU batch size of 4 on four GPUs, giving a global batch size of 16. CLiFT uses a per-GPU batch size of 1 on four GPUs, giving a global batch size of 4.

C Additional Results

C.1 Additional Evaluation on LVSM Decoder-Only

We extend the LVSM Decoder-Only evaluation to 15–120 source views using the same 1,633-scene RE10K test set as in the main experiments. Each scene is restricted to its first 200 valid views, with 12 target views evaluated at 256×256 resolution and a target-view batch size of 4. Full Context and TACC use identical source and target views, which are disjoint by construction. TACC uses a fixed $A = 10$ decoder-visible budget throughout this evaluation and reuses the Carrier checkpoint trained with $V \in \{80, 100, 120\}$ without source-count-specific fine-tuning.

As shown in Fig. 7, TACC's reconstruction quality increases up to 40 source views and remains nearly stable thereafter, with little variation in Effective FPS. In contrast, the frozen Full Context model shows decreasing reconstruction quality and target-time throughput as its source context grows. At 120 source views, TACC reaches 30.14 dB PSNR, compared with 22.26 dB for Full Context, while maintaining the fixed decoder-visible budget.

The Full Context results reflect a substantial shift from the pretrained LVSM setting: the scene-level backbone is trained with 2 source views (Jin et al., 2025), whereas the 120-view evaluation uses $60\times$ more source views and exposes 122,880 source patch tokens at 256×256 resolution. This larger context increases rendering cost and may contribute to the observed quality degradation. Related observations in VGGT suggest that additional redundant views may dilute informative geometric evidence rather than consistently improve reconstruction quality (Park et al., 2026). Our comparison therefore concerns the frozen pretrained backbone, rather than a model retrained for dense source inputs.

Carrier storage. For LVSM Decoder-Only, the cached node-Carriers form a BF16 tensor of shape $V \times (2P - 1) \times C$, where $P = 1024$ and $C = 768$. At 120 source views, this tensor occupies 359.82 MiB. This calculated footprint excludes other retained tensors, hierarchy metadata, and temporary allocations; it does not represent the total Carrier cache or incremental peak GPU-memory overhead. The Carrier hierarchies are constructed once per source set and reused across target views.

Target-aware Top-A control. To examine whether selecting relevant complete views alone accounts for TACC's reconstruction quality, we evaluate a Top-A Full Views control at 120 source views. For each target, this control retains the ten highest-ranked source views according to TACC's geometric relevance score, without Carrier compression or cross-view token allocation. Both methods expose exactly $A = 10$ complete-view equivalents to the decoder.

As shown in Tab. 6, TACC substantially improves reconstruction quality over Top-A Full Views while maintaining comparable target-time throughput. This result shows that selecting target-relevant complete views alone is insufficient to match TACC under the same decoder-visible budget.

Table 6: **Target-aware Top-A control on LVSM Decoder-Only at 120 source views.** Both methods use the same $A = 10$ decoder-visible budget.

Method	PSNR ↑	SSIM ↑	LPIPS ↓	Eff. FPS ↑
Top-A Full Views	26.7087	0.8517	0.1154	**11.11**
TACC	**30.1441**	**0.9243**	**0.0750**	10.03

C.2 Evaluation under a Tighter Decoder-Visible Budget

We further evaluate LagerNVS on DL3DV with 60, 80, and 100 source views under a tighter $A = 6$ decoder-visible budget. TACC reuses the Carrier checkpoint trained with an $A = 10$ budget without budget-specific fine-tuning; the ZPressor checkpoint follows the protocol in Appendix B.5.

As shown in Tab. 7, TACC achieves the highest reconstruction quality among compressed methods across all three source-view settings. At 60 and 80 source views, it also exceeds the frozen Full Context checkpoint on all three reconstruction metrics. At 100 source views, where the source-to-budget ratio reaches approximately $16.7\times$, Full Context retains a slightly higher PSNR, while TACC achieves higher SSIM and lower LPIPS than all compared methods. TACC maintains Effective FPS near 40 across the three settings.

Table 7: **Comparison on DL3DV with a tighter compression budget.** Compressed methods use a fixed $A = 6$ decoder-visible budget, while Full Context uses the complete LagerNVS representation.

	60 Source Views				80 Source Views				100 Source Views			
Method	PSNR ↑	SSIM ↑	LPIPS ↓	Eff. FPS ↑	PSNR ↑	SSIM ↑	LPIPS ↓	Eff. FPS ↑	PSNR ↑	SSIM ↑	LPIPS ↓	Eff. FPS ↑
LagerNVS-based												
Full Context	26.2985	0.8247	0.0892	5.87	25.6766	0.8064	0.0963	4.40	25.2932	0.7953	0.1012	3.57
ToMe	19.5020	0.5217	0.3859	47.93	18.8116	0.4944	0.4265	47.23	18.6572	0.4877	0.4408	47.61
DivPrune	21.1870	0.6289	0.2364	47.97	21.0042	0.6201	0.2429	47.20	20.8862	0.6148	0.2472	47.93
ZPressor	23.7222	0.7653	0.1638	47.61	23.5503	0.7589	0.1664	47.93	23.4514	0.7551	0.1671	47.58
Ours	27.1015	0.8623	0.0819	40.40	26.6201	0.8504	0.0827	39.52	25.0192	0.8040	0.0954	40.01

C.3 Additional Results on CLiFT

We additionally evaluate CLiFT's official pretrained Native checkpoint under the same DL3DV evaluation protocol as Tab. 3. Tab. 8 reports its reconstruction quality and target-time throughput across 24, 36, and 48 source views. These results provide a reference for the released CLiFT model. Unlike the Native baseline in Tab. 3, the official checkpoint is not retrained under our matched compression-module training protocol.

Table 8: **Evaluation of the official CLiFT Native checkpoint on DL3DV.** Results use the same source and target views as Tab. 3, with a fixed $A = 6$ decoder-visible budget.

Source Views	PSNR ↑	SSIM ↑	LPIPS ↓	Eff. FPS ↑
24	19.9137	0.5764	0.3070	47.33
36	19.8713	0.5719	0.3137	47.46
48	19.8664	0.5718	0.3142	47.08

Fig. 8 provides complementary qualitative results for the CLiFT comparison.



Figure 8: **Additional qualitative results on CLiFT with DL3DV.** Native uses the checkpoint retrained under the matched setting described in Appendix B.5. Compressed methods use a fixed $A = 6$ decoder-visible budget, while Full Context uses all source tokens. TACC better preserves local structures and fine-grained details.

C.4 Additional Ablations and Controls

Carrier and allocation controls. We conduct additional experiments using LagerNVS on DL3DV with 80 source views and a fixed $A = 10$ decoder-visible budget. In Tab. 4, *w/o Carrier* retains TACC's per-view quotas but replaces Carrier readout with approximately uniform spatial subsampling of the original tokens. *Uniform Alloc.* instead uses approximately uniform, exact-budget quotas during both training and evaluation; its Carrier is retrained following the protocol in Appendix B.1.

We further evaluate *Residual PMA + Alloc.*, which replaces the learned regional Carriers with residual Pooling by Multihead Attention (PMA) aggregation (Lee et al., 2019) while retaining TACC's target-conditioned per-view allocation. The PMA module is independently trained under the same training and evaluation protocol as TACC, using identical rendering and quality-gated RGB distillation losses and weights, but without the Carrier-specific reconstruction regularizer.

As shown in Tab. 9, under the shared allocation rule, Residual PMA + Alloc. achieves 26.91 dB PSNR, compared with 27.34 dB for TACC, with the same reported LPIPS in this setting.

Table 9: **Residual PMA control on DL3DV.** Both variants use LagerNVS with 80 source views and the same $A = 10$ decoder-visible budget.

Variant	PSNR ↑	SSIM ↑	LPIPS ↓
Residual PMA + Alloc.	26.9088	0.8550	0.0791
TACC	**27.3415**	**0.8649**	**0.0791**

Relevance-score components. We isolate the directional and distance terms in Eq. (5) on the official 140-scene DL3DV benchmark, using LagerNVS with 80 source views, 20 target views, and $A = 10$. *Direction Only* uses $r_i^t = \langle \hat{d}_i, \hat{d}_t \rangle$, whereas *Moment Only* uses $r_i^t = -\delta(m_i, m_t)$. All other inference components remain unchanged.

As shown in Tab. 10, moment distance yields higher reconstruction quality than directional similarity alone in this setting. Combining the two terms further improves PSNR from 27.21 to 27.34 dB, with corresponding improvements in SSIM and LPIPS.

Table 10: **Relevance-score ablation on DL3DV.** All variants use LagerNVS with 80 source views and a fixed $A = 10$ decoder-visible budget.

Variant	PSNR ↑	SSIM ↑	LPIPS ↓
Direction Only	26.8438	0.8507	0.0899
Moment Only	27.2093	0.8600	0.0832
Full Score	**27.3415**	**0.8649**	**0.0791**

C.5 Target-Time and End-to-End Runtime Analysis

Target-time profiling. We measure target-time latency on an NVIDIA A800 GPU under BF16 inference. One complete scene is used for warm-up and excluded from measurement. GPU execution is synchronized at the boundaries of each timed region. Stage-wise profiling and end-to-end timing are performed in separate runs to avoid introducing profiling synchronization into the end-to-end measurements.

For the target-time analysis, we use the official 140-scene DL3DV evaluation set and the Front180 protocol described in Appendix B.4, with 20 target views per scene and a target-view batch size of 4. We profile allocation, frontier readout, and decoder execution separately. For each stage, we accumulate latency across all target batches of a scene, average over the 140 scenes, and divide by the 20 rendered target views. Effective FPS is computed from the corresponding target-time latency.

Table 11: **Target-time runtime analysis on DL3DV.** Full Context processes all source tokens, whereas TACC exposes a fixed $A = 10$ representation to the decoder. Latencies are reported in milliseconds per target view.

Source Views	Alloc.+Readout	TACC Decoder	Full Decoder	Decoder Speedup	Eff. FPS
80	3.045	32.737	227.035	6.94×	27.95
100	3.094	32.468	279.945	8.62×	28.12
120	3.344	32.419	334.353	10.31×	27.96

As shown in Tab. 11, the fixed decoder-visible budget keeps TACC's decoder latency nearly unchanged as the source context grows from 80 to 120 views, whereas Full Context becomes progressively slower. Allocation and frontier readout introduce a small per-target overhead, allowing TACC to maintain nearly constant Effective FPS across these settings.

Amortized end-to-end throughput. Effective FPS excludes source encoding and the one-time Carrier construction. To examine the amortized cost of these operations, we evaluate LagerNVS on the 140-scene DL3DV benchmark and LVSM Decoder-Only on the 1,633-scene RE10K evaluation set. Both use 80 source views and a fixed $A = 10$ decoder-visible budget for TACC, while varying the number of rendered targets.

Both backbones follow the same target-list construction procedure, using the first 180 valid frames for LagerNVS on DL3DV and the first 200 valid frames for LVSM Decoder-Only on RE10K, as defined in Appendix B.4. For each scene, we begin with the stored 40-view target list in ascending frame-index order. We then append 60 additional targets from the remaining candidate frames, excluding the fixed 80 source views and previously selected targets. These additional views are selected sequentially using deterministic farthest-first selection along the cumulative pose coordinate defined in Appendix B.4.

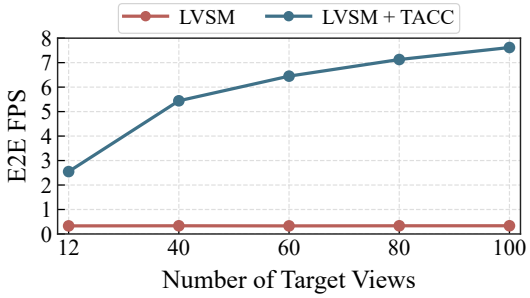


Figure 9: **Amortized E2E throughput on LVSM Decoder-Only.** E2E FPS on RE10K with 80 source views and a fixed $A = 10$ decoder-visible budget for TACC.

For LagerNVS, we evaluate target counts $K \in \{10, 40, 60, 80, 100\}$; for LVSM Decoder-Only, we use $K \in \{12, 40, 60, 80, 100\}$. Each target set contains the first K entries of the corresponding ordered 100-view list. The target sets are therefore nested and disjoint from the fixed source views. Full Context and TACC use identical source and target lists at every target count. TACC computes relevance scores and per-view quotas independently for each target; the target-view batch size of 4 affects decoder batching but not target selection.

We measure E2E runtime from source-view processing through target RGB prediction, including the complete model computation but excluding disk loading and image decoding. Source encoding is performed once per source set for both methods, and TACC additionally constructs the Carrier hierarchies once. Each of the 140 DL3DV scenes and 1,633 RE10K scenes is measured once at every target count under the corresponding backbone. E2E FPS is computed as the total number of rendered targets divided by the total measured wall-clock time.

The LagerNVS E2E results are presented in Fig. 4(c). The complementary LVSM Decoder-Only results in Fig. 9 show higher E2E throughput for TACC across all evaluated target counts.

C.6 Sensitivity to Common Translation of Allocation Coordinates

For LagerNVS, the distance term in Eq. (5) uses the Plücker moment $m = o \times d$ of the center ray. This quantity depends on the coordinate origin: under a common translation u applied to all camera origins, $o' = o + u$, the moment becomes $m' = m + u \times d$, while d remains unchanged. We therefore evaluate the sensitivity of the LagerNVS allocation rule to common translations applied only to the Plücker rays used for relevance computation.

We use LagerNVS on DL3DV with 60 source views, 20 target views, and an $A = 10$ decoder-visible budget. The perturbation is applied after the standard LagerNVS camera normalization, which recenters poses with respect to the first camera and scales camera translations by 1.35 times the maximum source-camera translation norm. For each scene, let ρ denote the maximum distance of the 60 source and 20 target camera centers from their joint centroid in this normalized coordinate system. We apply $u = \rho\gamma$, where the table labels specify the per-axis coefficient γ; for example, $x = 0.5$ denotes $\gamma = (0.5, 0, 0)$, while $xyz = 0.5$ denotes $\gamma = (0.5, 0.5, 0.5)$. The same u is applied to all source and target Plücker rays used for allocation within a scene. Source and target view indices, rendering inputs, evaluation images, and model parameters remain unchanged.

Let $q_{i,t}^{(0)}$ and $q_{i,t}^{(u)}$ denote the quota assigned to source view i for target view t before and after translation, respectively, with $\sum_i q_{i,t}^{(0)} = \sum_i q_{i,t}^{(u)} = B$. We measure the allocation change for each

Table 12: **Sensitivity of LagerNVS allocation to common translations of the allocation coordinates.** Translation labels denote scene-radius-scaled per-axis coefficients. Top-1 Change measures changes in the highest-relevance source view, while Quota TV measures the normalized total variation of the resulting per-view allocation.

Translation	**PSNR ↑**	**SSIM ↑**	**LPIPS ↓**	**Top-1 Change ↓**	**Quota TV ↓**
None	27.5815	0.8691	0.0792	–	–
$x = 0.1$	27.5854	0.8692	0.0792	1.04%	0.292%
$x = 0.5$	27.5722	0.8692	0.0792	4.64%	0.700%
$x = 1.0$	27.5505	0.8685	0.0796	7.96%	1.052%
$y = 0.5$	27.5829	0.8692	0.0793	4.04%	0.602%
$z = 0.5$	27.5678	0.8686	0.0797	3.86%	0.554%
$xyz = 0.5$	27.5585	0.8686	0.0797	6.07%	0.807%

target view as

$$\mathrm{TV}_t = \frac{1}{2B} \sum_{i=1}^{V} \left| q_{i,t}^{(u)} - q_{i,t}^{(0)} \right|, \qquad \mathrm{Top1}_t = \mathbf{1}\left[\arg\max_i r_{i,t}^{(u)} \neq \arg\max_i r_{i,t}^{(0)} \right]. \tag{23}$$

Quota TV and Top-1 Change are averaged over all $140 \times 20 = 2{,}800$ scene–target-view pairs and reported as percentages.

As shown in Tab. 12, common translations produce modest changes in the resulting allocation. The largest mean Quota TV is 1.052%, attained for $x = 1.0$, where the highest-relevance source view changes for 7.96% of the evaluated targets. The corresponding reconstruction metrics vary little, with a maximum PSNR decrease of approximately 0.031 dB and similarly small changes in SSIM and LPIPS. Although the LagerNVS moment-distance term is not strictly translation invariant, its allocation remains empirically stable under the tested coordinate perturbations.