STAC: Plug-and-Play Spatio-Temporal Aware Cache Compression for Streaming 3D Reconstruction

Supplementary Material

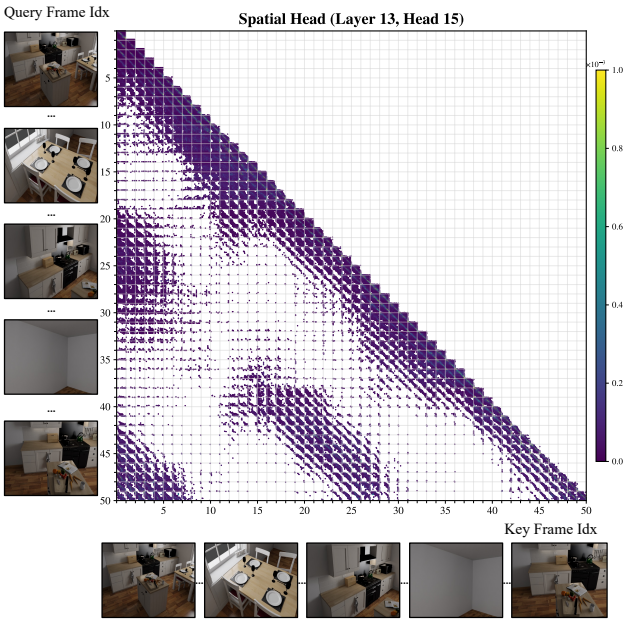

Figure 1. Visualization of a spatial attention head, where the top 1024 relevant key tokens are retained for each query. The axes are labeled at the frame level to facilitate identification of the corresponding frames.

This supplementary material provides additional empirical observations, technical details, and extended experiments that complement the main paper.

A. KV Cache Similarity

Building on the observation that spatial attention heads in Causal-VGGT [5, 16] are highly sensitive to viewpoint changes (Fig. 1), we investigate the spatial properties of their key/value (KV) tokens. In the context of streaming 3D reconstruction, the camera frequently revisits previously observed regions. When cache management relies solely on temporal correlations across viewpoints, relevant tokens from earlier views may be prematurely evicted due to viewpoint shifts. This motivates a deeper analysis of the spatial structure and similarity present in the KV cache.

We hypothesize that tokens in Causal-VGGT are aligned with the 3D scene structure, exhibiting both local redundancy[1] and distance-dependent similarity. To validate this hypothesis, we associate each key/value pair $m_i = \{k_i, v_i\}$ with a 3D coordinate in the reconstructed scene and discretize these tokens into a voxel grid with fixed spatial resolution. We then analyze: (i) *intra-voxel similarity* (Fig. 2), quantifying redundancy among tokens within the same spatial region; (ii) *inter-voxel similarity* (Fig. 3), characterizing how similarity decays with spatial separation; and (iii) *layer-wise similarity* (Fig. 4), measuring how consistently local redundancy is preserved across layers. Together, these analyses show that KV cache is locally redundant and spatially structured across Causal-VGGT's transformer layers, directly motivating the voxel-wise cache compression used in the **STAC** framework.

[1] In our context, *local similarity* and *redundancy* refer to the same phenomenon of high feature similarity among spatially proximal tokens.

A.1. Intra-voxel Similarity

We first assess intra-voxel similarity to quantify local redundancy in the KV cache and understand how token representations cluster within the same spatial region.

For each voxel $u \in \mathcal{V}$, we collect the set of T_u tokens:

$$\mathcal{M}_u = \{m_{u,1}, m_{u,2}, \ldots, m_{u,T_u}\} \tag{1}$$

where each $m_{u,i}$ is a token assigned to voxel u.

We compute the pairwise cosine similarity between tokens as follows:

$$\begin{aligned} \cos_k(m_i, m_j) &= \frac{k_i \cdot k_j}{\|k_i\|\|k_j\|}, \\ \cos_v(m_i, m_j) &= \frac{v_i \cdot v_j}{\|v_i\|\|v_j\|}. \end{aligned} \tag{2}$$

To capture the internal structure within a voxel, we perform K-means clustering [4] over the key-state tokens in $\mathcal{M}_u$, partitioning them into n clusters:

$$\text{Cluster}_u = \{\mathcal{C}_{u,1}, \mathcal{C}_{u,2}, \ldots, \mathcal{C}_{u,n}\}, \quad \mathcal{C}_{u,i} \subseteq \mathcal{M}_u \tag{3}$$

The intra-voxel similarity score is defined as the average cosine similarity between each token and its corresponding cluster centroid within a voxel:

$$\begin{aligned} \text{IntraSim}_k(u; n) &= \frac{1}{N_u(n)} \sum_{i=1}^{n} \sum_{m \in \mathcal{C}_{u,i}} \cos_k(\hat{m}_{u,i}, m), \\ \text{IntraSim}_v(u; n) &= \frac{1}{N_u(n)} \sum_{i=1}^{n} \sum_{m \in \mathcal{C}_{u,i}} \cos_v(\hat{m}_{u,i}, m), \end{aligned} \tag{4}$$

where $\hat{m}_{u,i} = \text{Mean}(\mathcal{C}_{u,i})$ is the centroid of cluster $\mathcal{C}_{u,i}$, and

$$N_u(n) = \sum_{i=1}^{n} |\mathcal{C}_{u,i}| \tag{5}$$

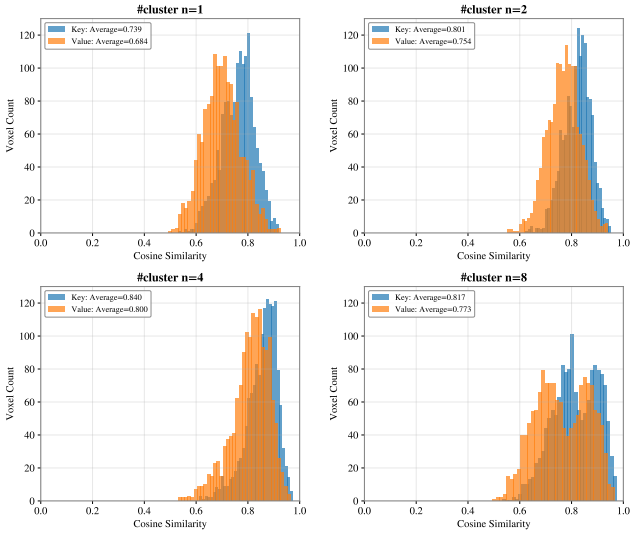

Figure 2. **Intra-voxel similarity distribution vs. cluster count.** Distribution of intra-voxel similarity scores across voxels under varying cluster counts $n \in \{1, 2, 4, 8\}$. Each score reflects the average cosine similarity of key/value tokens within voxel clusters.

is the total number of tokens across all clusters in voxel u.

To evaluate the effect of clustering granularity, we compute intra-voxel similarity distributions across all voxels in a scene under varying cluster counts, as shown in Fig. 2. The results reveal that key/value tokens within the same voxel generally exhibit strong mutual similarity, suggesting notable local redundancy in the KV cache. As the number of clusters n increases, intra-cluster similarity improves due to finer partitioning. However, beyond $n = 4$, the gains diminish, and smaller cluster sizes introduce noise due to insufficient token samples. Empirically, we find that $n = 4$ provides a good trade-off between representational granularity and robustness. Notably, value-state tokens follow a similar trend, showing consistent behavior with their corresponding key tokens across different clustering levels.

A.2. Inter-voxel Similarity

We next evaluate inter-voxel similarity to characterize how token representations evolve across spatially distinct regions. By measuring feature similarity between tokens in different voxels, we aim to uncover the degree of spatial locality encoded in the KV cache and assess how it correlates with 3D distance.

Given two voxels u_a and u_b with associated token sets $\mathcal{M}_{u_a}$ and $\mathcal{M}_{u_b}$, we define their inter-voxel similarity in the key and value embedding spaces as follows:

$$\mathrm{InterSim}_k(u_a, u_b) = \frac{1}{|\mathcal{M}_{u_a}|\,|\mathcal{M}_{u_b}|} \sum_{\substack{m_a \in \mathcal{M}_{u_a} \\ m_b \in \mathcal{M}_{u_b}}} \cos_k(m_a, m_b),$$

$$\mathrm{InterSim}_v(u_a, u_b) = \frac{1}{|\mathcal{M}_{u_a}|\,|\mathcal{M}_{u_b}|} \sum_{\substack{m_a \in \mathcal{M}_{u_a} \\ m_b \in \mathcal{M}_{u_b}}} \cos_v(m_a, m_b). \tag{6}$$

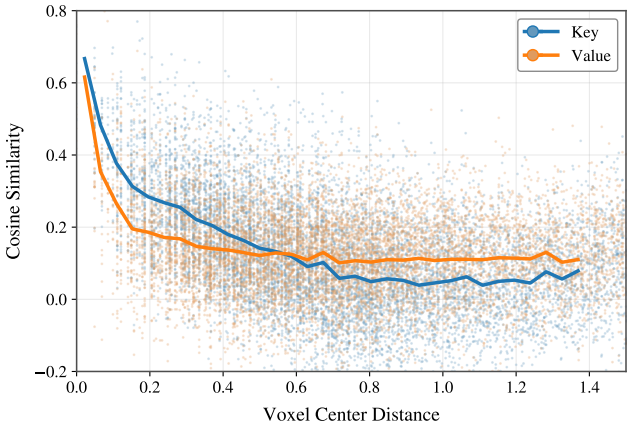

Figure 3. **Inter-voxel similarity vs. 3D distance.** Mean and samples of $\mathrm{InterSim}(u_a, u_b)$ are shown as a function of Euclidean distance between voxel centers, averaged over sampled voxel pairs across scenes.

To analyze spatial trends, we sample voxel pairs from the 3D grid, compute their inter-voxel similarities, and measure the Euclidean distance between voxel centers. As shown in Fig. 3, similarity consistently decays with increasing distance: nearby voxels share highly similar key/value tokens, while distant ones become nearly orthogonal in feature space. This confirms that Causal-VGGT preserves spatial locality in its KV cache, yielding geometrically structured and semantically coherent representations.

Such findings reinforce the intuition behind our voxel-wise caching and compression scheme: restricting attention and aggregation to local 3D neighborhoods not only aligns with the spatial correlation of features but also mitigates the risk of conflating semantically unrelated content. This spatially localized organization thus serves as a principled foundation for efficient and effective representation compression in voxel space.

A.3. Layer-wise Similarity

We further analyze the intra-voxel similarity across different layers of the Causal-VGGT model. For each layer, we compute the similarity score per attention head and average them, as shown in Fig. 4. The results indicate that Causal-VGGT consistently exhibits high intra-voxel similarity across all layers. We attribute this behavior to

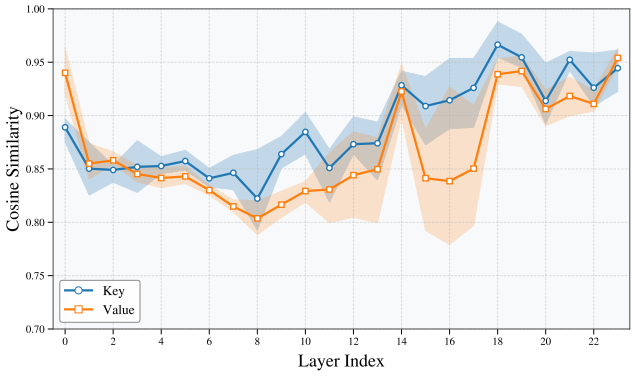

Figure 4. **Layer-wise intra-voxel similarity in Causal-VGGT.** Average intra-voxel similarity scores computed across all attention heads per layer.

the model being trained specifically for 3D reconstruction tasks, which encourages the emergence of spatially coherent features. As a result, attention heads tend to capture features aligned with the same 3D regions across different views, leading to strong local redundancy in the key/value tokens at all layers. Given this consistency, our **STAC** framework adopts a unified cache compression strategy across all layers, without introducing layer- or head-specific selection heuristics.

B. Details of Token Merging

B.1. Online Token Merging Algorithm

Algorithm 1 summarizes the online token merging procedure within a local voxel u. The goal is to maintain a compact merged token set $\mathcal{G}_u$ in an online streaming setting with limited memory.

B.2. Implementation Details for Token Merging

In our merging module, each cluster $\mathcal{C}_i$ is aggregated around a pivot token m_i^q, selected according to its attention score. For notational simplicity, we omit the voxel index u in the following derivations, since all operations are performed within a fixed local voxel. We adopt a similarity-weighted aggregation scheme, following recent KV compression methods such as D2O [9], so that tokens more similar to the pivot receive larger weights. Formally, the merged token $\hat{m}_i$ is computed as:

$$\hat{m}_i = \frac{1}{Z_i} \sum_{m \in \mathcal{C}_i} \omega(m_i^q, m)\, m, \tag{7}$$

$$\omega(m_i^q, m) = \exp\left(\cos_k(m_i^q, m)\right), \tag{8}$$

$$Z_i = \sum_{m \in \mathcal{C}_i} \omega(m_i^q, m), \tag{9}$$

$$m_i^q = \arg\max_{m \in \mathcal{C}_i} \text{score}(m), \tag{10}$$

Algorithm 1 Online Token Merging

Require: Incoming tokens $\mathcal{M}_u$, merged buffer $\mathcal{G}_u$, evicted buffer $\mathcal{E}_u$, similarity threshold λ

1: **for all** $m \in \mathcal{M}_u$ **do**
2: **if** $\mathcal{G}_u \neq \emptyset$ **then**
3: $\hat{m}^p \leftarrow \arg\max_{\hat{m} \in \mathcal{G}_u} \cos_k(m, \hat{m})$
4: **if** $\cos_k(m, \hat{m}^p) \geq \lambda$ **then** ▷ Merging
5: $w \leftarrow \text{Weight}(\hat{m}^p, m)$
6: $\mathcal{G}_u.\text{merge}(\hat{m}^p, m, w)$
7: **else**
8: $\mathcal{E}_u.\text{append}(m)$
9: **end if**
10: **else**
11: $\mathcal{E}_u.\text{append}(m)$
12: **end if**
13: **if** $|\mathcal{E}_u| = N_{\text{evict}}$ **then** ▷ Aggregation
14: $\hat{m}_{new}, Z(\hat{m}_{new}) \leftarrow \text{Aggregate}(\mathcal{E}_u)$
15: $\mathcal{E}_u.\text{clear}()$
16: **if** $|\mathcal{G}_u| = N_{\text{merge}}$ **then** ▷ Re-merging
17: $\hat{m}^l \leftarrow \arg\min_{\hat{m} \in \mathcal{G}_u} Z(\hat{m})$
18: $\hat{m}^p \leftarrow \arg\max_{\hat{m} \in \mathcal{G}_u \setminus \{\hat{m}^l\}} \cos_k(\hat{m}^l, \hat{m})$
19: $\hat{w}^l \leftarrow e^{-1} Z(\hat{m}^l) \text{Weight}(\hat{m}^p, \hat{m}^l)$
20: $\mathcal{G}_u.\text{merge}(\hat{m}^p, \hat{m}^l, \hat{w}^l)$
21: $\mathcal{G}_u.\text{pop}(\hat{m}^l)$
22: **end if**
23: $\mathcal{G}_u.\text{append}(\hat{m}_{new}, Z(\hat{m}_{new}))$
24: **end if**
25: **end for**

where $\omega(\cdot,\cdot)$ is a cosine-based similarity weight (see Eq. (2)) and Z_i is a normalization term reflecting the information content of the cluster. Each merged token $\hat{m}_i$ is stored in the merged buffer $\mathcal{G}_u$, which maintains compact representatives of recent clusters. In practice, such aggregation is performed only when the evicted buffer $\mathcal{E}_u$ accumulates enough unmatched tokens (Algorithm 1, lines 12–19).

In the online streaming setting, each incoming token is processed on arrival: we first match it to $\mathcal{G}_u$ and merge it when the similarity exceeds λ; otherwise we buffer it in $\mathcal{E}_u$ for later aggregation. Concretely, for an incoming token m, we find its nearest neighbor $\hat{m}^p \in \mathcal{G}_u$ and directly absorb m when $\cos_k(m, \hat{m}^p) \geq \lambda$ (Algorithm 1, lines 3–7). Given intra-cluster coherence, we use the matched representative $\hat{m}^p$ as the reference for similarity weighting:

$$\hat{m}^p \leftarrow \frac{Z(\hat{m}^p)\hat{m}^p + \omega(\hat{m}^p, m)\, m}{Z(\hat{m}^p) + \omega(\hat{m}^p, m)}. \tag{11}$$

Otherwise, m is appended to $\mathcal{E}_u$ and summarized via many-to-one aggregation once the buffer is full, yielding a new representative $\hat{m}_{\text{new}}$.

When $\mathcal{G}_u$ is already at capacity, inserting $\hat{m}_{\text{new}}$ would exceed the budget. We therefore trigger a lightweight *re-*

merging step that frees one slot while preserving information: it fuses the least important token with its most similar neighbor. The least important token $\hat{m}^l$ is determined using the aggregation weight:

$$\hat{m}^l := \hat{m}_i^l = \arg\min_{\hat{m} \in \mathcal{G}_u} Z(\hat{m}), \tag{12}$$

where index i indicates that $\hat{m}_i^l$ was originally aggregated from cluster $\mathcal{C}_i$, and its closest neighbor $\hat{m}^p$ is chosen as:

$$\hat{m}^p := \hat{m}_j^p = \arg\max_{\hat{m} \in \mathcal{G}_u \setminus \{\hat{m}^l\}} \cos_k(\hat{m}^l, \hat{m}), \tag{13}$$

where index j indicates the corresponding cluster $\mathcal{C}_j$.

In an ideal offline setting, the re-merged representation would be computed over the union of the original clusters:

$$\hat{m}^* = \frac{\sum_{m \in \mathcal{C}_i \cup \mathcal{C}_j} \omega(m^{q*}, m)\, m}{\sum_{m \in \mathcal{C}_i \cup \mathcal{C}_j} \omega(m^{q*}, m)}, \tag{14}$$

$$m^{q*} = \arg\max_{m \in \{m_i^q, m_j^q\}} \text{score}(m). \tag{15}$$

However, in our streaming regime, the original tokens are fused and discarded once a cluster is merged, so $\mathcal{C}_i$ and $\mathcal{C}_j$ are no longer accessible and the ideal formulation above is infeasible.

To derive a tractable approximation without storing all original tokens, we leverage intra-cluster coherence and adopt a *small-angle approximation*:

$$m_i \approx m_i^q \approx \hat{m}_i^l, \quad m_j \approx m_j^q \approx \hat{m}_j^p,$$

so that the pivot, the merged token, and individual tokens in a cluster are treated as nearly aligned in feature space.

We also use $Z(\cdot)$ as a proxy for token importance: larger $Z(\hat{m})$ reflects greater accumulated information within the cluster. Combined with Eq. (12), this gives the following choice of the new pivot:

$$m^{q*} \approx \hat{m}_j^p = \arg\max_{\hat{m} \in \{\hat{m}_i^l, \hat{m}_j^p\}} Z(\hat{m}),$$

from which the re-merged token is obtained as:

$$\begin{aligned} \hat{m}^* &\approx \frac{\sum_{m_j \in \mathcal{C}_j} \omega(\hat{m}_j^p, m_j)\, m_j + \sum_{m_i \in \mathcal{C}_i} \omega(\hat{m}_j^p, m_i)\, m_i}{\sum_{m_j \in \mathcal{C}_j} \omega(\hat{m}_j^p, m_j) + \sum_{m_i \in \mathcal{C}_i} \omega(\hat{m}_j^p, m_i)} \\ &= \frac{Z_j \hat{m}_j^p + \sum_{m_i \in \mathcal{C}_i} \omega(\hat{m}_j^p, m_i)\, m_i}{Z_j + \sum_{m_i \in \mathcal{C}_i} \omega(\hat{m}_j^p, m_i)}. \end{aligned} \tag{16}$$

To further simplify, we invoke Eq. (13) and assume $\hat{m}_i^l \approx \hat{m}_j^p$, i.e. the two merged tokens form a small angle in feature space. Under assumptions above, the weight term can be approximated as follows.

Proof. Assume $\hat{m}_j \approx \hat{m}_i$ and $m_i \approx \hat{m}_i$ (omitting superscripts for clarity). We aim to show

$$\omega(\hat{m}_j, m_i) \approx e^{-1} \omega(\hat{m}_j, \hat{m}_i)\, \omega(\hat{m}_i, m_i). \tag{17}$$

Let the normalized key vector of token m_i be $n_i = \frac{k_i}{\|k_i\|}$, so the cosine similarities in Eq. (2) can be rewritten as:

$$\begin{aligned} \cos_k(\hat{m}_j, \hat{m}_i) &= \hat{n}_j \cdot \hat{n}_i, \\ \cos_k(m_i, \hat{m}_i) &= n_i \cdot \hat{n}_i. \end{aligned}$$

Define the small deviations

$$\eta = \hat{n}_j - \hat{n}_i \approx 0, \quad \delta = n_i - \hat{n}_i \approx 0.$$

Then

$$\begin{aligned} \omega(\hat{m}_j, m_i) &= \exp\left(\cos_k(\hat{m}_j, m_i)\right) \\ &= \exp\left(\hat{n}_j \cdot n_i\right) \\ &= \exp\left(\hat{n}_j \cdot (\hat{n}_i + \delta)\right) \\ &= \exp\left(\hat{n}_j \cdot \hat{n}_i + \delta \cdot (\hat{n}_i + \eta)\right). \end{aligned} \tag{18}$$

Neglecting the second-order term $\delta \cdot \eta$ and using $\delta \cdot \hat{n}_i = n_i \cdot \hat{n}_i - 1$, Eq. (18) gives

$$\begin{aligned} \omega(\hat{m}_j, m_i) &\approx \exp\left(\hat{n}_j \cdot \hat{n}_i + n_i \cdot \hat{n}_i - 1\right) \\ &= e^{-1} \omega(\hat{m}_j, \hat{m}_i)\, \omega(\hat{m}_i, m_i), \end{aligned}$$

which proves Eq. (17). □

Thus, the final re-merged token can be written as

$$\begin{aligned} \hat{m}^* &\approx \frac{Z_j \hat{m}_j^p + e^{-1} \omega(\hat{m}_j^p, \hat{m}_i^l) \sum_{m_i \in \mathcal{C}_i} \omega(\hat{m}_i^l, m_i)\, m_i}{Z_j + e^{-1} \omega(\hat{m}_j^p, \hat{m}_i^l) \sum_{m_i \in \mathcal{C}_i} \omega(\hat{m}_i^l, m_i)} \\ &= \frac{Z_j \hat{m}_j^p + e^{-1} \omega(\hat{m}_j^p, \hat{m}_i^l)\, Z_i \hat{m}_i^l}{Z_j + e^{-1} \omega(\hat{m}_j^p, \hat{m}_i^l)\, Z_i}. \end{aligned} \tag{19}$$

C. More Experiments

C.1. Additional Qualitative Visualizations

We provide additional qualitative comparisons with representative baselines. Figure 5 presents side-by-side reconstructions on challenging scenes. Under the same runtime memory budget, our method preserves finer structures and yields more temporally consistent reconstructions.

C.2. Video Depth Evaluation

We conduct video depth estimation by evaluating both per-frame depth quality and inter-frame consistency, aligning predicted depth maps to ground truth using a per-sequence scale following [12, 14]. Experiments are conducted on three benchmark datasets—Sintel [2], Bonn [7],

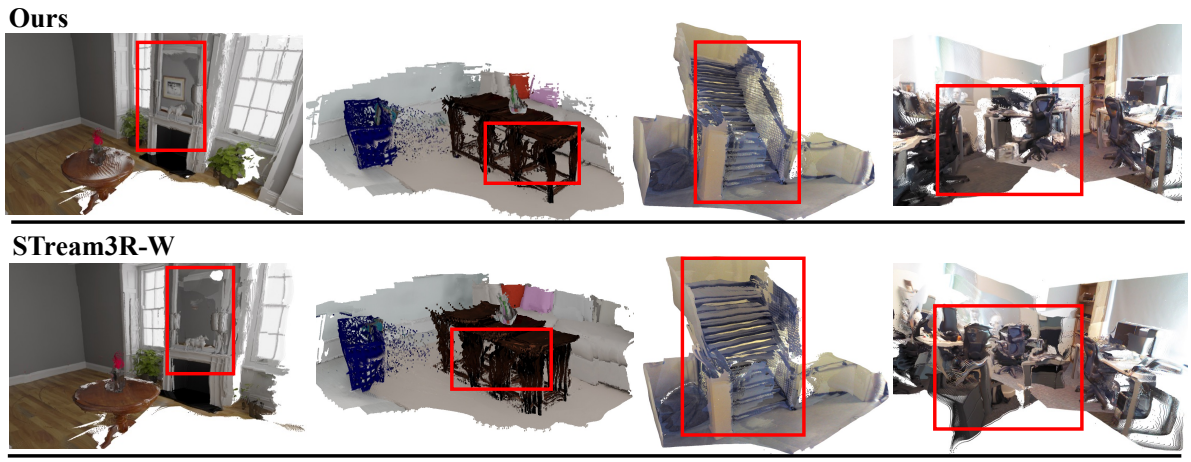

Figure 5. Additional qualitative comparisons with baselines.

Table 1. Comparison of scale-invariant depth estimation results (using per-sequence alignment) on the Sintel [2], Bonn [7], and KITTI [3] datasets. We evaluate accuracy (Abs Rel) and consistency ($\delta < 1.25$) across different categories of methods. STream3R-W and StreamVGGT-W serve as sliding-window attention baselines, while our STream3R-STAC and StreamVGGT-STAC apply the cache compression strategy under the same runtime memory budget. Methods that rely on global alignment are marked as "GA". The "Type" column indicates whether the method is optimization-based ("Optim"), streaming ("Online"), or offline ("Offline").

Method	Type	Sintel		Bonn		KITTI	
		Abs Rel ↓	$\delta < 1.25$ ↑	Abs Rel ↓	$\delta < 1.25$ ↑	Abs Rel ↓	$\delta < 1.25$ ↑
DUST3R-GA [13]	Optim	0.656	45.2	0.155	83.3	0.144	81.3
MASt3R-GA [6]	Optim	0.641	43.9	0.252	70.1	0.183	74.5
MonST3R-GA [15]	Optim	0.378	55.8	0.067	96.3	0.168	74.4
CUT3R [12]	Online	0.421	47.9	0.078	93.7	0.118	88.1
Spann3R [10]	Online	0.622	42.6	0.144	81.3	0.198	73.7
Point3R [14]	Online	0.452	48.9	0.060	96.0	0.136	84.2
VGGT [11]	Offline	0.297	68.8	0.057	96.8	0.061	96.8
STream3R [5]	Online	0.264	**70.5**	0.070	95.2	0.080	94.7
STream3R-W8	Online	0.277	68.9	**0.066**	**96.5**	0.086	94.8
STream3R-STAC	Online	**0.259**	69.6	0.071	94.4	**0.075**	**95.7**
StreamVGGT [16]	Online	**0.323**	**65.7**	**0.059**	**97.2**	**0.173**	**72.2**
StreamVGGT-W8	Online	0.349	62.8	0.068	96.4	0.201	67.3
StreamVGGT-STAC	Online	0.327	65.2	0.060	97.1	0.192	67.6

and KITTI [3]—covering dynamic indoor and outdoor scenarios.

Results are summarized in Table 1. STAC provides a *training-free* mechanism for managing the KV cache under streaming input in Causal-VGGT backbones, including STream3R [5] and StreamVGGT [16]. Under the same runtime memory budget, STAC consistently outperforms the sliding-window attention baselines across most datasets. Moreover, despite requiring no architectural changes or task-specific fine-tuning, it achieves accuracy comparable to full-cache Causal-VGGT models [5, 16] and remains robust across diverse scenes.

Furthermore, STAC preserves the representational power of the Causal-VGGT backbone, enabling it to surpass streaming-based methods such as CUT3R [12], Spann3R [10], and Point3R [14], as well as optimization-based approaches like DUST3R-GA [13], MASt3R-GA [6], and MonST3R-GA [15], on most benchmarks. Notably, on Sintel, equipping STream3R with STAC outperforms the offline full-attention VGGT [11], validating the effectiveness of cache compression without sacrificing model capacity.

C.3. Behavior of STAC for Long Video Streams

We analyze the behavior of STAC on a long indoor stream to verify its scalability under prolonged causal inference. As shown in Figure 6, the left panel reports the per-frame backbone inference time with a breakdown of the main components (attention+FFN, retrieval, and merging), while the right panel shows the evolution of KV-cache memory usage

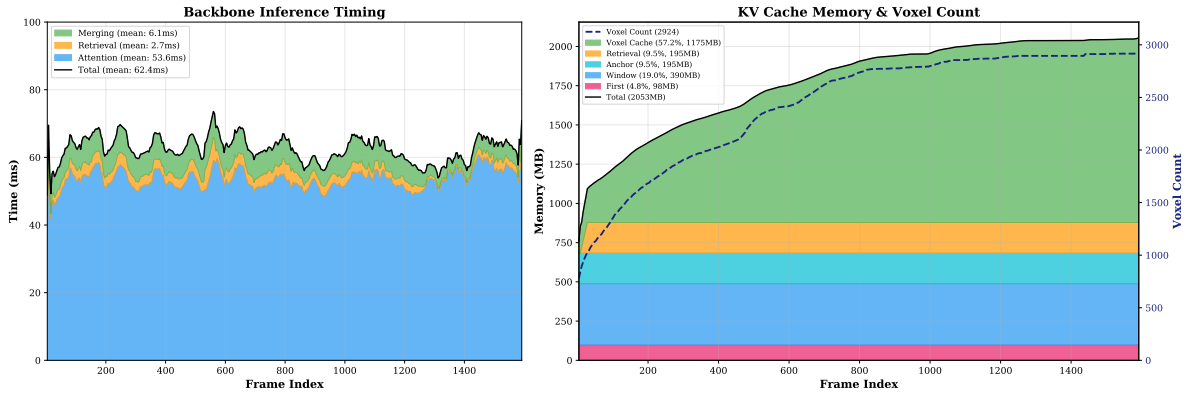

Figure 6. Behavior of STAC on a long indoor stream from the *whiteroom* scene of the NRGBD dataset [1]. **Left:** backbone inference time per frame with a breakdown of major components. **Right:** KV-cache memory usage and the number of active voxels (dashed, right axis) over the stream.

Table 2. Effect of voxel resolution r and merging threshold λ on reconstruction quality, memory, and runtime. Experiments are conducted on the NRGBD [1] dataset using Stream3R [5] with stride 5. Unless otherwise specified, STAC uses the default setting r=0.05 and λ=0.8.

r	λ	ACC ↓	Comp ↓	NC ↑	Mem(GB) ↓	Time(ms) ↓
0.025	0.8	0.064	0.014	0.695	4.861	78.85
0.10	0.8	0.065	0.015	0.698	1.119	73.52
0.05	0.6	0.065	0.015	0.696	1.755	68.78
0.05	0.9	0.064	0.014	0.700	2.251	74.70
0.05	0.8	0.065	0.014	0.700	2.210	71.18

together with the number of voxels maintained by the long-term spatial cache. Across the stream, the inference time remains stable (typically 70–90 ms), indicating that retrieval and cache updates introduce a bounded overhead without accumulating latency as the sequence grows. Meanwhile, the KV-cache memory increases only in the early stage and then gradually plateaus, and the active-voxel count converges to a stable range. These trends suggest that STAC effectively reuses and compresses revisited spatial evidence, preventing the unbounded cache growth that would otherwise occur with full KV caching.

C.4. Ablation of Hyperparameters

We study two key hyperparameters in STAC: the voxel grid resolution r used to discretize 3D space and the merging threshold λ that controls how aggressively evicted tokens are fused into long-term voxel representatives. Together, they determine the granularity of spatial grouping and the strength of compression, and thus trade off reconstruction quality, memory usage, and retrieval efficiency. As shown in Table 2, r=0.05 and λ=0.8 provide a robust balance across these factors.

Specifically, decreasing r yields a finer voxelization and activates more voxels along the stream, which increases cache capacity demand and enlarges the neighborhood set queried during retrieval, leading to higher memory and runtime costs with only marginal accuracy gains. Conversely, an overly large r groups geometrically dissimilar observations into the same voxel, forcing heterogeneous tokens to share representatives and causing information loss. For the merging threshold, a smaller λ triggers more frequent one-to-one merges, which may over-compress and degrade reconstruction quality, while a larger λ reduces direct merging and shifts more tokens into buffered aggregation, increasing long-term cache growth pressure and retrieval overhead.

Table 3. Ablation study on 3D reconstruction performance using the NRGBD and 7-Scenes [8] datasets. Metrics include Accuracy (ACC), Completion (Comp), and Normal Consistency (NC). The baseline uses a sliding-window attention mechanism. "Random Selection" selects tokens without relevance estimation; "Uniform Merging" merges tokens uniformly; and "Ours" applies attention-guided selection and weighted merging.

Policy	NRGBD			7-Scenes		
	ACC ↓	Comp ↓	NC ↑	ACC ↓	Comp ↓	NC ↑
Baseline	0.078	0.015	0.687	0.107	0.035	0.587
Random Selection	0.076	0.021	0.696	0.101	0.039	0.591
Uniform Merging	0.065	0.015	0.699	0.047	0.025	0.606
Ours(STAC)	0.065	0.014	0.700	0.047	0.024	0.606

C.5. Ablation of Compression Strategies

To assess the effectiveness of STAC, we conduct an ablation study within STream3R [5] on NRGBD [1] and 7-Scenes [8] for 3D reconstruction. Table 3 compares different cache compression strategies.

- *Selection Strategy.* We compare token selection policies in the *Working Temporal Token Cache*. Our attention-guided policy retains anchor tokens and evicts less relevant ones for merging, whereas random selection chooses tokens indiscriminately. The attention-guided strategy better preserves temporally persistent evidence under lim-

ited memory and outperforms random selection.

- *Merging Strategy.* We analyze token fusion in the *Long-term Spatial Token Cache*. Uniform averaging treats all tokens equally and ignores feature relevance. In contrast, our weighted merging better preserves informative components during fusion and yields consistently better reconstruction quality across both datasets. Notably, since tokens are grouped by voxel regions, the spatial partition itself already introduces a degree of implicit discrimination, which partially explains why uniform merging achieves relatively competitive performance.

References

[1] Dejan Azinović, Ricardo Martin-Brualla, Dan B Goldman, Matthias Nießner, and Justus Thies. Neural rgb-d surface reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6290–6301, 2022. 6

[2] Daniel J Butler, Jonas Wulff, Garrett B Stanley, and Michael J Black. A naturalistic open source movie for optical flow evaluation. In *European Conference on Computer Vision*, pages 611–625, 2012. 4, 5

[3] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The international journal of robotics research*, 32(11):1231–1237, 2013. 5

[4] Xin Jin and Jiawei Han. *K-Means Clustering*, pages 563–564. Boston, MA, 2010. 1

[5] 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. 1, 5, 6

[6] Vincent Leroy, Yohann Cabon, and Jérôme Revaud. Grounding image matching in 3d with mast3r. In *European Conference on Computer Vision*, pages 71–91, 2024. 5

[7] Emanuele Palazzolo, Jens Behley, Philipp Lottes, Philippe Giguere, and Cyrill Stachniss. Refusion: 3d reconstruction in dynamic environments for rgb-d cameras exploiting residuals. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7855–7862. IEEE, 2019. 4, 5

[8] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–580. IEEE, 2012. 6

[9] Zhongwei Wan, Xinjian Wu, Yu Zhang, Yi Xin, Chaofan Tao, Zhihong Zhu, Xin Wang, Siqi Luo, Jing Xiong, Longyue Wang, et al. D2o: Dynamic discriminative operations for efficient long-context inference of large language models. *arXiv preprint arXiv:2406.13035*, 2024. 3

[10] Hengyi Wang and Lourdes Agapito. 3d reconstruction with spatial memory. *arXiv preprint arXiv:2408.16061*, 2024. 5

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

[12] Qianqian Wang, Yifei Zhang, Aleksander Holynski, Alexei A Efros, and Angjoo Kanazawa. Continuous 3d perception model with persistent state. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 10510–10522, 2025. 4, 5

[13] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20697–20709, 2024. 5

[14] Yuqi Wu, Wenzhao Zheng, Jie Zhou, and Jiwen Lu. Point3r: Streaming 3d reconstruction with explicit spatial pointer memory. *arXiv preprint arXiv:2507.02863*, 2025. 4, 5

[15] 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*, 2024. 5

[16] Dong Zhuo, Wenzhao Zheng, Jiahe Guo, Yuqi Wu, Jie Zhou, and Jiwen Lu. Streaming 4d visual geometry transformer. *arXiv preprint arXiv:2507.11539*, 2025. 1, 5