ICME 2026 Paper Title

Anonymous ICME submission

***Abstract*—Feed forward 3D foundation models such as VGGT deliver strong reconstruction quality in a single pass setting, but their use of global attention across all frames scales quadratically with sequence length. In dense capture regimes, where many frames observe almost the same content with heavy visual overlap, the marginal value of additional frames quickly saturates while the attention cost continues to rise. Existing acceleration methods focus on redundancy inside the attention layers through various forms of token compression. We take a data driven approach and target dense sequences directly. From the features already computed inside VGGT, we estimate a simple measure of frame density based on pairwise affinities and partition the sequence into a small number of subscenes. A soft assignment scheme produces balanced groups while respecting gradual geometric changes. A central idea is to let all subscenes share a common reference frame, so that VGGT can process them independently without any additional alignment. This turns dense sequences into parallel subproblems and yields substantial reductions in global attention cost without degrading reconstruction quality.**

***Index Terms*—**

I. Introduction

We explore redundancy directly from input.

II. Method

a) VGGT Input and Alternating-Attention Backbone.: Given a sequence of N RGB images $\mathbf{I} = \{\mathbf{I}_s\}_{s=1}^{N}$, each image $\mathbf{I}_s \in \mathbb{R}^{3\times H\times W}$ is encoded by the DINOv2 backbone into P patch tokens:

$$\mathbf{F}_s = f_{\text{DINO}}(\mathbf{I}_s) \in \mathbb{R}^{P\times C}.$$

Following VGGT, we additionally append one camera token and four register tokens to each frame, so that each frame actually contains $(P + 5)$ tokens in total. VGGT processes these tokens with L layers of *alternating attention*. In each layer, a frame-wise self-attention is first applied independently to each frame, followed by a global cross-frame attention operating on the concatenation of all frame tokens:

$$\text{Concat}[\mathbf{F}_1, \ldots, \mathbf{F}_N] \in \mathbb{R}^{(N(P+5))\times C}.$$

Thus, every global attention layer must jointly process $N(P+5)$ tokens, making it the dominant computational cost in long sequences.

b) Motivation.: In multi-view 3D reconstruction, an implicit assumption is that input images share sufficient visual overlap—either densely (as in video) or sparsely (as in unordered image collections). However, when the sequence becomes long, the degree of overlap is highly non-uniform: some portions contain redundant viewpoints, while others exhibit rapid geometric changes. VGGT's global attention treats all frames equally and must process $N(P + 5)$ tokens at once, making its computational cost scale quadratically with sequence length and causing unrelated frames to interact unnecessarily.

Our goal is therefore to estimate the *scene density* of the sequence and reorganize it into a small number of coherent subscenes so that global attention is applied only within meaningful, density-consistent regions. Rather than clustering purely by visual similarity, we aim to group frames that provide sufficient mutual support for 3D reasoning, while separating frames whose overlap or contextual support is weak. This naturally leads to a graph interpretation, where frames form nodes and their affinities define edge weights. Subscene grouping becomes a lightweight graph partitioning problem that reduces computational cost while preserving the structural relationships most relevant to 3D reconstruction.

c) Stage 1: Frame-Level Affinity Estimation.: A straightforward way to assess frame overlap is to use retrieval-style techniques such as VLAD or other feature aggregation methods, but these approaches introduce extra computation and latency when the sequence is long. In practice, we find that the DINOv2 feature maps produced for each frame already provide sufficiently reliable cues for measuring mutual support. To further accelerate the process, we abstract each frame into a lightweight descriptor obtained by averaging its P patch tokens along the patch dimension, producing a single C-dimensional vector per frame.

Using these descriptors, we compute pairwise cosine affinity across all frames and obtain an $N \times N$ similarity matrix $\mathbf{S} \in \mathbb{R}^{N\times N}$, where each entry S_{ij} reflects how strongly frame i is supported by frame j in terms of viewpoint overlap and structural consistency. This matrix serves as the basis for estimating scene density and deriving subscene groups in the following stages.

d) Stage 2: Estimating Scene Density and Determining Group Count.: The similarity matrix $\mathbf{S}$ reveals how densely each frame is supported by its neighbors. Frames in stable, slowly changing regions exhibit many high-affinity connections, whereas frames near scene transitions have much weaker support. To quantify this structure, we count for each frame how many other frames exceed a fixed similarity threshold, and average this quantity over the entire sequence. This scalar serves as a lightweight estimate of the sequence's overall density.

Based on this density statistic, we determine the number of subscenes K by mapping it to a bounded integer range. Dense sequences naturally yield fewer groups, while sequences with rapid viewpoint or content changes produce more groups. This adaptive choice of K allows the model to adjust to both video-like dense inputs and sparse multi-view collections without

manual tuning.

e) Stage 3: Subscene Partition via Soft Assignments.: Once the number of groups K is determined, we partition the sequence into subscenes. The goal of this step is twofold. First, partitioning allows the model to process subscenes in parallel, significantly reducing global-attention cost. Second, we want the resulting subscenes to be directly mergeable without performing expensive point-cloud alignment between them, which requires each subscene to retain internally consistent viewpoints while remaining sufficiently separated from others.

To achieve this, we introduce a soft assignment matrix

$$\mathbf{A} \in \mathbb{R}^{N\times K}, \qquad A_{sk} \geq 0, \quad \sum_{k=1}^{K} A_{sk} = 1,$$

where each row encodes the tentative membership of frame s to the K subscenes. The soft formulation enables us to impose several desirable structural properties before converting the assignments into hard labels.

We derive three simple losses that encourage the partition to reflect the support structure encoded in $\mathbf{S}$ while preventing degenerate outcomes.

f) (1) Consistency Loss.: Frames within a subscene should share similar affinity neighborhoods. For group k, we define its aggregated affinity profile

$$\mathbf{g}_k = \frac{1}{m_k} \sum_{s=1}^{N} A_{sk}\, \mathbf{S}_{s:}, \qquad m_k = \sum_{s=1}^{N} A_{sk},$$

and encourage it to match the global average affinity pattern

$$\mathbf{g}_{\text{avg}} = \frac{1}{N} \sum_{s=1}^{N} \mathbf{S}_{s:}.$$

The consistency loss

$$\mathcal{L}_{\text{cons}} = \sum_{k=1}^{K} \|\mathbf{g}_k - \mathbf{g}_{\text{avg}}\|_2^2$$

ensures that each group corresponds to a coherent region of the sequence rather than capturing disjoint frames with unrelated neighborhoods. This is crucial for later merging, since subscenes with incompatible affinity structure would require geometric alignment, which we want to avoid.

g) (2) Coverage Loss.: Every frame must be confidently assigned to exactly one subscene; otherwise some frames may drift between groups. We enforce

$$\mathcal{L}_{\text{cov}} = \sum_{s=1}^{N} \left(\sum_{k=1}^{K} A_{sk} - 1 \right)^2,$$

which encourages each row of $\mathbf{A}$ to form a valid probability distribution, preventing ambiguous or incomplete assignments.

h) (3) Balance Loss.: To support parallel processing, each subscene should contain a comparable number of frames. Extremely small or extremely large groups reduce efficiency or overload a single subscene. We therefore encourage the soft group sizes m_k to be close to N/K:

$$\mathcal{L}_{\text{bal}} = \sum_{k=1}^{K} \left(m_k - \frac{N}{K} \right)^2.$$

i) Final Objective.: The overall objective is a weighted combination

$$\mathcal{L} = \lambda_{\text{cons}}\mathcal{L}_{\text{cons}} + \lambda_{\text{cov}}\mathcal{L}_{\text{cov}} + \lambda_{\text{bal}}\mathcal{L}_{\text{bal}}.$$

After optimization, $\mathbf{A}$ becomes sharply peaked. We extract hard assignments by selecting the dominant entry in each row and optionally applying a capacity-aware correction step. The resulting subscenes preserve internal coherence, remain well balanced, and can be processed independently without requiring subsequent geometric alignment.

j) Stage 4: Grouped Sequence Construction.: Let $\hat{k}(s) \in \{1, \ldots, K\}$ be the hard subscene label of frame s. For each group k, denote its member indices by

$$\mathcal{G}_k = \{\, s \mid \hat{k}(s) = k \,\}, \qquad k = 1, \ldots, K.$$

A distinctive property of VGGT is that all geometric predictions are expressed in the coordinate system anchored at frame 0. To preserve this shared reference across all subscenes, we explicitly prepend frame 0 to every group. Formally, for each subscene k we construct a new ordered index sequence

$$\mathcal{S}_k = [\, 0\,;\, \text{Sort}(\mathcal{G}_k)\,],$$

where "$[\,0\,;\,\cdot\,]$" denotes concatenation with frame 0 placed at the front. All sequences are then padded or truncated to a unified length S_{eff}.

After reindexing, the effective batched inputs become

$$\text{images}_{\text{eff}} = \text{Gather}\big(\text{images}, \mathcal{S}_k\big) \;\in\; \mathbb{R}^{(BK)\times S_{\text{eff}}\times 3\times H\times W},$$

and similarly for the corresponding token representations. By inserting the shared anchor frame into every subscene, VGGT processes each group independently while all predictions remain expressed in a common coordinate frame, eliminating the need for any cross-subscene geometric alignment.

III. Experiments

A. Baselines

We adopt VGGT [1] and the subsequent extensions of VGGT and DUSt3R [2] as comparison baselines, including Fast3R [3], which processes multiple views in parallel, and Spann3R [4], which replaces DUSt3R's costly global alignment stage with a sequential frame-processing scheme augmented by an external spatial memory. VGGT* denotes the VRAM-efficient variant of VGGT, enabling larger input sequences. FastVGGT [5] further represents an early train-free acceleration attempt for VGGT by introducing a token-merging strategy to improve inference efficiency. All evaluations are performed on a single NVIDIA A100 GPU, and for fair comparison, all models are loaded using the `bfloat16` data type.

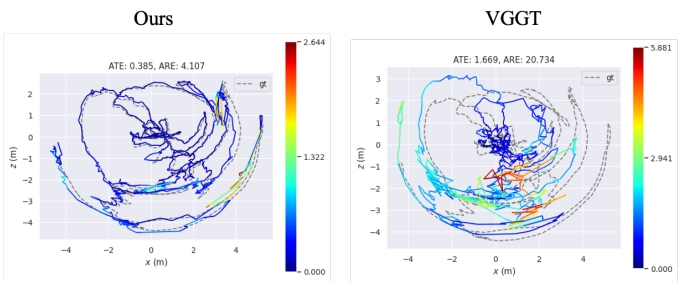

Fig. 1: Comparison of pose estimation performance between our method and VGGT.

B. Camera Pose Estimation

We evaluate camera pose estimation on unseen trajectories drawn from a uniformly sampled subset of ScanNet [6], reporting Absolute Trajectory Error (ATE), Absolute Rotation Error (ARE), Relative Pose Error of translation (RPE-trans), and Relative Pose Error of rotation (RPE-rot), along with reconstruction accuracy measured by the Chamfer Distance (CD). All experiments are conducted on input sequences of length 1000 to clearly expose the efficiency advantages of our method.

As shown in Table I, our method outperforms all compared baselines in both absolute pose error metrics and reconstruction quality, while achieving the highest runtime efficiency at nearly 5.7 FPS, corresponding to a 3.9× speedup over the original VGGT. Although follow-up variants of DUSt3R struggle with camera pose estimation on this dataset, they still deliver strong reconstruction quality and maintain competitive FPS performance. In contrast, Spann3R [4] fails on long sequences due to its memory limitations.

TABLE I: Quantitative results of camera pose estimation and point cloud reconstruction on the ScanNet dataset with input sequences of 1000 images. *OOM* denotes out-of-memory.

Method	ATE ↓	ARE ↓	RPE-rot ↓	RPE-trans ↓	CD ↓	FPS ↑
Fast3R	1.665	97.848	28.153	0.371	0.616	3.116
CUT3R	?	?	?	?	?	?
Spann3R	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*	*OOM*
VGGT*	0.190	4.351	0.864	0.038	0.463	1.458
FastVGGT	0.162	3.805	0.656	0.030	0.423	5.200
Ours	0.145	3.576	0.665	0.053	0.405	5.699

C. 3D Reconstruction

We compare the 3D reconstruction results of our method against scene-level benchmarks, including 7-Scenes [7] and Neural RGB-D [8]. For each scene, keyframes are sampled every 3 frames to form long-sequence inputs. As shown in Table II, our method consistently delivers robust reconstruction performance while providing substantial acceleration over all baselines. It is worth noting that evaluating the predicted point clouds is computationally expensive due to the required alignment procedure; therefore, we employ a uniform random sampling strategy to retain at most 10^6 points.

D. Ablation Study

1) Loss Effectiveness: In this section, we explore the effectiveness of our proposed loss functions on NRGBD [8] dataset.

References

[1] 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*, 2025, pp. 5294–5306.

[2] 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*, 2024, pp. 20697–20709.

[3] Jianing Yang, Alexander Sax, Kevin J Liang, Mikael Henaff, Hao Tang, Ang Cao, Joyce Chai, Franziska Meier, and Matt Feiszli, "Fast3r: Towards 3d reconstruction of 1000+ images in one forward pass," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 21924–21935.

[4] Hengyi Wang and Lourdes Agapito, "3d reconstruction with spatial memory," in *2025 International Conference on 3D Vision (3DV)*. IEEE, 2025, pp. 78–89.

[5] You Shen, Zhipeng Zhang, Yansong Qu, and Liujuan Cao, "Fastvggt: Training-free acceleration of visual geometry transformer," *arXiv preprint arXiv:2509.02560*, 2025.

[6] 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] 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*, 2013, pp. 2930–2937.

[8] 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*, 2022, pp. 6290–6301.

TABLE II: Quantitative results of point cloud reconstruction on the Neural RGB-D [8] dataset and 7-Scenes [7] dataset. Stride denotes keyframes sampled every 3 frames.

Method	NRGBD - Stride 3							7 Scenes - Stride 3						
	Acc ↓		Comp ↓		NC ↑		FPS ↑	Acc ↓		Comp ↓		NC ↑		FPS ↑
	Mean	Med.	Mean	Med.	Mean	Med.		Mean	Med.	Mean	Med.	Mean	Med.	
Fast3R	0.071	0.035	0.031	0.011	0.608	0.678	7.296	0.048	0.019	0.050	0.013	0.584	0.626	8.579
CUT3R	?	?	?	?	?	?	?	?	?	?	?	?	?	?
Spann3R	0.666	0.385	0.257	0.174	0.554	0.581	7.537	0.394	0.244	0.191	0.099	0.533	0.549	7.851
VGGT*	0.029	0.019	0.018	0.010	0.658	0.781	3.644	0.019	0.009	0.027	0.010	0.622	0.698	4.317
FastVGGT	0.031	0.020	0.019	0.010	0.663	0.790	8.027	0.018	0.008	0.026	0.010	0.626	0.739	9.625
Ours	0.033	0.023	0.017	0.010	0.642	0.739	11.001	0.023	0.012	0.022	0.008	0.623	0.700	11.731