HTTM: Head-wise Temporal Token Merging for Faster VGGT

Weitian Wang[1,2], Lukas Meiner[1], Rai Shubham[1], Cecilia De La Parra[1], Akash Kumar[2*]
[1]Robert Bosch GmbH, Renningen, Germany, [2]Ruhr University Bochum, Bochum, Germany
{weitian.wang, shubham.rai, cecilia.delaparra}@bosch.com, akash.kumar@ruhr-uni-bochum.de

Abstract

The Visual Geometry Grounded Transformer (VGGT) marks a significant leap forward in 3D scene reconstruction, as it is the first model that directly infers all key 3D attributes (camera poses, depths, and dense geometry) jointly in one pass. However, this joint inference mechanism requires global attention layers that perform all-to-all attention computation on tokens from all views. For reconstruction of large scenes with long-sequence inputs, this causes a significant latency bottleneck. In this paper, we propose head-wise temporal merging (HTTM), a training-free 3D token merging method for accelerating VGGT. Existing merging techniques merge tokens uniformly across different attention heads, resulting in identical tokens in the layers' output, which hinders the model's representational ability. HTTM tackles this problem by merging tokens in multi-head granularity, which preserves the uniqueness of feature tokens after head concatenation. Additionally, this enables HTTM to leverage the spatial locality and temporal correspondence observed at the head level to achieve higher merging ratios with lower merging costs compared to existing methods. Thus, HTTM achieves up to 7× acceleration with negligible performance drops in a GPU-based inference.

1. Introduction

The Visual Geometry Grounded Transformer (VGGT) [25] is a recently proposed feed-forward transformer model that directly infers all key 3D attributes of a scene from a variable number of views. By this direct inference, VGGT can outperform state-of-the-art methods while avoiding costly visual-geometry post-processing methods, marking an important breakthrough in 3D computer vision.

One of the key designs of VGGT is its alternating frame-wise and global attention. In the global attention layers, all tokens from different views participate in the multi-head attention computation. In practice, this approach results in extremely long token sequences (more than 20k tokens) even for small scenes. Thus, the global attention layers become the main latency bottleneck of VGGT, limiting its efficiency in medium and large scene reconstruction.

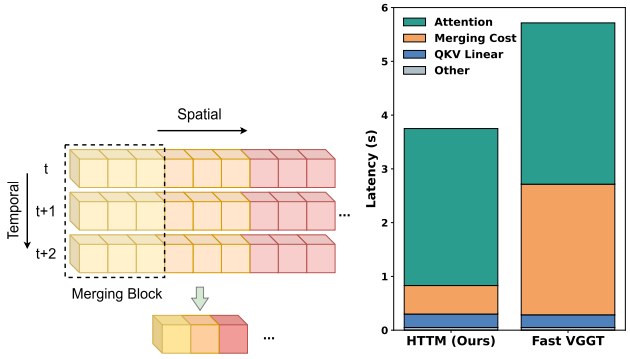


Figure 1. HTTM forms spatio-temporal merging blocks that jointly consider neighboring tokens across consecutive frames. This design exploits temporal coherence and spatial redundancy to merge tokens efficiently. With the same merging ratio, HTTM reduces the merging cost by **4.58**×

.

Motivated by the development of long-context large language models (LLMs) and vision language models (VLMs), many methods [7, 15, 23, 31, 32] have been proposed to ease the high computational costs of long-sequence attention layers. These methods are mainly sparsity-based approaches, aiming to exploit that attention scores in LLMs and VLMs tend to concentrate on a small set of tokens. However, the sparsity level of VGGT's global attention layers is lower compared to the attention layer in LLM, as shown in Fig. 2, which limits the latency improvement of these methods when applied to VGGT.

On the other hand, the narrow attention distribution patterns that appear in VGGT favor similarity-based methods, as the attention weight distributions do not change drastically from token to token. ToMe [3] gave rise to a variety of token merging approaches that accelerate transformers by merging redundant tokens based on feature similarity. Fol-

*Corresponding author

lowing this work, a number of extensions [2, 4, 14, 20, 30] have been proposed, demonstrating that similarity-based token reduction can effectively improve efficiency without retraining.

While several approaches [2, 20] are designed for, or can be adapted to VGGT, they fail to recognize the special head-level similarity pattern of VGGT, and exhibit high merging overhead on long input sequences.

To this end, we propose HTTM (Head-wise Temporal Token Merging), a training-free token merging approach tailored for VGGT's Global Attention layers. HTTM addresses the limitations of existing methods through three key innovations: (1) head-wise merging that allows each attention head to merge tokens independently, preserving head-specific information and avoiding feature collapse after concatenation; (2) block-wise token merging that reduces the token matching cost compared to global matching strategies. (3) temporal reordering that reorganizes tokens into spatio-temporal blocks to exploit both spatial redundancy and temporal coherence, improving merging quality in fixed merging block size; (4) head-wise adaptive outlier filtering that filters outliers across all heads under a global budget. These designs enable HTTM to achieve notable speedups while maintaining high reconstruction quality.

Our main contributions are summarized as follows:

- We conduct systematic explorations of token merging in VGGT, revealing distinct similarity patterns along both spatial and temporal dimensions. Through extensive analysis, we show that temporal correlations exhibit higher redundancy, motivating the need for temporally aware merging strategies.
- Recognizing the main computational overhead in existing token merging methods, we analyze the trade-off between merging cost and merging quality, and propose a block-wise merging strategy with temporal reordering that largely reduces merging cost while preserving merging quality by aligning spatially and temporally correlated tokens.
- We introduce an adaptive outlier filtering mechanism that filters outliers across all attention heads under a global budget, allocating more budget to heads with higher outlier density and improving overall quality with minimal overhead.

2. Related Work

2.1. Efficient Attention for Long Sequences

To address the quadratic complexity of attention layers in transform-based models over long sequences, sparse attention methods have recently gained popularity. Utilizing the inherently sparse attention mechanism in LLMs, methods like Sparse Transformer [7] and BigBird [32] accelerate inference and extend maximum sequence lengths through

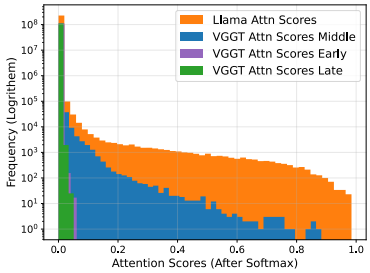


Figure 2. Attention score distribution comparison between VGGT and Llama 3.1 8B[12]. The distribution of attention scores in VGGT is heavily concentrated around low values in both its early and late layers. In the middle layers, attention distribution is still more skewed towards lower values compared to Llama.

local or block-based attention but often require retraining. StreamingLLM [18] enables LLMs to handle unlimited texts without fine-tuning by retaining the attention sinks (initial tokens) and the recent tokens. SparseVLM [33] accelerates VLMs through a text-guided training-free token pruning mechanism. For vision transformers, SparseViT [5] and MixA-Q [29] discard or compress unimportant tokens with small magnitudes for latency improvements. However, these methods are all dependent on the sparse attention pattern, which VGGT doesn't exhibit distinctly as shown in Fig. 2.

2.2. Accelerating Visual Geometry Models

Feed-forward models for 3D reconstruction tasks like DUSt3R [28], MASt3R [17], and VGGT [25] have emerged as strong alternatives to traditional multi-view reconstruction pipelines. One of the key designs of VGGT is its alternating global attention and frame attention layer. In frame attention layers, tokens interact within each frame independently. In global attention layers, tokens interact globally across all frames, helping the model build global 3D correspondence. However, as input sequences grow longer, VGGT's global attention becomes a computational bottleneck [20, 24].

Online methods like CUST3R [27] and TTT3R [6] maintain a global state that is updated for each new input frame, thus avoiding the all-to-all attention. Nevertheless, as reported in TTT3R Chen et al. [6], their performance is still not comparable to offline methods like VGGT.

While a few approaches [20, 24] try to accelerate VGGT by applying established methods for ViT or LLM token sequence compression, their analysis of VGGT's specific redundancies and resulting opportunities for compression is limited. FastVGGT [20] proposes to use the token merging from ToMeSD [2] to accelerate the global attention layers. Although it achieves high latency improvements, FastVGGT fails to adapt the merging method from the 2D vision tasks to the similarity patterns of VGGT, limiting its

performance. Block-sparse global attention [24] exploits the structure of attention matrices found in VGGT's middle layers. However, as shown in Fig 2, the sparsity level of VGGT is much lower compared to the LLMs, limiting the latency improvements it can achieve without hurting the performance.

2.3. Token Merging Methods

ToMe [3] pioneered token merging to reduce the number of tokens for ViTs, employing bipartite soft matching of tokens based on key similarity averaged across attention heads. Extensions such as ToMeSD [2] enable token merging for diffusion models by introducing unmerging operations to restore dense token sequences, while ToFu [14] allows for both pruning and merging of tokens. Token merging has also been extended to video domains [11, 13, 16]. These approaches explore the use of spatio-temporal merging, leveraging the redundancy across video frames to achieve better merging strategies. HTTM shares the same underlying intuition with these methods. However, given that token merging and unmerging need to be applied to every layer in VGGT, our method specifically focuses on limiting the cost of token similarity computations instead of devising a more sophisticated strategy for better merging quality.

3. Head-wise Temporal Token Merging

3.1. Observations and Insights

Before going into details of our head-wise temporal token merging method, we first present the observations and intuitions that led us to this particular approach. Unless otherwise stated, all the similarity matrices or similarity patterns shown in this section are self-cosine-similarity matrices over a sequence of tokens.

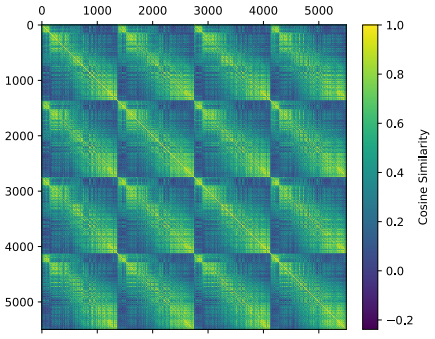


Figure 3. Cosine similarity patterns averaged across all heads between query tokens of 4 adjacent frames. High similarities observed along the block diagonals indicate that tokens within the same spatial region (local areas) and corresponding locations across consecutive frames share highly similar features.

Spatial & Temporal Token Similarity Fig. 3 illustrates the cosine similarity patterns between all the query tokens in the first 4 frames of a scene from the NRGBD [1] dataset. It can be observed that high similarity scores concentrate near the main and off-diagonals. The high similarity scores near the main diagonal indicate that each token mainly attends to similar tokens in its local neighborhood. On the other hand, the high similarity scores near the off-diagonals have offsets of frame length, meaning that the same area of different frames also shows high similarity. We will explore the reason behind these patterns in the following paragraphs.

RoPE Effect The strong periodic patterns in Fig. 3 emerge from the way the Rotary Position Embedding (RoPE) [22] is applied at each attention layer of VGGT [25]. Unlike models such as BERT [10] or Stable Diffusion [19], which use a fixed positional embedding added only once at the input, VGGT reapplies RoPE at every layer, which strengthens positional encoding effects throughout the network. In the global attention layers, where tokens from all views interact, RoPE differentiates between individual frames, enhancing spatial distinctiveness and reducing similarity between distant locations. In the frame attention layers, RoPE is applied identically to each frame, so corresponding regions across frames share similar positional encodings, inducing temporal coherence between the same spatial areas of adjacent frames. More discussion on this can be found in the Appendix.

Input Similarity Effect Although RoPE induces a periodic similarity pattern, intra- and inter-frame similarities also contribute to this observation. As shown in Fig. 4, we perform single-frame reconstruction on two images with different levels of visual redundancy in pixel patches. At a deep Global Attention layer (14th), the query tokens from the high-redundancy input frame (a wall) show much stronger spatial similarity than the low-redundancy input frame with cluttered objects.

In Fig. 5, we visualize the query tokens' similarity of a deep layer (14th) across 8 consecutive frames from three 30-frame reconstructions under varying degrees of visual continuity. When the input frames are temporally continuous and highly similar, strong off-diagonal responses appear, indicating that tokens corresponding to the same spatial regions in adjacent frames exhibit high similarity. As the overlap decreases, these off-diagonal structures become weaker and more dispersed, reflecting diminished temporal correspondence between tokens.

Summary These findings suggest that the observed similarity pattern originates from two intertwined factors: the architectural effect of RoPE, which enforces spatial distinctness in global attention layers and temporal correspon-

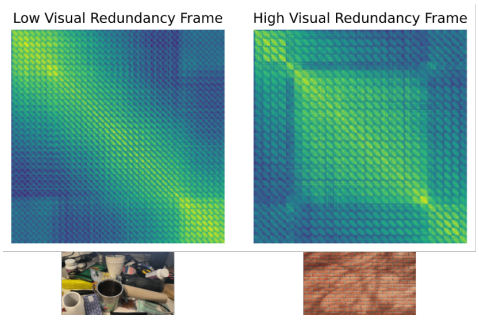


Figure 4. Cosine similarity between query tokens at the 14th deep global attention layer in single-frame reconstruction. The high visual redundancy frame (a wall) shows stronger spatial similarity compared to the low visual redundancy frame (cluttered objects).

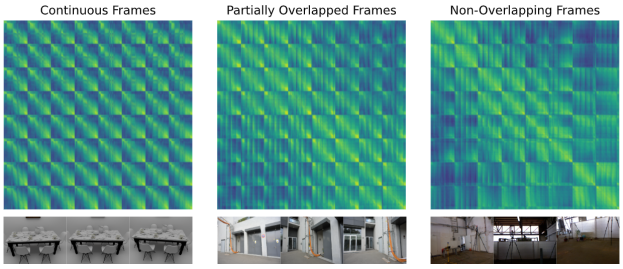


Figure 5. Cosine similarity between query tokens of 8 adjacent frames at the 14th global attention layer. High similarity between input frames leads to high temporal similarity of query tokens, as shown by the high scores on off-diagonals.

dence in frame attention layers, and the input-level similarity, which propagates through the network and reinforces correlations between spatially corresponding regions. Together, they shape the distinctive spatio–temporal similarity structure of tokens in VGGT. Hence, we devise a merging strategy that can jointly consider the spatial locality and temporal correspondence.

3.2. Head-wise Token Merging

Existing token merging methods [2, 3, 20] typically adopt a uniform merging strategy for all the heads. While this strategy simplifies the merging process, enforcing a shared merging pattern across all heads results in repetitive token representations after unmerging, as illustrated in Fig. 7, thereby limiting the model's representational diversity. To address this issue, we use a head-wise token merging strategy in HTTM, in which each head performs merging independently according to its own similarity patterns. This head-specific merging enables different heads to combine tokens in distinct ways, producing more diverse and complementary representations after concatenation.

However, the head-wise token merging introduces an overhead that increases proportionally to the number of heads, which is the reason why existing methods avoid this strategy. In Sec. 3.3, we present our approach to lower the merging overhead by using a temporal merging block.

We now describe the head-wise token merging procedure in HTTM. As shown in Fig. 6, for each head of a multi-head attention layer, its query and key tokens are merged independently using a merge module. Value tokens are merged following the key tokens, as the attention computation requires key-value consistency.

Formally, given a sequence of N input tokens $\mathbf{X} \in \mathbb{R}^{N\times d}$ with embedding dimension d, we apply standard multi-head attention projections to obtain $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{h\times N\times d_{\text{head}}}$, where h is the number of attention heads and $d_{\text{head}} = d/h$ is the embedding dimension per head. For each attention head $i \in \{1, \dots, h\}$, we then denote its head embeddings $\mathbf{Q}^{(i)}, \mathbf{K}^{(i)}, \mathbf{V}^{(i)} \in \mathbb{R}^{N\times d_{\text{head}}}$.

Merging Our head-wise temporal merging can be understood as a set of functions $\mathcal{M}_i : \mathbb{R}^{N\times d_{\text{head}}} \to \mathbb{R}^{M\times d_{\text{head}}}$ that reduce the head-wise token sequence length from N to M, independently for each head. Following ToMeSD [2], the tokens are partitioned into disjoint sets of source (`src`) tokens $\mathbf{S}^{(i)} \in \mathbb{R}^{N_s\times d_{\text{head}}}$ and destination (`dst`) tokens $\mathbf{D}^{(i)} \in \mathbb{R}^{N_d\times d_{\text{head}}}$. Then, we compute the cosine similarity matrix $\mathbf{Sim}^{(i)} \in \mathbb{R}^{N_s\times N_d}$ between $\mathbf{S}^{(i)}$ and $\mathbf{D}^{(i)}$:

$$\mathbf{Sim}^{(i)} = \text{RowNorm}\left(\mathbf{S}^{(i)}\right) \cdot \text{RowNorm}\left(\mathbf{D}^{(i)}\right)^{\top} \quad (1)$$

For each `src` token, we only keep the similarity score to its most similar `dst` token, which is considered its best match. Let $r = N - M$, we merge the top-r `src` tokens with the highest similarity scores into their best-matching `dst` token. Then, we compute r *merged tokens* as the mean of all of their *constituent tokens* (the matched `dst` tokens and all the `src` tokens that match it).

This merging process is performed separately for the head-wise queries and keys to obtain the merged tokens:

$$\tilde{\mathbf{Q}}^{(i)} = \mathcal{M}_i^q(\mathbf{Q}^{(i)}), \quad \tilde{\mathbf{K}}^{(i)} = \mathcal{M}_i^k(\mathbf{K}^{(i)}). \quad (2)$$

The values $\mathbf{V}^{(i)}$ are merged using the same matches computed in $\mathcal{M}_i^k(\mathbf{K}^{(i)})$, from which we then obtain $\tilde{\mathbf{V}}^{(i)}$. We can now efficiently perform the attention computation on the reduced-length $\tilde{\mathbf{Q}}^{(i)}/\tilde{\mathbf{K}}^{(i)}/\tilde{\mathbf{V}}^{(i)}$ tokens:

$$\mathbf{A}^{(i)} = \text{softmax}\left(\frac{\tilde{\mathbf{Q}}^{(i)}(\tilde{\mathbf{K}}^{(i)})^{\top}}{\sqrt{d_{\text{head}}}}\right) \in \mathbb{R}^{M\times M}, \quad (3)$$

$$\tilde{\mathbf{O}}^{(i)} = \mathbf{A}^{(i)}\tilde{\mathbf{V}}^{(i)} \in \mathbb{R}^{M\times d_{\text{head}}}. \quad (4)$$

Unmerging Finally, we need to restore the full token sequence through a head-wise unmerging step, formalized as

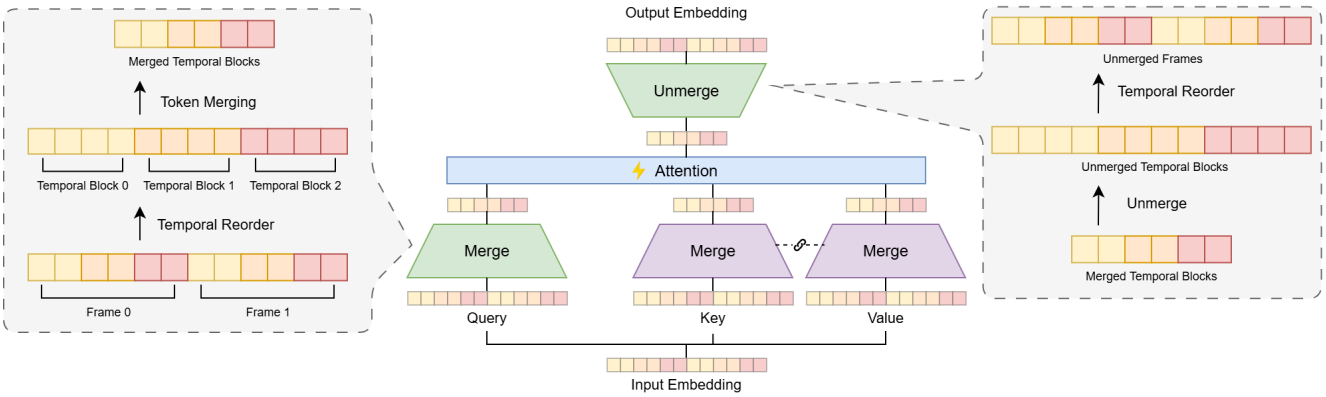


Figure 6. Overview of how HTTM accelerates attention layers by merging QKV tokens. HTTM merges and unmerges Q/K/V tokens before and after entering the attention kernel. Using temporal reordering 3.3, HTTM forms temporal blocks that consist of similar tokens (denoted with colors) and performs merging and unmerging within these blocks.

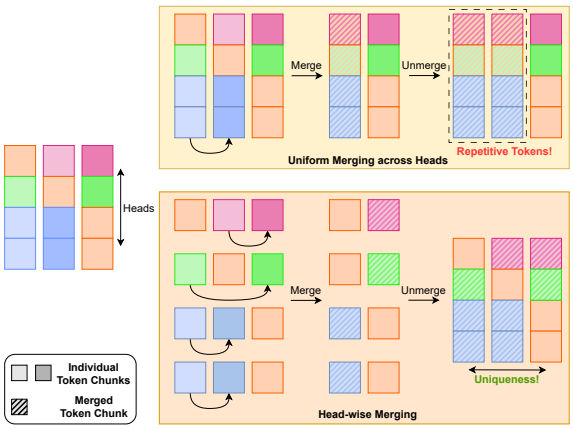


Figure 7. Head-wise merging can better keep the uniqueness of the output embedding. Different shades of the same color represent similar token chunks.

a set of functions $\mathcal{U}_i : \mathbb{R}^{M\times d_{\text{head}}} \to \mathbb{R}^{N\times d_{\text{head}}}$. Here, we utilize the simple yet effective unmerging procedure proposed in ToMeSD [2]: For each query token $\mathbf{q}_n^{(i)} \in \mathbf{Q}^{(i)}$ from the original query sequence that contributed to a merged query token $\tilde{\mathbf{q}}_m^{(i)} \in \tilde{\mathbf{Q}}^{(i)}$, its final output $\mathbf{o}_n^{(i)}$ is simply the copied output of the merged token $\tilde{\mathbf{o}}_m^{(i)}$:

$$\mathbf{o}_n^{(i)} := \tilde{\mathbf{o}}_m^{(i)} . \tag{5}$$

Tokens that were not merged retain their output. As a final step, we concatenate the unmerged outputs across all attention heads on the channel dimension to obtain:

$$\mathbf{O}^{(i)} = \mathcal{U}_i(\tilde{\mathbf{O}}^{(i)}) \in \mathbb{R}^{N\times d_{\text{head}}} , \tag{6}$$

$$\mathbf{O} = \text{Concat}(\mathbf{O}^{(1)}, \ldots, \mathbf{O}^{(h)}) \in \mathbb{R}^{N\times d} . \tag{7}$$

Note that while the queries and keys can make independent merging decisions per head, the unmerging is determined solely by the matches in $\mathcal{M}_i^q(\mathbf{Q}^{(i)})$, because the order of $\tilde{\mathbf{O}}^{(i)}$ is determined by $\tilde{\mathbf{Q}}^{(i)}$.

3.3. Temporal Reordering and Merging

As stated in Sec. 3.2, the main computational bottleneck of token merging is the computation of the similarity matrix in Eq. 1. To merge N tokens of dimension d_{head}, if we separate them into 25% `dst` tokens and 75% `src` tokens, the computation of the similarity matrix between them would require around $0.19N^2 d_{\text{head}}$ FLOPs, increasing quadratically with the number of tokens.

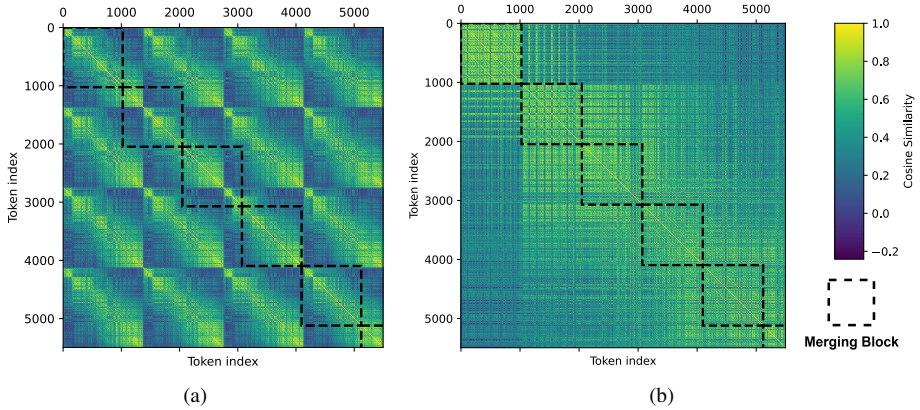
To address this challenge, we propose to split the whole token sequence into *merging blocks* of *fixed* size n_b and only perform the token merging within these blocks. In this way, the merging cost grows linearly with N. As stated in Sec. 3.2, we merge the top-r `src` based on the exact similarity matrix. This implies three key points[1] to formulate a good block-wise token merging strategy:

- The block similarity matrix computed between tokens in the merging block is composed of similarity scores from the global all-to-all similarity matrix.
- The quality of the block-wise token merging method is dependent on the number of high similarity scores that we can include in the block similarity matrix.
- Given a splitting strategy, larger merging block sizes are always better at the expense of higher merging costs.

Hence, the challenge is how to split the token sequence into merging blocks of a given size to maximize the number of high similarity scores in the block similarity matrix.

Given our findings in Sec. 3.1, highly similar tokens reside in neighborhoods, implying that - given the query tokens of a head $\mathbf{Q}^{(i)} \in \mathbb{R}^{N\times d_{\text{head}}}$ - if we merge these tokens in consecutive merging blocks along the N dimension, we can capture the spatial similarity along the main diagonal of

[1] A proof of these statements can be found in the Appendix.



(a) (b)

Figure 8. Token similarity in merging blocks of size 1024. (a) Without temporal reordering, many highly similar matches lie outside of merging blocks. We can't capture those matches unless we use a global merging block that is very costly. (b) Through temporal reordering, high-similarity matches shift inside merging blocks, leading to better merging quality.

the similarity matrix. The negative implication is that we will lose track of many highly similar matches outside the merging block. In Fig. 8a, we visualize the local similarity matrix of merging blocks of size 1024 on the global similarity matrix of 4 frames with 1374 tokens each. It can be observed that a lot of high similarity scores fall outside of the merging blocks' similarity matrices, making them unconsidered in the merging process. Intuitively, this is because the spatially expanding merging block fails to capture the temporal correspondence we observed in Sec 3.1.

To solve this problem, we propose a simple but effective technique: *Temporal Reordering*. As shown in Fig. 6, before the token merging process, we reorder the tokens so that similar spatial blocks of size n_s (marked with the same color) across n_t frames are stacked together to form temporal merging blocks of size $n_b = n_s \times n_t$ that consist of highly similar tokens. Then, we can perform efficient block-wise token merging in these temporal merging blocks of size $n_b << N$ and reduce the token length of Q/K/V, significantly accelerating the attention computation. After the attention computation, we first unmerge the tokens in temporal blocks to recover their original sizes, then we reorder the tokens to make sure that the order of output tokens aligns with the input tokens.

In Fig. 8b, we perform a temporal reordering where we stack spatial blocks of size $n_s = 128$ from $n_t = 8$ frames. Compared to Fig. 8a, more high similarity scores are included in the merging blocks after the temporal reordering, meaning that more highly-similar token matches can be considered by the merging process, leading to better overall merging quality. The criteria to choose the appropriate size of spatial blocks and number of temporal frames for the reordering and merging are discussed in Sec. 4.5.

3.4. Adaptive Outlier Filtering

For better parallelization, we use a fixed spatial block size and number of temporal frames when forming the merging blocks across heads, which does not always align with the similarity pattern across frames and heads. Furthermore, we use the same merging ratio for all these merging blocks. Hence, for merging blocks with low intra-block similarity, low-similarity tokens will be merged, resulting in large distances between the original tokens and the merged token. We call these tokens *outliers*. We filter outliers adaptively to improve the overall representational ability of the merged tokens:

1. We perform an initial merging step of query tokens inside the block as described in Sec. 3.2.
2. Given the merged queries $\tilde{\mathbf{Q}} \in \mathbb{R}^{h \times M \times d_{head}}$ for all heads, we compute the deviation from all the original query tokens to their corresponding merged tokens in L_2 distances. For the tokens that are not merged with other tokens, the deviation is set to zero.
3. We then identify the top $d\%$ of tokens with the largest deviations to their merged tokens across *all* heads and mark them as outliers, yielding a binary outlier mask $\mathbf{M}_o \in \{0,1\}^{h \times N}$.
4. We update the *merged tokens* if any of their *constituent tokens* (as defined in Sec. 3.3) are marked as outliers. The contributions of the outliers are subtracted from the merged tokens, and the outliers are no longer merged.

Step 3 provides greater flexibility to heads with more outliers under the same overall budget. Additionally, step 4 protects the representational ability of merged tokens and keeps the uniqueness of outliers. Note that we only apply the outlier filtering on query tokens.

We implement a custom CUDA kernel to realize this filtering logic efficiently through block-wise parallelization

			7 Scenes (Stride 10)			NRGBD (Stride 10)		
	Q Ratio	K/V Ratio	Acc.↓	Comp.↓	Time↓	Acc.↓	Comp.↓	Time↓
CUT3R [27]	1.00	1.00	0.041	0.029	4.2s	0.132	0.056	5.7s
VGGT* [25]	1.00	1.00	0.019	0.021	9.1s	0.010	0.010	13.9s
FastVGGT [20]	0.34	0.34	0.018	0.020	4.5s	0.016	0.013	7.0s
VGGT*+HTTM	**0.20**	0.30	0.020	0.023	4.3s	0.012	0.010	6.8s

Table 1. Comparison of 3D reconstruction performance in accuracy (Acc) and completeness (Comp). HTTM achieves better reconstruction quality on fine-grained datasets like NRGBD than FastVGGT using a shorter Q/K/V sequence.

and on-the-fly deviation computation. More details about this CUDA kernel are in the Appendix, and the latency overhead is reported in Sec. 4.2.

4. Experiments

4.1. 3D Reconstruction Results

In this section, we report the quantitative performance of applying HTTM on 7Scenes [21] and NRGBD [1] to reduce the token sequence length of VGGT.

Experiment Setup For the temporal reordering described in Sec. 3.3, we use a spatial block size of $n_s = 128$ and a temporal frame length of $n_t = 30$ to form temporal merging blocks of size $n_b = 3840$. We merge 90% of the query tokens and filter 10% outliers as stated in Sec. 3.4, so that in total, the reduced query sequence is 20% of its original length. For key and value tokens, we use a 70% merging ratio, resulting in 30% of the original length. For FastVGGT, we use its standard merging setup, which results in around 34% actual sequence length. We didn't further increase its merging ratios because FastVGGT already underperforms HTTM on NRGBD in this setup.

The reported reconstruction results use the depth and camera head of VGGT since they yield better results, as stated in the original VGGT paper[25]. For the baseline, we use the VRAM-efficient VGGT implemented in FastVGGT[20], denoted as VGGT*, which discards unused intermediate outputs during inference **without** affecting reconstruction quality. All inferences are performed in Bfloat16[26] using FlashAttention[9] on an Nvidia A100.

Results As shown in Table 1, we evaluate the 3D reconstruction performance on 7Scenes and NRGBD with keyframes sampled every 10 frames. Compared to the baseline VGGT, HTTM maintains comparable performance with much shorter Q/K/V sequence length. Compared to FastVGGT, HTTM surpasses FastVGGT on NRGBD and achieves comparable results on 7Scenes using smaller sequence lengths (higher merging ratios). As shown in Fig. 9, HTTM preserves more high-fidelity details of the reconstruction result from the original VGGT.

4.2. Latency

In this section, we evaluate the acceleration ability of HTTM for long input sequences. The experiment setup is the same as in Sec. 4.1. We compare the latency of HTTM to FastVGGT under similar merging ratios and comparable (or better) task performances in Table 1.

Under similar token merging ratios HTTM shows better latency performance due to our block-wise token merging design after temporal reordering as described in Sec. 3.3. As shown in Table 4, the latency for executing the attention computation is similar for HTTM and FastVGGT when using comparable merging ratios. However, by only performing token merging within temporal merging blocks, HTTM largely reduces the matching cost, leading to the overall latency improvement over FastVGGT in Table 3. Due to our adaptive outlier filtering design introduced in Sec. 3.4, the merged token aggregation latency of HTTM is higher, but the overhead is acceptable compared to the matching cost reduction. Combine the matching and aggregation overhead together, HTTM achieves 4.58× latency reduction on the merging cost as shown in Fig. 1.

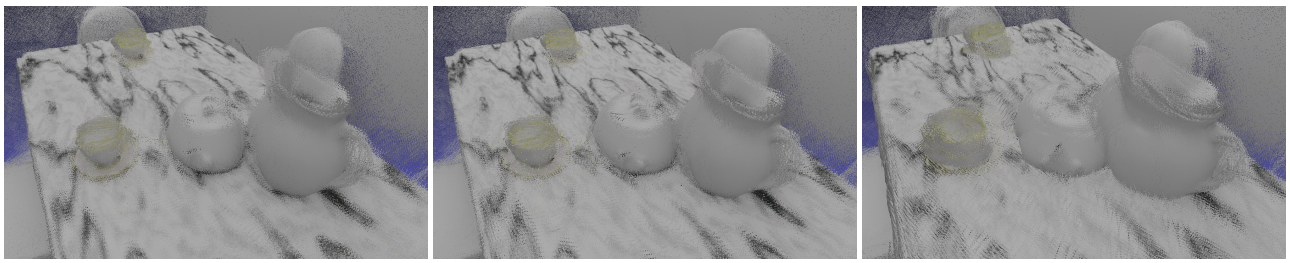
Under comparable task performance HTTM is able to achieve comparable task performance to the baseline VGGT and FastVGGT using more drastic merging ratios as shown in Sec. 4.1. Hence, we report the latency performance over long sequences under this configuration. As shown in Table 3, in this setup, HTTM further reduces the latency. With 1000 input frames, HTTM substantially reduces the latency by 7× compared to the baseline VGGT.

4.3. Experiments on Longer Sequence

In this section, we show more experiments on longer sequences with NRGBD [1] and ScanNet [8] dataset in Table 2.

Setup For the NRGBD dataset, we evaluate HTTM by sampling keyframes every 3 frames. For the ScanNet dataset, we randomly select 15 scenes with over 2000 input frames and sample keyframes every 2 frames. The setup of FastVGGT and our HTTM is the same as in Sec. 4.1.

Results As shown in Table 2, with longer sequence inputs, HTTM constantly shows similar performance to the original VGGT with substantially shorter latency.



(a) Standard VGGT (b) VGGT+HTTM (c) FastVGGT

Figure 9. Qualitative results. Compared to FastVGGT, HTTM preserves more high-fidelity details of VGGT.

			NRGBD (Stride 3)			ScanNet (500 Frames)			ScanNet (1000 Frames)		
	Q Ratio	K/V Ratio	Acc.↓	Comp.↓	Time↓	Acc.↓	Comp.↓	Time↓	Acc.↓	Comp.↓	Time↓
VGGT* [25]	1.00	1.00	0.010	0.009	135.1s	0.011	0.011	177.5s	0.028	0.022	724.6s
FastVGGT [20]	0.34	0.34	0.014	0.020	51.2s	0.012	0.011	52.3s	0.027	0.021	175.2s
VGGT*+HTTM	**0.20**	0.30	0.010	0.008	26.4s	0.011	0.010	35.8s	0.027	0.021	**102.8s**

Table 2. 3D reconstruction performance with longer sequence input. With longer sequence inputs, HTTM constantly shows similar performance to the original VGGT with substantially shorter latency.

4.4. Error Mitigation

In this section, we discuss the error mitigation effect of token merging on VGGT reported by FastVGGT [20].

The original VGGT shows a large error in camera pose estimation when the camera movement is high. For example, in Fig. 10, we visualize the camera pose estimation error on a big scene from ScanNet with large camera movement, using 500 keyframes sampled every 10 frames. Compared to FastVGGT (Fig. 10a), the original VGGT (Fig. 10b) shows much higher error in camera pose estimation. HTTM shows a smaller error compared to the original VGGT, but still higher than FastVGGT. Although FastVGGT offers a discussion of the observed improvement, the specific mechanism responsible for the enhanced error mitigation ability remains insufficiently clarified.

In order to understand this higher error mitigation ability of FastVGGT, we invested it further and we found that the error mitigation effect comes from the first-frame anchoring design in FastVGGT. FastVGGT assigns all tokens in the first frame as `dst` tokens, referring to them as "Reference Tokens" to preserve their strong representativeness. However, we find that the crucial factor is to reduce tokens from subsequent frames that are highly similar to those in the first frame. As shown in Fig. 10d,10e,10f, by adding first frame anchoring and allowing more temporal merging (so that more tokens from subsequent frames can be merged to the first frame), HTTM achieves a similar error mitigation effect to FastVGGT. We speculate that tokens from later frames can mislead the Global Attention module. Because these tokens are highly similar to first-frame tokens, the model may incorrectly treat them as part of the reference frame, weakening the coordinate anchor and amplifying drift. By explicitly designating all first-frame tokens as `dst` tokens and merging highly similar tokens from subsequent frames into them, the ambiguity is suppressed and the reference frame remains stable.

Note that in a large scene with long-sequence input, the tokens from the first frames consists less than 1% of the whole token set, so activating first-frame anchoring introduces negligible overhead for HTTM.

4.5. Spatial Merging vs. Temporal Merging

In this section, we want to explore the effectiveness of spatial merging versus temporal merging. To do that, we first define the *merging cost* (as discussed in Sec. 3.3) as the merging block size $n_b = n_s \times n_t$, where n_s is the spatial block size and n_t is the temporal frame length during the temporal reordering. The *merging quality* is defined as the 10th quantile of the similarities between *merged* matches.

In Fig. 11, we visualize the Pareto front between the merging cost and merging quality on two scenes, one with highly similar, temporally continuous frames, and one with sparse-view frames. Additionally, we visualize the cost composition with colors. Given a fixed cost n_b, points with larger n_s will become reddish, and points with larger n_t will become greenish.

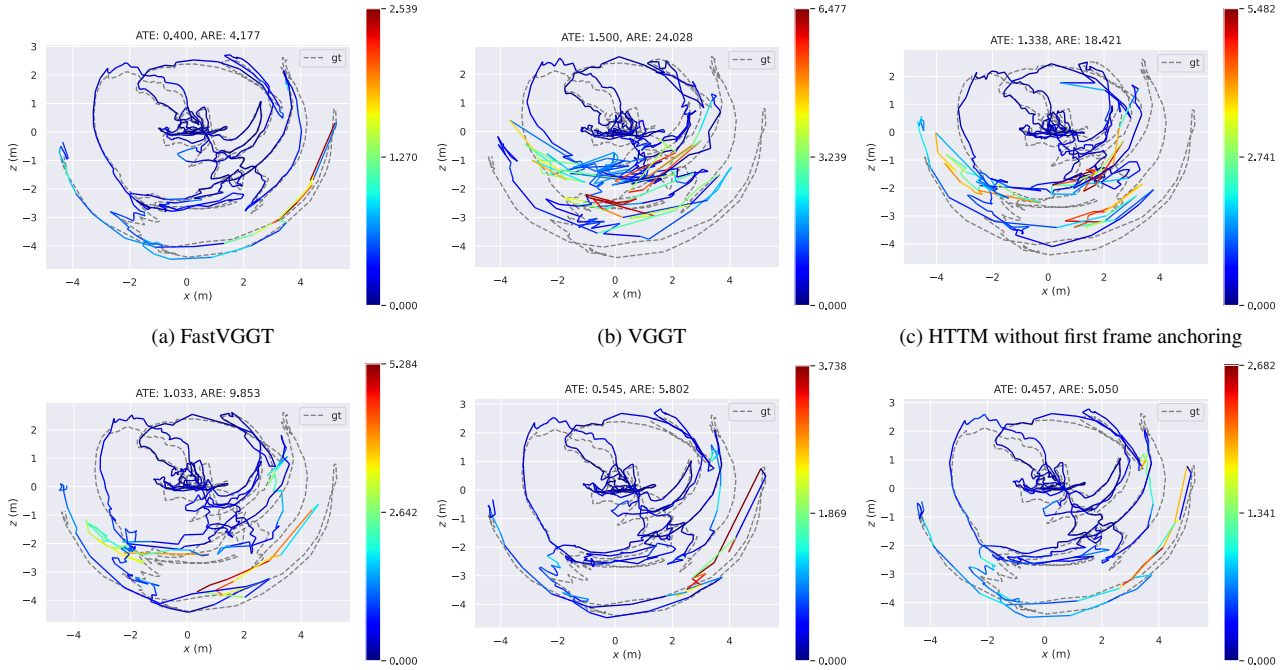
It can be observed that in both cases, merging quality grows with the merging cost. For continuous frames (Fig. 11a), merging predominantly along the temporal dimension yields better performance, which aligns with our observation in Sec. 3.1. For sparse-view frames (Fig. 11b), merging mainly along the spatial dimension becomes more effective. However, when given a higher cost budget ($\geq$ 800), it is still beneficial to perform temporal merging

	Token Ratio		Number of Frames			
	Q	K/V	100	300	500	1000
VGGT*	1.00	1.00	9.1s	60.7s	177.5s	724.6s
Fast VGGT	0.34	0.34	4.5s	22.4s	52.3s	175.2s
VGGT*+HTTM	0.4	0.3	4.4s	18.6s	38.0s	130.0s
VGGT*+HTTM	0.2	0.3	4.3s	16.3s	35.8s	**102.8s**

Table 3. Latency Comparison. At 1000 frames, we are **7**× faster than the baseline VGGT with FlashAttention in Bfloat16.

	HTTM	FastVGGT
Attention Kernel	2.95s	2.97s
Matching	0.12s	2.31s
Merged Token Aggr.	0.41s	0.11s

Table 4. Averaged latency composition of Global Attention layers in HTTM and FastVGGT using comparable merging ratios over 1000 frames.



(a) FastVGGT (b) VGGT (c) HTTM without first frame anchoring

(d) HTTM⋆ with 10 frames of temporal merging (e) HTTM⋆ with 30 frames of temporal merging (f) HTTM⋆ with 40 frames of temporal merging

Figure 10. Comparison of camera pose estimation performance between FastVGGT, VGGT, HTTM without first frame anchoring, and HTTM with first frame anchoring (denoted as HTTM⋆) across different numbers of temporal frames. Colors indicate the deviation from the ground-truth camera trajectory.

(green and yellow points) rather than solely performing spatial merging (red points).

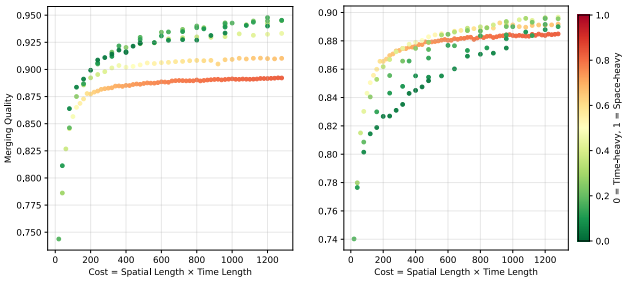
5. Ablation Study

5.1. Adaptive Outlier Filtering

In this section, we show the necessity of our adaptive outlier filtering. We evaluate HTTM on the NRGBD under three merging setups for the query tokens. For the key and value tokens, we merge 70% of them in both cases.

- Merging 90% query tokens with 10% outlier filtering.
- Merging 85% query tokens with 5% outlier filtering.
- Merging 80% query tokens.

These three setups result in the same number of Q/K/V tokens after merging. As shown in Table 5, although the three setups use the same token length, not performing outlier filtering leads to a catastrophic performance drop, showing the



(a) Continuous Frames (b) Sparse-view Frames

Figure 11. The Pareto front illustrates the trade-off between merging cost and merging quality, with color indicating the composition of the cost. Greenish points merge more frames along the temporal dimension, while redish points merge more tokens along the spatial dimension.

necessity of outlier filtering in block-wise token merging as discussed in Sec. 3.4.

	Acc.↓	Comp.↓
Without outlier filtering	0.240	0.310
With 5% outlier filtering	0.013	0.011
With 10% outlier filtering	0.012	0.010

Table 5. Accuracy and completeness on NRGBD with and without outlier filtering using the same token sequence length.

6. Conclusion

In this work, we propose HTTM, a training-free token merging approach that accelerates VGGT's inference. We conduct systematic explorations of similarity patterns in VGGT and analyze the main limitations of existing methods in merging efficiency and representational ability. To leverage the spatial locality and temporal correspondence of VGGT's tokens at the head-embedding level, we introduce a temporal reordering and head-wise adaptive outlier filtering technique that helps HTTM merge tokens efficiently while preserving their uniqueness, leading to substantial acceleration up to **7**$\times$ over long input sequences without performance degradation.

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. 3, 7

[2] Daniel Bolya and Judy Hoffman. Token Merging for Fast Stable Diffusion. *CVPR Workshop on Efficient Deep Learning for Computer Vision*, 2023. 2, 3, 4, 5

[3] 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. 1, 3, 4

[4] Mengzhao Chen, Wenqi Shao, Peng Xu, Mingbao Lin, Kaipeng Zhang, Fei Chao, Rongrong Ji, Yu Qiao, and Ping Luo. Diffrate: Differentiable Compression Rate for Efficient Vision Transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17164–17174, 2023. 2

[5] Xuanyao Chen, Zhijian Liu, Haotian Tang, Li Yi, Hang Zhao, and Song Han. Sparsevit: Revisiting activation sparsity for efficient high-resolution vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2061–2070, 2023. 2

[6] Xingyu Chen, Yue Chen, Yuliang Xiu, Andreas Geiger, and Anpei Chen. TTT3R: 3D Reconstruction as Test-Time Training. *arXiv preprint arXiv:2509.26645*, 2025. 2

[7] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating Long Sequences with Sparse Transformers. *arXiv preprint arXiv:1904.10509*, 2019. 1, 2

[8] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017. 7

[9] Tri Dao. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. In *International Conference on Learning Representations (ICLR)*, 2024. 7

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019. 3

[11] Zhanzhou Feng, Jiaming Xu, Lei Ma, and Shiliang Zhang. Efficient Video Transformers via Spatial-temporal Token Merging for Action Recognition. *ACM Transactions on Multimedia Computing, Communications and Applications*, 20 (4):1–21, 2024. 3

[12] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, and et al. The llama 3 herd of models, 2024. 2

[13] Jeongseok Hyun, Sukjun Hwang, Su Ho Han, Taeoh Kim, Inwoong Lee, Dongyoon Wee, Joon-Young Lee, Seon Joo Kim, and Minho Shim. Multi-Granular Spatio-Temporal Token Merging for Training-Free Acceleration of Video LLMs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 23990–24000, 2025. 3

[14] Minchul Kim, Shangqian Gao, Yen-Chang Hsu, Yilin Shen, and Hongxia Jin. Token Fusion: Bridging the Gap between Token Pruning and Token Merging. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1383–1392, 2024. 2, 3

[15] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The Efficient Transformer. In *International Conference on Learning Representations*, 2020. 1

[16] Seon-Ho Lee, Jue Wang, Zhikang Zhang, David Fan, and Xinyu Li. Video Token Merging for Long-form Video Understanding. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2024. Curran Associates Inc. 3

[17] Vincent Leroy, Yohann Cabon, and Jerome Revaud. Grounding Image Matching in 3D with MASt3R, 2024. 2

[18] Hao Liu, Matei Zaharia, and Pieter Abbeel. Ring attention with blockwise transformers for near-infinite context. *arXiv preprint arXiv:2310.01889*, 2023. 2

[19] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-Resolution Image Synthesis with Latent Diffusion Models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. 3

[20] You Shen, Zhipeng Zhang, Yansong Qu, and Liujuan Cao. FastVGGT: Training-Free Acceleration of Visual Geometry Transformer. *arXiv preprint arXiv:2509.02560*, 2025. 2, 4, 7, 8

[21] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. Scene Coordinate Regression Forests for Camera Relocalization in RGB-D Images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2930–2937, 2013. 7

[22] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced Transformer with Rotary Position Embedding. *Neurocomputing*, 568: 127063, 2024. 3, 1

[23] Apoorv Vyas, Angelos Katharopoulos, and François Fleuret. Fast Transformers with Clustered Attention. *Advances in Neural Information Processing Systems*, 33:21665–21674, 2020. 1

[24] Chung-Shien Brian Wang, Christian Schmidt, Jens Piekenbrinck, and Bastian Leibe. Faster VGGT with Block-Sparse Global Attention, 2025. 2, 3

[25] 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. 1, 2, 3, 7, 8

[26] Norman P. Jouppi Wang, Cliff Young, and David Patterson. BFloat16: The secret to high performance on Cloud TPUs. In *Google White Paper*, 2019. 7

[27] 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. 2, 7

[28] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. DUSt3R: Geometric 3D Vision Made Easy. In *CVPR*, 2024. 2

[29] Weitian Wang, Rai Shubham, Cecilia De La Parra, and Akash Kumar. Mixa-q: Revisiting activation sparsity for vision transformers from a mixed-precision quantization perspective. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22143–22152, 2025. 2

[30] Zhenhailong Wang, Senthil Purushwalkam, Caiming Xiong, Silvio Savarese, Heng Ji, and Ran Xu. DyMU: Dynamic Merging and Virtual Unmerging for Efficient VLMs. *arXiv preprint arXiv:2504.17040*, 2025. 2

[31] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient Streaming Language Models with Attention Sinks. In *The Twelfth International Conference on Learning Representations*, 2024. 1

[32] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big Bird: Transformers for Longer Sequences. *Advances in neural information processing systems*, 33:17283–17297, 2020. 1, 2

[33] Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, et al. SparseVLM: Visual Token Sparsification for Efficient Vision-Language Model Inference. In *International Conference on Machine Learning*, 2025. 2

HTTM: Head-wise Temporal Token Merging for Faster VGGT

Supplementary Material

A. RoPE's Effect on the Similarity Pattern

In this section, we investigate the Rotary Position Embedding (RoPE [22])'s effect on the strong periodic patterns observed in Fig. 3. To show that the high similarity values near the off-diagonals do emerge from the RoPE, in Fig. 12, we visualize the similarity map of query tokens in the early Frame Attention layers with **non-overlapping** input frames. It can be observed that the input feature of the first Frame Attention layer (DINO features) does not exhibit high temporal values near off-diagonals (Fig. 12a), which align with the non-overlapping input frames. However, after applying the frame-wise RoPE for the first time, high similarity values near the off diagonals emerge as shown in Fig. 12b. After that, it can be observed that, in each layer, the spatial distinctiveness within frames is enhanced after applying the frame-wise RoPE.

In Fig. 13, we visualize the similarity maps with temporally continuous input frames. With these inputs, the changes in similarity patterns before and after applying RoPE are similar, but high similarity values near the off-diagonals are more vivid.

B. Theoretical Proofs For Block-Wise Token Merging

We provide proofs for the three statements made in Sec. 3.3. For clarity, we omit the head index i in this section. Let the global source and destination token sets be $\mathcal{S}$ and $\mathcal{D}$, $\mathcal{S} \cap \mathcal{D} = \emptyset$. The entries of the global similarity matrix $W \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{D}|}$ is defined as:

$$W_{ij} = \mathrm{sim}(s_i, d_i), \quad s_i \in \mathcal{S}, d_j \in \mathcal{D}$$

For a merging budget r, the merging rule selects the *top-r* source–destination pairs with the largest similarities. For any selected set $\mathcal{M}$ of merging candidates, the merging quality Q is defined as the *average* similarity between merged matches:

$$Q(\mathcal{M}) := \frac{1}{r} \sum_{(s,d)\in\mathcal{M}} \mathrm{sim}(s, d).$$

In block-wise token merging, we partition the tokens into K disjoint blocks $\{\mathcal{B}_1, \ldots, \mathcal{B}_K\}$. Assuming the same splitting strategy, the source and destination token sets $\mathcal{S}_k$ and $\mathcal{D}_k$ inside block $k \in \{1, \ldots, K\}$ are:

$$\mathcal{S}_k = \mathcal{S} \cap \mathcal{B}_k, \qquad \mathcal{D}_k = \mathcal{D} \cap \mathcal{B}_k,$$

After establishing the notations, we proceed to prove the aforementioned statements.

1. Block similarity matrices are submatrices of the global matrix

Prop. For every block $\mathcal{B}_k$, its block-wise similarity matrix $W^{(k)}$ is a submatrix of W.

Proof. For each $\mathcal{B}_k$, the entries of its similarity matrix is defined as:

$$W^{(k)}_{i,j} = \mathrm{sim}(s_i, d_j), \qquad s_i \in \mathcal{S}_k,\ d_j \in \mathcal{D}_k.$$

Since $\mathcal{S}_k \subseteq \mathcal{S}, \mathcal{D}_k \subseteq \mathcal{D}$, it follows that $s_i \in \mathcal{S},\ d_j \in \mathcal{D}$. Therefore, each entry $\mathrm{sim}(s_i, d_j)$ is also a entry in W.

2. Merging quality depends on how many high-similarity pairs fall inside blocks

Prop. Let $\mathcal{E} = \mathcal{S} \times \mathcal{D}$ be all possible source–destination pairs, and let

$$\mathcal{E}_{\mathrm{blk}} := \bigcup_{k=1}^{K} (\mathcal{S}_k \times \mathcal{D}_k)$$

be the set of pairs permitted by block-wise merging. If more large entries of W lie inside $\mathcal{E}_{\mathrm{blk}}$, then the block-wise merging quality increases.

Proof. As stated in Sec. 3.2, we pick the top-r best matches with the highest similarity, which yields the global optimal merging quality:

$$\mathcal{M}^\star = \arg\max_{\mathcal{M}} \mathrm{sim}(s, d) = \arg\max_{\mathcal{M}} Q(\mathcal{M})$$
$$\text{s.t. } (s, d) \subset \mathcal{M}, \quad \mathcal{M} \subseteq \mathcal{E}, \quad |\mathcal{M}| = r.$$

Block-wise merging is constrained to subsets of $\mathcal{E}_{\mathrm{blk}}$:

$$\mathcal{M}^\star_{blk} = \arg\max_{\mathcal{M}} \mathrm{sim}(s, d)$$
$$\text{s.t. } (s, d) \subset \mathcal{M}, \quad \mathcal{M} \subseteq \mathcal{E}_{blk}, \quad |\mathcal{M}| = r.$$

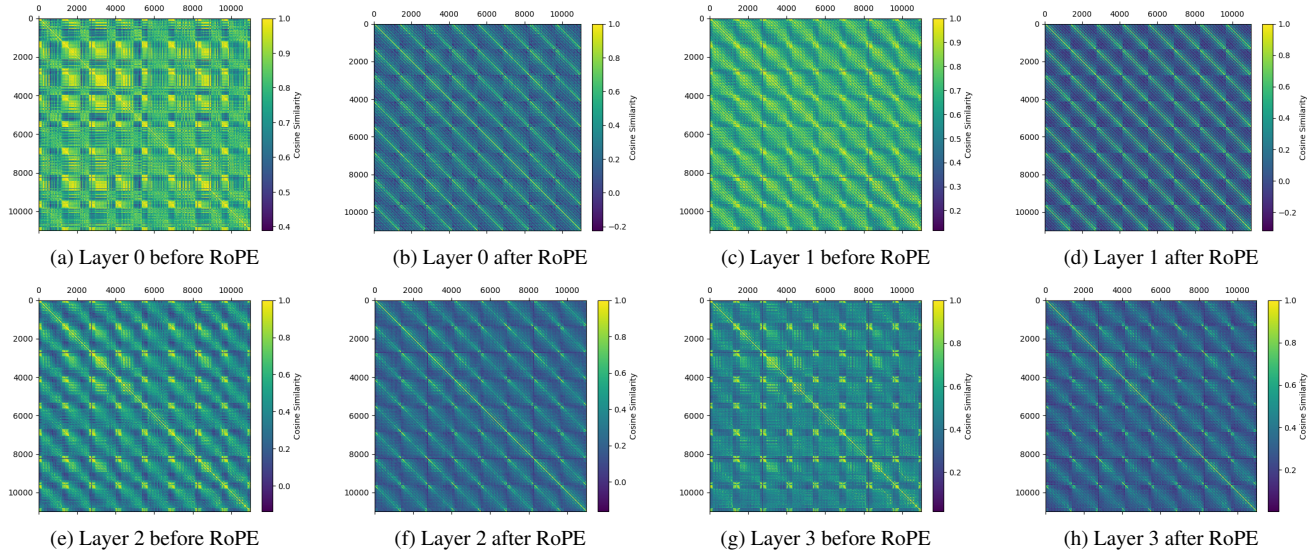
Since $\mathcal{E}_{\mathrm{blk}} \subseteq \mathcal{E}$, the feasible set of block-wise solutions is smaller, hence

$$Q(\mathcal{M}^\star_{\mathrm{blk}}) \le Q(\mathcal{M}^\star).$$

Define H_{blk} as the number of optimal merging candidates included in $\mathcal{M}^\star_{blk}$:

$$H_{blk} := |\mathcal{M}^\star_{blk} \cap \mathcal{M}^\star|$$

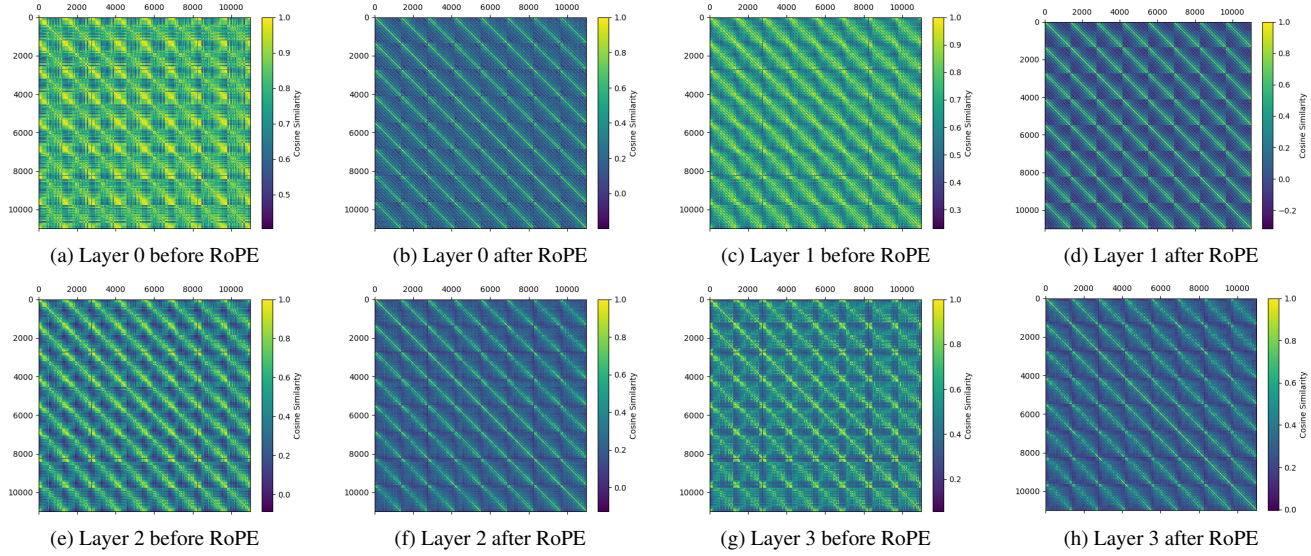
If $H_{blk} < r$, then $Q(\mathcal{M}^\star_{blk})$ is strictly smaller than $Q(\mathcal{M}^\star)$. By including more s, d pairs with high similarity in $\mathcal{E}_{blk}$, we can only increase H_{blk}. Therefore, block-wise merging quality is monotone in the number of high-similarity entries of W located inside the blocks.



(a) Layer 0 before RoPE (b) Layer 0 after RoPE (c) Layer 1 before RoPE (d) Layer 1 after RoPE

(e) Layer 2 before RoPE (f) Layer 2 after RoPE (g) Layer 3 before RoPE (h) Layer 3 after RoPE

Figure 12. Query token similarity maps before and after RoPE in Frame Attention layers with non-overlapping input frames



(a) Layer 0 before RoPE (b) Layer 0 after RoPE (c) Layer 1 before RoPE (d) Layer 1 after RoPE

(e) Layer 2 before RoPE (f) Layer 2 after RoPE (g) Layer 3 before RoPE (h) Layer 3 after RoPE

Figure 13. Query token similarity maps before and after RoPE in Frame Attention layers with temporally continuous input frames.

3. Larger blocks improve merging quality but require more computations

Prop. Fix a splitting strategy that forms blocks of size n_b. If the block size increases, then (i) the achievable merging quality does not decrease, and (ii) the computational cost grows approximately linearly in n_b.

Proof. *(i) Larger blocks improve or maintain quality.*

Let two block sizes $n_b^{(1)} < n_b^{(2)}$ be given. Under the same splitting strategy, every small block is contained in a unique larger block. Hence

$$\mathcal{E}_{\text{blk}}^{(1)} \subseteq \mathcal{E}_{\text{blk}}^{(2)}.$$

Thus, the feasible merging sets under smaller blocks are a subset of those under the larger blocks, so

$$\max_{\substack{\mathcal{M}\subseteq\mathcal{E}_{\text{blk}}^{(1)} \\ |\mathcal{M}|=r}} Q(\mathcal{M}) \;\leq\; \max_{\substack{\mathcal{M}\subseteq\mathcal{E}_{\text{blk}}^{(2)} \\ |\mathcal{M}|=r}} Q(\mathcal{M}).$$

Therefore, the optimal block-wise average similarity is non-decreasing in n_b.

(ii) Cost increases linearly with block size.

Let $n_s^{(k)} = |\mathcal{S}_k|$ and $n_d^{(k)} = |\mathcal{D}_k|$, with $n_s^{(k)} + n_d^{(k)} = n_b$. Computing similarities inside block k costs $\Theta(n_s^{(k)} n_d^{(k)} d)$ operations, where d is the head dimension.
Assuming a fixed source/destination ratio:

$$n_s^{(k)} = \alpha n_b \quad n_d^{(k)} = (1-\alpha) n_b \,,$$

We have that

$$n_s^{(k)} n_d^{(k)} = \alpha(1-\alpha) n_b^2 \,.$$

With $K \approx N/n_b$ blocks, the total cost can be approximated as

$$\sum_{k=1}^{K} \Theta(\alpha(1-\alpha) n_b^2 d) \approx \Theta\big(\alpha(1-\alpha)\frac{N}{n_b}\, n_b^2 d\big) = O(N n_b d),$$

which grows linearly with block size n_b.

Combining both parts, larger blocks always improve (or maintain) merging quality but incur proportionally higher computational costs.