PREPARE: PRE-AA TOKEN PRUNING FOR FROZEN MULTI-VIEW GEOMETRY TRANSFORMERS

Anonymous authors
Paper under double-blind review

ABSTRACT

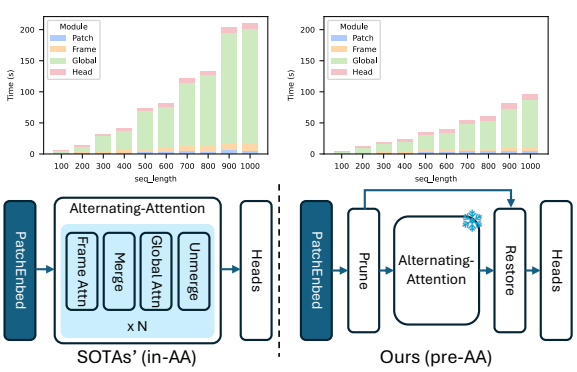
Multi-view geometry transformers are feed-forward 3D foundation models that jointly predict depth maps, point maps, and camera poses for N images in a single forward pass. Most of them build on an alternating-attention (AA) stack whose cross-view attention runs over the patch tokens of all N views in every block, so its cost grows quadratically in N: on 300-frame ScanNetv2 clips, VGGT peaks at 72.3 GiB, and MapAnything does not fit on a 93 GiB card. Existing methods reduce tokens or attention inside the AA stack, so the full patch grid still enters it, and the pre-AA interface, where the encoder hands its tokens to the first AA block, has been overlooked. We propose PrePARE, which prunes patch tokens once at the pre-AA interface and restores the dense grid once after the last AA block. Every block of the frozen AA stack then runs on the reduced tokens, while the prediction heads receive the full grid. A Token Scorer, supervised by how the unpruned AA stack uses each token, and a Feature-guided Restoration module are the only trained parts. The Token Routing module is rule-based. On ScanNetv2 at $N = 300$, PrePARE reduces the peak memory of VGGT by 80 %, from 72.3 to 14.4 GiB, below every in-AA method we ran, and runs 5.4 times faster, while its per-scene Chamfer distance is lower than the unpruned model's on 34 of 50 scenes (sign test $p = 0.015$). On π^3 and Depth Anything 3 it saves 15 and 20 % of peak memory, where the in-AA methods save at most 9 %, and at 1000 frames it is the only method we ran that fits Depth Anything 3 and MapAnything on one GPU.

1 INTRODUCTION

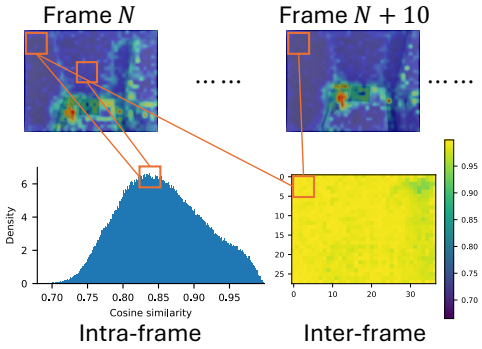
Multi-view geometry transformer, e.g., VGGT (Wang et al., 2025b), π^3 (Wang et al., 2026d), MapAnything(Keetha et al., 2026), DepthAnything3(Lin et al., 2025) are feed-forward 3D foundation models that takes a set of N images and jointly predicts their depth maps, point maps, and camera poses in a single forward pass. Most of these models build on an alternating-attention (AA) stack (Wang et al., 2025b). The stack alternates within-view attention, which refines the tokens of each image, with cross-view attention, which lets the tokens of all images exchange information so that the predicted depth, point maps, and camera poses agree across views. Because the cross-view attention runs over the patch tokens of all N views in every block, its compute grows quadratically in N (Wang et al., 2025a; Ren et al., 2026), and the AA stack becomes the main cost of the model as the input grows longer (Shen et al., 2026). This will affect all models using AA stack in their pipeline.

Recent efficient optimization works mainly focus on reducing processing in AA stack (in-AA): tokens are merged within global attention (Shen et al., 2026; Shu et al., 2025; Chen et al., 2025; Wang et al., 2026c), queries and keys are reduced asymmetrically (Tang et al., 2026), or the attention operator is sparsified (Wang et al., 2025a; Ren et al., 2026). Current findings suggest that token similarity not only highly redundant among intra-frame and inter-frame (Figure 1b), but also barely changes from one layer to the next (Shu et al., 2025). Aforementioned SOTAs reduce tokens only after the full patch grid has entered the AA stack, so much of the stack still runs on all NP tokens. For example, Co-Me starts merging at the 15th of VGGT's 24 blocks (Chen et al., 2025), and SAF3R makes the attention inside each block sparse (Deng et al., 2026a). A token removed at the interface of vision encoder and AA blocks (pre-AA) will save the computaional cost and decision cost in every block of the stack (Figure 1a), which also shows in Section 3.1, i.e., **the pre-AA pruning, potentially yield lerger efficiency improvement, has been overlooked.**

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



(a) Structural difference between SOTAs and Ours pruning.



(b) Pairwise cosine similarity of DINO patch tokens at the output of the frozen DINO backbone.

Figure 1: Token pruning in Geometry Transformer. (a) In-AA methods prune inside the AA stack; PrePARE prunes before it and restores after it, every AA block runs on fewer tokens and latency grows slower. (b) Vision Encoder's patch tokens are highly similar within and across frames.

We propose PrePARE (**Pre**-AA **P**runing and post-**A**A **RE**storation), which prunes patch tokens once at the pre-AA interface and restores the dense patch grid once at the post-AA interface. Every block of the frozen AA stack therefore runs on the reduced tokens, and the prediction heads still receive the full grid (Figure 2). PrePARE consists of three lightweight modules. The *Token Scorer* reads each frame's encoder feature map and predicts how much the AA stack will rely on each patch token. The *Token Routing module* splits each frame's scored tokens into keep, merge, and prune sets under a per-frame keep budget. It folds each merge token into its most similar kept token in the same frame and passes the reduced sequence to the unmodified AA stack. The *Feature-guided Restoration module* rebuilds the dense grid per frame, in which the encoder features of all patch tokens query the AA outputs of the kept tokens. The scorer has to decide before any cross-view attention runs, so we supervise it with how the unpruned AA stack uses each token. In one unpruned forward pass, we record how much attention each token receives from the anchor tokens that the pose or scale head decodes, and the largest cross-view attention it places on another view. We first distill this AA-usage target into the scorer. We then train the scorer and the restoration module together to reproduce the unpruned model's features and predictions. The backbone stays frozen in both stages.

Because PrePARE changes the backbone only at the pre-AA and post-AA interfaces, we can apply the our framework to VGGT, π^3, MapAnything, and Depth Anything 3 by specifying each model's interfaces and anchor tokens (Table A2). On ScanNetv2 at $N = 300$, PrePARE lowers the peak memory of VGGT by 80 %, from 72.3 to 14.4 GiB, and it runs 5.4 times faster, outperforming existing SOTAs. On MapAnything, whose unpruned model runs out of memory on 300-frame clips, PrePARE runs these clips in 27.5 GiB and 1.2 times faster than Co-Me. On π^3 and Depth Anything 3, it saves 15 and 20 % of peak memory and runs 2.6 and 3.6 times faster than the respective unpruned models without a significant change in Chamfer distance, while the in-AA methods save at most 9 %.

The main contributions are as follows. 1) We identify the pre-AA pruning's potential to optimize the latency and memory cost across the whole AA stack of a frozen multi-view geometry transformer. 2) We propose PrePARE, a pre-AA framework with a Token Scorer, a Token Routing module, and a Feature-guided Restoration module. 3) Evaluation on four backbone models using three datasets shows that PrePARE outperforms SOTAs while maintaining the quality among all tasks.

2 Related Work

Efficient geometry transformers. Works on making geometry transformers faster falls into four groups. Token reduction merges or selects tokens inside the attention layers of the AA stack (Shen et al., 2026; Shu et al., 2025; Wang et al., 2026c; You et al., 2026; Tang et al., 2026; Zheng et al., 2026), or merges them from the middle of the encoder onward (Chen et al., 2025). Attention redesign makes the cross-view attention sparse (Wang et al., 2025a; Kim et al., 2026; Huang et al., 2026; Ren et al., 2026; Deng et al., 2026a) or replaces it with cheaper operators (Wang & Xu, 2026; Zhang et al., 2026; Wang et al., 2026a). Sequence-level methods split long inputs into chunks that are aligned afterwards (Deng et al., 2026b; Park et al., 2026; Li et al., 2026), or run causally with a bounded key-value cache (Zhuo et al., 2025; Yuan et al., 2026; Wang et al., 2026b). Quantization lowers the

precision of the trained model (Feng et al., 2026; Pan et al., 2026). Most of them hand the full token grid of each frame to the first AA block. PrePARE instead prunes once at the pre-AA interface and restores the full grid with a learned module, so it can run together with in-AA methods (Section 4.4).

Pruning criteria. Criteria for which tokens to keep come from feature similarity, from attention or confidence readouts, or from learned gates and rates. ToMe (Bolya et al., 2023) merges tokens by cosine similarity, and LiteVGGT (Shu et al., 2025) adds a geometry-aware similarity. EViT (Liang et al., 2022) ranks tokens by the attention they receive from the class token, ATS (Fayyaz et al., 2022) samples tokens in proportion to that attention, Co-Me (Chen et al., 2025) keeps tokens by a predicted confidence, and Good Token Hunting (Zheng et al., 2026) scores tokens by attention entropy. DynamicViT (Rao et al., 2021), A-ViT (Yin et al., 2022), and IA-RED^2 (Pan et al., 2021) learn per-token gates or halting, and DiffRate (Chen et al., 2023a) learns the compression rate of each layer. TokenLearner (Ryoo et al., 2021), Evo-ViT (Xu et al., 2022), and SparseViT (Chen et al., 2023b) learn which tokens or windows to compute, and point-cloud transformers keep their accuracy after most of their tokens are merged (Tran et al., 2025). Each of these criteria reads the features or the attention at the place where it reduces. The criterion of PrePARE reads encoder features but learns from the cross-view attention of the unpruned AA stack, which runs after the cut.

Trained and training-free criteria. Training-free methods run on released checkpoints with at most a calibration pass, and they read signals that the running model exposes, such as token similarity (Shen et al., 2026), attention magnitude (Wang et al., 2025a), and head sensitivity (Kim et al., 2026). Trained methods instead learn from targets that exist only offline. Co-Me (Chen et al., 2025) trains a confidence predictor, Speed3R (Ren et al., 2026) trains a compression branch, TurboVGGT (Huang et al., 2026) learns its representative tokens, DiffRate (Chen et al., 2023a) learns its compression rates, and VGGT-Ω (Wang et al., 2026a) and Lite3R (Zhang et al., 2026) train or fine-tune the network itself. PrePARE is a trained method, because its target, the attention of the unpruned AA stack on the training clips, exists only offline, and Table A1 gives its training cost.

3 METHOD

3.1 PRELIMINARY

A multi-view geometry transformer (Wang et al., 2025b; 2026d; Keetha et al., 2026; Lin et al., 2025) consists of an encoder E, an AA stack A of L blocks, and prediction heads H. The encoder maps each of the N input frames to P patch tokens with D dimensions $F_f \in \mathbb{R}^{P\times D}$, and the model adds a small set $\mathcal{T}$ of special tokens, such as camera and register tokens. The blocks of A alternate within-view attention, which runs over the tokens of one frame, with cross-view attention, which runs over the tokens of all frames, $\mathcal{T} \cup \{F_f\}_{f=1}^{N}$. The encoder is usually a separate network, such as DINOv2 (Oquab et al., 2023), and in Depth Anything 3 it is the within-view prefix of a single ViT. Dense heads read each frame's patch tokens at the last block, or at several tapped blocks for DPT-style heads (Ranftl et al., 2021), and heads read a special token or pool the patch tokens.

Within-view attention costs $O(NP^2d)$ per layer and cross-view attention costs $O(N^2P^2d)$, where d is the width of A, and the cross-view term grows quadratically with N. The timing plots in Figure 1a show that global attention, the cross-view attention of VGGT, takes most of the forward time at every length from 100 to 1000 frames. A token-reduction method adds the *decision cost* about deciding which tokens to reduce, and *applying cost* about the cost of prune step itself. Table 1 gives the share of the forward pass that each method spends on each. FastVGGT recomputes a bipartite token matching in each of the 24 global-attention layers of VGGT (Shen et al., 2026), and LiteVGGT computes it at 4 of the 24 and reuses it (Shu et al., 2025), but both merge before and unmerge after every global attention, take 10% and 6.9% of total cost of the forward pass. However, PrePARE decides once at the pre-AA interface, and its scoring and routing, which include the merge, take 5.4% in total of the forward pass. A reduction made once before A pays each of the two costs once, and all L blocks, with their attention, projections, and MLPs, run on the reduced tokens. With a keep ratio r, both attention terms fall by r^2 and the linear layers by r in every block. The pre-AA interface is therefore where a single reduction reaches every block of A at the lowest total overhead in Table 1, but pruning

Table 1: Reduction overhead on VGGT.

			Share (%)	
Method	**Decisions**	**Reduced in**	**Decide**	**Apply**
FastVGGT	24	global attn.	6.0	4.0
LiteVGGT	4	global attn.	1.5	5.4
PrePARE	1	all L blocks	0.4	5.0

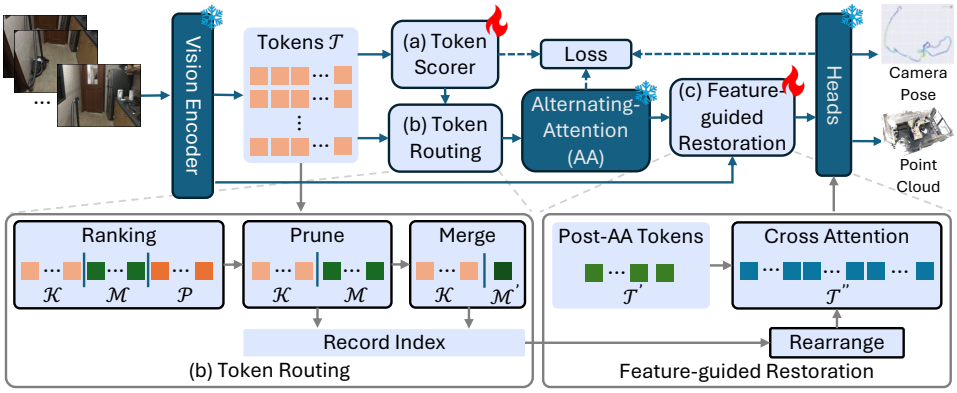


Figure 2: Overview of PrePARE. (a) The Token Scorer scores each encoder patch token. (b) Token Routing splits the tokens into keep, merge, and prune sets $\mathcal{K}$, $\mathcal{M}$, $\mathcal{P}$ under keep ratio r and merges $\mathcal{M}$ into $\mathcal{K}$. (c) Feature-guided Restoration rebuilds the dense grid after the frozen AA stack.

there must meet two requirements: the dense heads still need the full patch grid, and the tokens must be chosen before any cross-view attention has run.

3.2 PrePARE Framework

Motivatied by observation of Section 3.1, we design PrePARE framework, shows in Figure 2. The patch embedding, the AA stack, and the heads stay frozen, and only the (a) Token Scorer and the (c) Feature-guided Restoration modules are trained. A rule-based (b) Token Routing module sits at the pre-AA interface, before the first cross-view attention, and (c) Feature-guided Restoration sits at the post-AA interface, on the patch-token streams that the dense heads read. The pre-AA interface is the encoder output on multi-view geometry transformers.

The *Token Scorer* reads the encoder features F_f of each frame and gives every patch token a score in $[0, 1]$ (Section 3.3). *Token Routing* ranks the tokens of each frame by this score, puts the top $\lceil Pr \rceil$ into the keep set $\mathcal{K}$, where r is the keep ratio, and splits the rest into a merge set $\mathcal{M}$ and a prune set $\mathcal{P}$. It drops $\mathcal{P}$, folds each token of $\mathcal{M}$ into a kept token of the same frame, and records the grid position of every kept token (Section 3.4). Scoring and routing run once for each batch before the first AA block. The frozen AA stack then runs all L blocks on the $N\lceil Pr \rceil$ kept tokens and the $|\mathcal{T}|$ special tokens, which skip routing inside AA. *Feature-guided Restoration* then uses the recorded positions to rearrange the AA outputs by frame and rebuilds the full grid of P tokens per frame with cross-attention (Section 3.4). One restoration module serves each stream that a dense head reads, so each dense head receives a tensor of the same shape as in the unpruned model. The camera head reads the camera token, which skips both routing and restoration.

The Loss block in Figure 2 marks where training reads its targets, from the AA stack and from the heads (Sections 3.3 and 3.5). The same three modules serve all four backbones, and Table A2 lists what each backbone sets: the interface, the special and anchor tokens, and the tapped streams.

3.3 Pruning Criterion

The Token Scorer (in Figure 2-(a)) runs before any cross-view attention and sees only encoder features, so we build its criterion from the AA stack itself. One unpruned forward pass per scene records the cross-view attention of the frozen stack, and the target $S^\star$ combines two readouts of this attention, one for each role a patch token plays in the stack, supporting the pose or scale estimate and matching across views. For patch token i of frame f, after per-frame min-max normalization $\mathrm{Norm}_f(\cdot)$, the target is

$$S^\star(i) \;=\; \alpha \cdot \mathrm{Norm}_f(S_{\text{anchor}}(i)) \;+\; (1-\alpha) \cdot \mathrm{Norm}_f(S_{\text{match}}(i))\,, \qquad \alpha \in [0,1], \tag{1}$$

where S_{anchor} and S_{match} are defined below and α sets their balance.

Let $\mathcal{A}_f$ be the anchor tokens of frame f, the special tokens that a pose or scale head reads. Let $\mathcal{L}_x$ be the set of cross-view blocks and H the number of attention heads per block. The anchor score averages the attention from $\mathcal{A}_f$ to i over these blocks and heads,

$$S_{\text{anchor}}(i) = \frac{1}{|\mathcal{L}_x|\,H\,|\mathcal{A}_f|} \sum_{l\in\mathcal{L}_x} \sum_{h=1}^{H} \sum_{a\in\mathcal{A}_f} A^{l,h}_{a\to i}, \quad S_{\text{match}}(i) = \max_{j\neq i} \frac{1}{|\mathcal{L}_x|\,H} \sum_{l\in\mathcal{L}_x} \sum_{h=1}^{H} A^{l,h}_{i\to j}, \tag{2}$$

where $A^{l,h}_{u\to i}$ is the attention weight from query u to key i at head h of block l, and j runs over the other patch tokens of the scene. A token with a high anchor score is one that the camera token of its frame attends to strongly, so the pose estimate of the unpruned model draws on it. The matching score is high when token i attends to one token elsewhere in the scene much more than to the rest, as a token does when it has a clear match in another view. Figure 3a shows the tokens that $S^\star$ keeps.

The Token Scorer regresses $S^\star$ from each frame's encoder feature map $F_f \in \mathbb{R}^{h\times w\times D}$, with $hw = P$, and runs once per frame. It compares each token with its neighbors through one self-attention layer inside 7×7 windows, and a linear layer with a sigmoid outputs $S \in [0,1]^{h\times w}$ (Section B). The windowed attention keeps the cost linear in P. As a control, we score tokens from the encoder alone with the ToMe merge criterion (Bolya et al., 2023) and use its complement as a distinctiveness score, $S_{\text{red}}(i) = 1 - \max_{j\in f,\, j\neq i} \cos(F_i, F_j)$, where F_i and F_j are the encoder feature vectors of patch tokens i and j in frame f. Section 4.5 compares S_{red} with $S^\star$ under the same routing and restoration.

3.4 Routing and Restoration

A rule-based Token Routing module (in Figure 2-(b)) splits the patch tokens of each frame into a keep set $\mathcal{K}$, a merge set $\mathcal{M}$, and a prune set $\mathcal{P}$ under one keep ratio r and one merge fraction γ. The top $\lceil Pr \rceil$ tokens of each frame by score form $\mathcal{K}$, so the number of tokens that enter A depends only on r. The clip has a merge budget of γNP tokens, and each frame receives a share in proportion to the mean score of its non-kept tokens (Equation (9)). Within a frame, the highest-scoring non-kept tokens fill this share and form $\mathcal{M}$, and the rest form $\mathcal{P}$. Each merge token joins the kept token of its own frame with the most similar encoder feature, and the kept token becomes a score-weighted average of itself and the tokens that join it, with its own weight fixed at 1 (Equation (11)). The same-frame rule keeps the local layout that the dense heads read after restoration, and the score weights carry the gradient of the task losses to the scorer, because top-K selection is not differentiable. Routing records which kept token each merge token joined.

After A, Feature-guided Restoration (in Figure 2-(c)) rebuilds the full grid of P tokens of each frame at every stream t that a dense head reads. It first builds a copy-back grid $B^{t,f}$ from the routing record. A kept position takes its own AA output, a merged position takes the AA output of the kept token it joined, and a pruned position takes an inverse-distance average of the four nearest kept positions in the grid. A cross-attention layer then corrects this grid,

$$G^{t,f}_{\text{dense}} = B^{t,f} + \lambda_{t,c}\,\text{CrossAttn}\Big(W_Q F^f,\; W_K F^f_{\mathcal{K}},\; W^t_V G^{t,f}\Big), \tag{3}$$

where F^f denotes the encoder feature map F_f of all P positions, $F^f_{\mathcal{K}}$ its kept-token subset, and $G^{t,f}$ the kept tokens' AA outputs at tapped stream t. The matrices W_Q, W_K, and W^t_V are learned query, key, and value projections, respectively. The encoder features therefore decide which kept tokens each position reads from. A learned gate $\lambda_{t,c}$ for each stream and each position class $c \in \{\text{kept}, \text{merged}, \text{pruned}\}$ scales the correction, so each class can be corrected by a different amount. Restoration runs frame by frame, so its cost grows linearly with N. Heads that pool patch tokens, such as the camera head of π^3, also read the restored grids, and heads that read a special token read it directly from A, because special tokens skip both routing and restoration.

3.5 Training and Inference

The multi-view geometry transformer stays frozen, and only the Token Scorer and the Feature-guided Restoration module are trained. Both modules are lightweight: together they add 5.3 M trainable parameters on VGGT, 0.42 % of the frozen model, and 2.1 to 7.8 M on the other three backbone models (Table A1). All training signals come from after the pre-AA interface, either from the AA stack of the unpruned model or from the task. Training runs in two stages.

Stage 1 trains the Token Scorer alone to fit the AA-usage target of Equation (1) with a binary cross-entropy averaged over all patch tokens, $\mathcal{L}_{\text{stage1}} = \frac{1}{NP}\sum_{i=1}^{NP} \text{BCE}\big(S_i, S^\star(i)\big)$. Stage 2 starts from this scorer and trains it together with the restoration module under $\mathcal{L}_{\text{stage2}} = \mathcal{L}_{\text{stage1}} + \mathcal{L}_{\text{restore}} + \mathcal{L}_{\text{downstream}}$. We use three terms because each supervises a different part of the pipeline, with $\mathcal{L}_{\text{stage1}}$ choosing which tokens reach the AA stack, $\mathcal{L}_{\text{restore}}$ compares each restored stream with the same stream of the unpruned stack (Equation (4)), and $\mathcal{L}_{\text{downstream}}$ reads the predictions of the heads. It compares the predictions of the pruned model, such as camera poses, depth or point maps, and confidences, with those of the frozen model on the same clip, and it compares the predicted depth

with ground-truth depth (Section B.2). Because top-K selection is not differentiable, $\mathcal{L}_{\text{restore}}$ and $\mathcal{L}_{\text{downstream}}$ reach the scorer only through the merge weights of Section 3.4, and $\mathcal{L}_{\text{stage1}}$ keeps supervising which tokens are kept and pruned. Running distillation first and the joint stage second gives a lower Chamfer distance than either stage alone or a single joint stage (Table 6b).

At inference, one trained model serves every keep ratio. We train at a single r and choose r at inference, which changes only how many tokens routing keeps in each frame (Table A3).

4 Experiments

4.1 Setup

Datasets. We train PrePARE on a 262-scene building-disjoint split of ScanNetv2 (Dai et al., 2017) and evaluate on the ScanNetv2 at $N \in \{100, 300, 500, 1000\}$ frames, zero-shot on strides 3 and 10 of 7-Scenes (Shotton et al., 2013) and NRGBD (Azinović et al., 2022).

Metrics. The main text reports one dense (3D reconstruction) and one sparse (camera pose) output of the frozen heads; depth and point tracking (VGGT only) are in Sections F.2 and F.3. Reconstruction is measured by Chamfer distance (CD) on ScanNetv2 and by accuracy, completeness and normal consistency (Acc, Comp, NC) zero-shot, and pose by ATE, ARE, RPE-rot and RPE-trans. Table 2 also reports time per clip (Time) and peak GPU memory (Mem). Parity with the unpruned model is judged by a two-sided exact paired sign test on the per-scene metric, reported as scenes on which PrePARE is better out of the paired scenes, with its p-value.

Backbones and Baselines. We evaluate PrePARE on four backbones containing AA: VGGT (Wang et al., 2025b), π^3 (Wang et al., 2026d), MapAnything (MA) (Keetha et al., 2026) and Depth Anything 3-Giant (DA3) (Lin et al., 2025), comparing the unpruned models with in-AA methods from their released code at theirs released head-sparsity profiles.

Hyperparameters and Hardware. PrePARE uses $\gamma = 0.30$ and $\alpha = 0.25$ on VGGT and DA3. π^3 and MA have no per-frame camera token and use $\alpha = 0$. The inference keep ratio is $r = 0.40$ on VGGT, 0.6 on π^3, and 0.8 on MA and DA3 (Table A3). Training uses AdamW at learning rates 10^{-5} (scorer) and 2×10^{-4} (restoration) on 2- to 8-frame clips. All runs use one 93 GiB H100 (Section E).

4.2 Quantitative Results

PrePARE uses less peak memory and lower time than in-AA methods, both were measured in Table 2. It also runs VGGT at 500 and 1000 frames where the unpruned models run out of GPU memory. Its lead in camera pose also grows with the clip length: on VGGT, the ATE gap to the best in-AA method, LiteVGGT, widens from 0.012 m at 100 frames (0.095 against 0.107) to 0.025 m at 300 (0.092 against 0.117), and on DA3 the gap to SAF3R widens from 100 to 500 frames. We ascribe it to the matching score. With more frames, each surface appears in more views, so the tokens it keeps have more partners in other views and constrain the pose more tightly. On π^3 the error between consecutive frames of PrePARE reaches that of the unpruned model at 500 frames.

The saving over the unpruned model also grows with the clip length, more from 300 to 500 frames than from 100 to 300: on π^3, PrePARE runs 2.3, 2.6, and 3.3 times faster at 100, 300, and 500 frames and saves 13, 15, and 33 % of peak memory. The cross-view term grows with N^2 and PrePARE shrinks it by r^2 in every block, so the unpruned time rises faster. The part of the peak memory that does not scale with the tokens entering A, such as the model weights, is a large share at 100 frames and a small share at 500, so the smaller token count shows more at longer clips. Beyond 300 frames the memory of PrePARE on VGGT grows linearly, by about 3.2 GiB per 100 frames, and reaches 37.1 GiB at 1000 frames. From 300 frames on, the time gap to Co-Me grows at a nearly constant 2.2 to 2.5 s per 100 frames, and LiteVGGT follows neither pattern, with gaps of 2.1, 15.0, and 7.6 s.

3D reconstruction. The efficiency and performance gain of PrePARE improves as the clip grows compared to other methods. At $N = 100$, it has the best results on VGGT and DA3, second-best behind Co-Me on π^3 (CD+0.0272) and MA (CD+0.0255). At $N = \{300, 500, 1000\}$, PrePARE reach SOTA on all four backbones among all compared methods.

Table 2: **3D reconstruction on ScanNetv2**: Chamfer distance (CD), wall-clock per clip (Time, s) and peak memory (Mem, GiB) at N frames; first row per block: unpruned. OOM: Out of GPU Memory.

	100			300			500			1000		
Method	**CD↓**	**Time↓**	**Mem↓**	**CD↓**	**Time↓**	**Mem↓**	**CD↓**	**Time↓**	**Mem↓**	**CD↓**	**Time↓**	**Mem↓**
VGGT	0.4406	5.4	27.2	0.4512	48.9	72.3	OOM	OOM	OOM	OOM	OOM	OOM
FastVGGT	0.4473	3.0	8.3	0.4508	15.0	20.7	0.5014	35.4	32.5	0.4954	99.4	59.8
Co-Me	0.4590	1.9	9.4	0.4258	12.4	23.0	0.4983	29.2	34.7	0.4955	77.8	61.2
LiteVGGT	0.4360	1.7	7.0	0.4589	11.2	16.5	0.4897	35.9	25.8	0.4909	66.0	47.6
PrePARE	**0.4185**	**1.6**	**6.5**	**0.4187**	**9.1**	**14.4**	**0.4761**	**20.9**	**20.9**	**0.4883**	**58.4**	**37.1**
π^3	0.4161	11.4	7.8	0.4301	35.9	15.7	0.4376	146.7	29.5	0.4306	365.7	43.9
Co-Me	**0.3956**	7.1	7.3	0.4539	22.2	15.3	0.4327	87.9	23.4	0.4489	340.9	43.6
SAF3R	0.4301	5.8	7.6	0.4320	14.9	14.3	0.4523	65.4	23.8	0.4849	173.5	35.9
PrePARE	0.4228	**5.0**	**6.8**	**0.4203**	**13.9**	**13.3**	**0.4257**	**45.0**	**19.8**	**0.4468**	**159.4**	**35.2**
Depth Anything 3	0.3882	9.2	17.6	0.3929	35.4	40.2	0.4094	92.6	71.3	OOM	OOM	OOM
Co-Me	0.3962	8.5	17.0	0.4327	20.9	37.6	0.3797	98.8	58.2	OOM	OOM	OOM
SAF3R	0.3974	7.5	17.3	0.4187	14.3	43.0	0.4009	99.1	63.6	OOM	OOM	OOM
PrePARE	**0.3900**	**7.2**	**14.2**	**0.3924**	**9.7**	**32.3**	**0.3588**	**63.5**	**50.4**	**0.3721**	**176.2**	**88.9**
MapAnything	0.3995	10.0	52.9	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
Co-Me	**0.3940**	7.7	36.4	0.4705	15.6	63.0	0.4820	75.8	89.7	OOM	OOM	OOM
PrePARE	0.4195	**5.1**	**16.0**	**0.4236**	**13.5**	**27.5**	**0.4383**	**63.7**	**42.8**	**0.4680**	**226.8**	**78.4**

Table 3: **Camera pose estimation on ScanNetv2** at N frames: ATE (m) and ARE (degrees) over the trajectory, rotation and translation RPE_r (degrees) and RPE_t (m) between consecutive frames, RMSE after Umeyama alignment with scale. OOM: Out of GPU Memory.

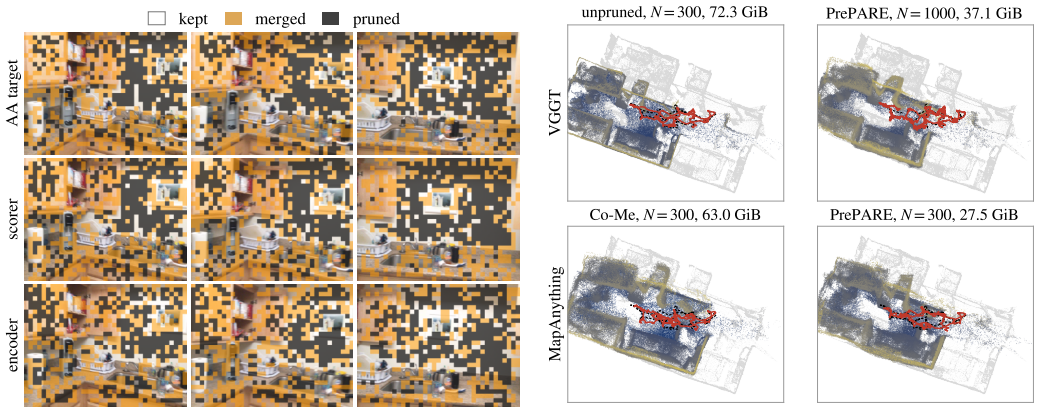
	N=100				N=300				N=500			
Method	**ATE↓**	**ARE↓**	**RPE_r↓**	**RPE_t↓**	**ATE↓**	**ARE↓**	**RPE_r↓**	**RPE_t↓**	**ATE↓**	**ARE↓**	**RPE_r↓**	**RPE_t↓**
VGGT	0.092	3.92	1.16	0.055	0.096	4.17	0.74	0.039	OOM	OOM	OOM	OOM
FastVGGT	0.118	4.30	1.71	0.086	0.123	4.58	1.13	0.059	0.129	4.72	1.05	0.054
Co-Me	0.132	4.68	1.94	0.091	0.127	4.91	1.36	0.070	0.134	4.98	1.24	0.063
LiteVGGT	0.107	4.06	1.42	0.072	0.117	4.46	0.95	0.048	0.122	4.63	0.91	0.046
PrePARE	**0.095**	**3.82**	**1.28**	**0.061**	**0.092**	**3.99**	**0.75**	**0.036**	**0.098**	**4.13**	**0.69**	**0.033**
π^3	0.054	2.36	0.84	0.035	0.053	2.48	0.49	0.020	0.054	2.41	0.40	0.017
Co-Me	0.062	2.63	1.08	0.046	0.061	2.76	0.70	0.031	0.061	2.80	0.60	0.027
SAF3R	0.081	3.51	1.28	0.054	0.079	3.80	0.90	0.033	0.079	3.46	0.80	0.028
PrePARE	**0.060**	**2.51**	**0.86**	**0.037**	**0.057**	**2.55**	**0.50**	**0.021**	**0.059**	**2.58**	**0.40**	**0.017**
Depth Anything 3	0.084	4.05	1.19	0.044	0.083	4.24	0.86	0.028	0.084	4.19	0.84	0.025
Co-Me	0.094	4.24	1.23	0.050	0.097	4.55	1.15	0.046	0.108	4.41	1.12	0.027
SAF3R	0.088	4.19	1.40	0.049	0.088	4.37	0.98	0.031	0.088	4.39	1.05	0.027
PrePARE	**0.086**	**4.05**	**1.19**	**0.046**	**0.085**	**4.32**	**0.92**	**0.029**	**0.081**	**4.20**	**0.92**	**0.026**
MapAnything	0.127	5.58	1.73	0.080	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
Co-Me	0.241	11.03	2.66	0.120	0.228	11.65	1.88	0.086	0.236	11.42	1.79	0.082
PrePARE	**0.219**	**8.16**	**2.46**	**0.118**	**0.211**	**9.39**	**1.52**	**0.068**	**0.218**	**9.07**	**1.64**	**0.061**

PrePARE removes the prune set before the AA stack, whereas LiteVGGT and Co-Me merge tokens, so every merged token still contributes to an average. A 100-frame clip has fewer overlapping views of each surface than a longer clip of the same scene, so a pruned token more often holds content that no kept token in another view repeats, and restoration has to fill its position from its spatial neighbors. A longer clip shows each surface in more views, so a pruned token more often has a kept counterpart in another view that cross-view attention can read. The CD of PrePARE holds at 0.4185 and 0.4187 and that of Co-Me, falls from 0.4590 to 0.4258, while those of the unpruned model, FastVGGT, and LiteVGGT rise.

Camera pose. PrePARE reach the SOTA among compared methods on four backbone models Table 3. We attribute this lead to the way PrePARE chooses tokens, which follows the camera by design. Both readouts of its target come from the attention that estimates the pose. The anchor score keeps the tokens that the camera token reads, and the matching score keeps the tokens that link views, which constrain the relative pose. The camera token skips routing and restoration, so the pose head reads it from the AA stack as in the unpruned model. Compared methods instead choose tokens by feature similarity or by the confidence of the dense outputs, or sparsifies attention heads, so none of them reads the attention of the camera token to decide what to reduce. The effect is largest where the camera enters the target directly. On VGGT and DA3, whose anchor score reads a per-frame camera token, PrePARE matches or improves on the unpruned model in several settings, while on π^3 and MA, which use the matching score alone, it stays behind the unpruned model in trajectory error.

4.3 Qualitative Results

In Figure 3a the target and the learned scorer keep the dish rack and the picture frame and prune much of the bare wall, while the encoder control merges most of the rack; over one clip per scene,



(a) Token selection at $r = 0.4$ (b) Long sequences

Figure 3: **Qualitative results on ScanNetv2.** (a) Kept, merged and pruned tokens on three frames of one 4-frame clip, routed from the AA-usage target, the learned scorer and the encoder-similarity control. (b) Top-down reconstruction of one scene over the ground-truth mesh (gray), with the estimated camera path (red) over ground truth (dashed).

Table 4: **Zero-shot 3D reconstruction**, ScanNetv2-trained modules at $r = 0.40$.

	7-Scenes Stride 3			7-Scenes Stride 10			NRGBD Stride 3			NRGBD Stride 10		
Method	**Acc↓**	**Comp↓**	**NC↑**	**Acc↓**	**Comp↓**	**NC↑**	**Acc↓**	**Comp↓**	**NC↑**	**Acc↓**	**Comp↓**	**NC↑**
VGGT	0.0191	0.0280	0.675	0.0191	0.0291	0.690	0.0330	0.0172	0.644	0.0139	0.0149	0.907
Co-Me	0.0206	0.0291	0.601	0.0370	0.0316	0.621	0.0319	0.0193	0.659	0.0597	0.0274	0.706
LiteVGGT	0.0382	0.0279	0.591	0.0204	0.0280	0.612	0.0329	0.0189	0.670	0.0217	0.0182	0.672
PrePARE	**0.0179**	**0.0243**	**0.628**	**0.0175**	**0.0276**	**0.651**	**0.0317**	**0.0171**	**0.678**	**0.0189**	**0.0161**	**0.777**

the learned keep set shares 73 % of the target's tokens and the control's 55 % (40 % by chance; Section F.4). On the scene of Figure 3b, PrePARE on VGGT at $N = 1000$ reaches the CD of the unpruned model at $N = 300$ (0.553) in half its memory, and on MA Co-Me reconstructs better than PrePARE (0.433 against 0.497) in 2.3 times the memory.

In Figure 3b the estimated camera paths stay close to the ground truth in all four panels. On this scene ATE is 0.062 m for the unpruned VGGT and 0.130 and 0.143 m for PrePARE and Co-Me on MA.

4.4 Discussion

Zero-shot transfer of the trained modules. PrePARE trains only once per backbone model, on short ScanNetv2 clips and in 2.2 to 4.0 GPU-hours (Table A1). Compared training-needed methods follows the same training protocal of PrePARE. We eveluate the ScanNetv2-trained modules on 7-Scenes and NRGBD without retraining. Table 4 reports reconstruction on all 7-Scenes and NRGBD clips at strides 3 and 10 for the VGGT instance at $r = 0.40$; against the other efficiency approaches, PrePARE is more accurate and higher in NC at both strides on both datasets compared to SOTAs.

Combining with in-AA methods. Table 5 demonstrates that combining pre-AA and in-AA methods without retraining lowers the cost with errors raised. Across the four compositions, the time is 8.0 to 10.2 s and the memory 12.1 to 16.2 GiB, below every in-AA setting in the table, while the CD rises by 0.017 to 0.023 over the unpruned model's 0.4512 and the ATE rises from 0.096 to between 0.121 and 0.199. The sign test against the unpruned model separates none of the composed models, the same outcome as for both in-AA methods at their milder merge, and the best composed model keeps the ATE of the in-AA methods at their default merge. The degrades come from stacking two reductions that choose tokens in different ways. in-AA merger works on precut token sequences and merges them by feature similarity alone, so the second cut removes content that the heads depend on, thus the pose error rises more than the CD. To close the remaining gap, training the scorer and the restoration module with the merger in the stack, or letting the scorer tell the merger which tokens to keep, is a direct way to keep the lowest time and memory, which we leave to future work.

4.5 Ablations

Lowering the keep ratio in Table 6a removes most of the cost at the first step, to $r = 0.6$, where peak memory falls from 72.3 to 17.8 GiB and time from 48.9 to 15.8 s, while the CD stays at parity

Table 5: **Composition with an in-AA merger**, ScanNetv2, $N = 300$: sign test against the unpruned model (better scenes, p); composed rows and their matched baseline route the first frame.

Method	CD↓	ATE↓	ARE↓	RPE_r↓	RPE_t↓	Time↓	Mem↓	Sign test
VGGT	0.4512	0.096	4.17	0.74	0.039	48.9	72.3	–
LiteVGGT r=0.9	0.4589	0.117	4.46	0.95	0.048	11.2	16.5	22/50, 0.48
FastVGGT r=0.9	0.4508	0.123	4.58	1.13	0.059	15.0	20.7	24/50, 0.89
LiteVGGT r=0.6	0.4258	0.117	4.11	0.79	0.034	10.3	29.9	31/50, 0.12
FastVGGT r=0.6	0.4501	0.101	4.02	0.78	0.037	19.4	34.1	20/50, 0.20
PrePARE r=0.4 + LiteVGGT r=0.9	0.4726	0.150	6.19	1.54	0.072	8.0	12.1	21/50, 0.32
PrePARE r=0.4 + FastVGGT r=0.9	0.4744	0.199	7.73	2.38	0.096	9.4	14.4	20/50, 0.20
PrePARE r=0.2 + FastVGGT r=0.8	0.4685	0.137	5.76	1.41	0.051	10.2	16.2	22/50, 0.48
PrePARE r=0.3 + FastVGGT r=0.7	0.4681	0.121	5.73	1.31	0.064	9.7	15.8	21/50, 0.32

Table 6: **Ablations** on VGGT, ScanNetv2, $N = 300$: Time (s), Mem (GiB), sign test vs. unpruned (better scenes, p). Panel (b) is CD only, unpruned 0.4512 Section G.

(a) Variant	CD↓	ATE↓	Time	Mem	Sign test
Keep ratio ($N = 300$; default scorer, learned restoration, first frame protected)					
unpruned	0.4512	0.096	48.9	72.3	–
$r = 0.6$	0.4186	0.109	15.8	17.8	26/50, 0.89
$r = 0.4$	0.4187	0.092	9.1	14.4	34/50, 0.015
$r = 0.2$	0.4379	0.101	8.0	10.5	31/50, 0.12
Restoration and training clips, against the $r = 0.4$ default above					
ToMe-style unmerging	0.5229	0.122	11.2	63.6	16/50, 0.02
wide-stride training clips (WS)	0.4867	0.120	11.4	14.4	16/50, 0.02

(b) Factor	Variant and CD↓ (default in bold)			
Restoration	zero fill 0.5136	bilinear 0.4762	**learned 0.4187**	
Merge fraction γ	0: 0.4630	**0.30: 0.4187**	0.60: 0.4421	1: 0.4537
Blend α	0: 0.4742	**0.25: 0.4187**	0.75: 0.4394	1: 0.4986
Schedule	stage 1 only 0.4851	stage 2 only 0.4946	joint 0.4759	**two-stage 0.4187**
Criterion	encoder cosine 0.5193	$S^\star$ oracle 0.4668	**learned 0.4187**	

at $r = 0.6$ and falls below the unpruned model's at $r = 0.4$, where the pruned model beats the unpruned one on 34 of 50 scenes ($p = 0.015$). Going further to $r = 0.2$ saves little time (9.1 to 8.0 s), most likely because the encoder, the restoration and the heads still run on the full grid, and it raises the mean CD to 0.4379 even though 31 of 50 scenes beat the unpruned model, so the extra error is concentrated in a few scenes and $r = 0.4$ stays the default.

At the same keep ratio, ToMe-style unmerging loses parity (16 of 50 scenes better, $p = 0.02$) and pushes peak memory back to 63.6 GiB, close to the unpruned 72.3 GiB, so the memory saving depends on how the grid is rebuilt, not only on how many tokens enter the AA stack. Training on wide-stride clips (WS) keeps the peak memory of the default at 14.4 GiB but loses parity by the same margin, so the spacing of the training frames alone can cost parity at the same token budget.

Table 6b shows that the structural choices matter more than the tuning knobs: every alternative restoration, schedule and criterion ends above the unpruned 0.4512, whereas γ and α stay below it except at their extremes. The gap that bilinear interpolation leaves to the learned restoration suggests that a pruned position needs features chosen by content, and the failing extremes suggest that merging and pruning do different jobs and that the target needs both its matching and its anchor score.

The encoder-cosine rule reads the same encoder features as the learned scorer but gives the worst CD in panel (b), worse than zero filling, while routing by the AA-usage target $S^\star$ itself lands between the two, so what the scorer learns to predict matters more than what it reads. The learned scorer also beats the target it was trained on, while stage 2 alone is the worst schedule, which suggests that the task losses can refine the selection that the target sets up but cannot find one on their own.

5 Conclusion

We presented PrePARE, which prunes patch tokens once before the AA stack of a frozen multi-view geometry transformer and restores the dense grid once after it, so every AA block runs on fewer tokens. Its Token Scorer learns from how the unpruned AA stack uses each token. On ScanNetv2 at $N = 300$, it lowers the peak memory of VGGT from 72.3 to 14.4 GiB and runs 5.4 times faster without raising the Chamfer distance. On π^3 and DA3 it saves 15 and 20 %, where the in-AA methods save at most 9 %. At 1000 frames, only PrePARE fits DA3 and MapAnything on one GPU.

REPRODUCIBILITY STATEMENT

The code of this project will be released upon acceptance. The evaluation follows the FastVGGT protocol (Shen et al., 2026): ScanNetv2 (Dai et al., 2017) at $N \in \{100, 300, 500, 1000\}$ and, zero-shot from the ScanNetv2-trained modules, 7-Scenes (Shotton et al., 2013) and NRGBD (Azinović et al., 2022) at strides 3 and 10. Metric definitions, the paired-test protocol, and the peak-memory and timing protocols are given in Section E; the per-scene statistics behind the main-text tables are in Table A3; per-backbone interfaces, anchor tokens and loss terms are in Table A2 and Section D; training hyperparameters and cost are in Table A1. All measurements use one NVIDIA H100 NVL (93 GiB usable) in FP16 with Flash-Attention 2 (Dao, 2024). Training uses seed 42 with no seed replicates, evaluation point sampling seed 33, and bootstrap seed 0 with 10,000 resamples. The backbone models are the public checkpoints of VGGT (Wang et al., 2025b), π^3 (Wang et al., 2026d), MapAnything (Keetha et al., 2026) in images-only mode, and Depth Anything 3 (Lin et al., 2025) (giant), and the baselines run from their released code at the settings their papers recommend.

ETHICS STATEMENT

PrePARE accelerates publicly released feed-forward 3D reconstruction models, adds 5.3 M trained parameters on VGGT (Table A1 lists the other backbone models), and collects no new data. Training uses ScanNetv2 train scenes and evaluation uses ScanNetv2, 7-Scenes and NRGBD under their published terms, each of which permits non-commercial research. The backbone and baseline checkpoints carry mixed licenses (Meta's research license, BSD, Apache 2.0, MIT, and CC BY-NC 4.0); the Co-Me and Meta terms are non-commercial, so a release of this wrapper inherits that restriction for the configurations built on them. The societal profile of the wrapped model is that of its backbone model. Lower peak memory reduces the compute and energy cost of multi-view reconstruction and allows longer clips on a single GPU, and the same capability shares the dual-use profile of the backbone models, since faster reconstruction of a scene from images can also serve surveillance or mapping of spaces without the consent of the people in them.

AI USE STATEMENT

Large language models were used to help with English writing and editing the manuscript, with systematic literature review before this project, and with parts of the evaluation and analysis scripts.

REFERENCES

Dejan Azinović, Ricardo Martin-Brualla, Dan B. Goldman, Matthias Nießner, and Justus Thies. Neural RGB-D surface reconstruction. In *CVPR*, 2022.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your ViT but faster. In *ICLR*, 2023.

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 (ICCV)*, 2023a.

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 (CVPR)*, pp. 2061–2070, 2023b.

Yutian Chen, Yuheng Qiu, Ruogu Li, Ali Agha, Shayegan Omidshafiei, Jay Patrikar, and Sebastian Scherer. Co-me: Confidence-guided token merging for visual geometric transformers, 2025. URL https://arxiv.org/abs/2511.14751.

Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNetv2: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, 2017.

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024.

Jianing Deng, Yuanzhe Li, Jialu Wang, Song Wang, Tianlong Chen, Huanrui Yang, and Jingtong Hu. Saf3r: Dynamic sparse attention for feed-forward 3d reconstruction transformers. *arXiv preprint arXiv:2607.03612*, 2026a.

Kai Deng, Zexin Ti, Jiawei Xu, Jian Yang, and Jin Xie. VGGT-Long: Chunk it, loop it, align it – pushing vggt's limits on kilometer-scale long rgb sequences. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2026b. arXiv:2507.16443.

Carl Doersch, Ankush Gupta, Larisa Markeeva, Adrià Recasens, Lucas Smaira, Yusuf Aytar, João Carreira, Andrew Zisserman, and Yi Yang. TAP-Vid: A benchmark for tracking any point in a video. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2022.

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

Weilun Feng, Haotong Qin, Mingqiang Wu, Chuanguang Yang, Yuqi Li, Xiangqi Li, Zhulin An, Libo Huang, Yulun Zhang, Michele Magno, and Yongjun Xu. Quantized visual geometry grounded transformer. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2509.21302.

David Huang, Guile Wu, Chengjie Huang, Bingbing Liu, and Dongfeng Bai. TurboVGGT: Fast visual geometry reconstruction with adaptive alternating attention. *arXiv preprint arXiv:2605.14315*, 2026.

Nikhil Keetha, Norman Müller, Johannes Schönberger, Lorenzo Porzi, Yuchen Zhang, Tobias Fischer, Arno Knapitsch, Duncan Zauss, Ethan Weber, Nelson Antunes, Jonathon Luiten, Manuel Lopez-Antequera, Samuel Rota Bulò, Christian Richardt, Deva Ramanan, Sebastian Scherer, and Peter Kontschieder. MapAnything: Universal feed-forward metric 3D reconstruction. In *International Conference on 3D Vision (3DV)*, 2026. arXiv:2509.13414.

Yongsung Kim, Wooseok Song, Jaihyun Lew, Hun Hwangbo, Jaehoon Lee, and Sungroh Yoon. HeSS: Head sensitivity score for sparsity redistribution in VGGT. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026. arXiv:2603.25336.

Xinze Li, Pengxu Chen, Yiyuan Wang, Weifeng Su, and Wentao Cheng. S-VGGT: Structure-aware subscene decomposition for scalable 3D foundation models. In *IEEE International Conference on Multimedia and Expo (ICME)*, 2026. arXiv:2603.17625.

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

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

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Bowen Pan, Rameswar Panda, Yifan Jiang, Zhangyang Wang, Rogerio Feris, and Aude Oliva. IA-RED^2: Interpretability-aware redundancy reduction for vision transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 24898–24911, 2021.

Zhizhen Pan, Hesong Wang, and Huan Wang. QVGGT: Post-training quantized visual geometry grounded transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026. arXiv:2605.31124.

Jinsoo Park, Donggyu Choi, Ahyun Seo, Minsu Cho, and Jeany Son. Diversity-aware view partitioning for scalable VGGT. In *European Conference on Computer Vision (ECCV)*, 2026. arXiv:2607.01885.

René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *ICCV*, 2021.

Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. DynamicViT: Efficient vision transformers with dynamic token sparsification. In *NeurIPS*, 2021.

Weining Ren, Xiao Tan, and Kai Han. Speed3R: Sparse feed-forward 3D reconstruction models. In *CVPR Findings*, 2026. arXiv:2603.08055.

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

You Shen, Zhipeng Zhang, Yansong Qu, Xiawu Zheng, Jiayi Ji, Shengchuan Zhang, and Liujuan Cao. Fastvggt: Fast visual geometry transformer. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://openreview.net/forum?id=asl8NJlIMe.

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 *CVPR*, 2013.

Zhijian Shu, Cheng Lin, Tao Xie, Wei Yin, Ben Li, Zhiyuan Pu, Weize Li, Yao Yao, Xun Cao, Xiaoyang Guo, and Xiao-Xiao Long. Litevggt: Boosting vanilla vggt via geometry-aware cached token merging. *arXiv preprint arXiv:2512.04939*, 2025.

Zecheng Tang, Jiaye Fu, Qiankun Gao, Haijie Li, Yanmin Wu, Jiaqi Zhang, Siwei Ma, and Jian Zhang. Spark3R: Asymmetric token reduction makes fast feed-forward 3D reconstruction. *arXiv preprint arXiv:2605.06270*, 2026.

Tuan Anh Tran, Duy M. H. Nguyen, Hoai-Chau Tran, Michael Barz, Khoa D. Doan, Roger Wattenhofer, Ngo Anh Vien, Mathias Niepert, Daniel Sonntag, and Paul Swoboda. How many tokens do 3D point cloud transformer architectures really need? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Chung-Shien Brian Wang, Christian Schmidt, Jens Piekenbrinck, and Bastian Leibe. Block-sparse global attention for efficient multi-view geometry transformers. *arXiv preprint arXiv:2509.07120*, 2025a.

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025b.

Jianyuan Wang, Minghao Chen, Shangzhan Zhang, Nikita Karaev, Johannes Schönberger, Patrick Labatut, Piotr Bojanowski, David Novotny, Andrea Vedaldi, and Christian Rupprecht. VGGT-Ω. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026a. arXiv:2605.15195.

Runze Wang, Yuxuan Song, Youcheng Cai, and Ligang Liu. STAC: Plug-and-play spatio-temporal aware cache compression for streaming 3D reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026b. arXiv:2603.20284.

Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jérôme Revaud. DUSt3R: Geometric 3d vision made easy. In *CVPR*, 2024.

Weitian Wang, Lukas Meiner, Rai Shubham, Cecilia De La Parra, and Akash Kumar. HTTM: Head-wise temporal token merging for faster VGGT. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026c. arXiv:2511.21317.

Yifan Wang, Jianjun Zhou, Haoyi Zhu, Wenzheng Chang, Yang Zhou, Zizun Li, Junyi Chen, Jiangmiao Pang, Chunhua Shen, and Tong He. π^3: Permutation-equivariant visual geometry learning. In *International Conference on Learning Representations (ICLR)*, 2026d. URL https://openreview.net/forum?id=DTQIjngDta.

Zipeng Wang and Dan Xu. FlashVGGT: Efficient and scalable visual geometry transformers with compressed descriptor attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026. arXiv:2512.01540.

Yifan Xu, Zhijie Zhang, Mengdan Zhang, Kekai Sheng, Ke Li, Weiming Dong, Liqing Zhang, Changsheng Xu, and Xing Sun. Evo-ViT: Slow-fast token evolution for dynamic vision transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 2964–2972, 2022.

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

Jinhao You, Shuo Lyu, Zhuohang Lyu, Tanxuan Li, Zibo Zhao, Jiaxiang Hu, Kai Tang, and Yichen Guo. RegimeVGGT: Layer-wise spatially preserving redundancy removal for visual geometry grounded transformer. *arXiv preprint arXiv:2606.18439*, 2026.

Shuai Yuan, Yantai Yang, Xiaotian Yang, Xupeng Zhang, Zhonghao Zhao, Lingming Zhang, and Zhipeng Zhang. InfiniteVGGT: Visual geometry grounded transformer for endless streams. *arXiv preprint arXiv:2601.02281*, 2026.

Haoyu Zhang, Zeyu Zhang, Zedong Zhou, Yang Zhao, and Hao Tang. Lite3R: A model-agnostic framework for efficient feed-forward 3D reconstruction. *arXiv preprint arXiv:2605.11354*, 2026.

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

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

A Training Configuration and Cost

Stage 1 trains the Token Scorer at a constant learning rate of 1×10^{-5}. Stage 2 trains the scorer and the restoration modules under the objective of Section 3.5, with the two AdamW learning rates of Section 4.1, 100 warm-up steps followed by a cosine schedule that decays to 0.1 of the peak rate, weight decay 0.05, and gradient clipping at 1.0. Table A1 lists the remaining settings and the cost of each stage.

The modules are trained on clips of 2 to 8 frames on VGGT and 2 to 4 frames on the other three backbone models, and they are evaluated on clips of 100 to 1000 frames.

B Token Scorer and Training Losses

AA-usage target and Token Scorer. The target $S^\star$ of Equation (1) is computed once from the unpruned AA stack on each training clip and cached before stage 1. The Token Scorer projects each encoder token to width 256 with a linear layer and applies one 4-head self-attention layer within each non-overlapping 7×7 window of the patch grid. The grid is padded to a multiple of 7, and the padded positions are masked. A residual connection and layer normalization follow, and a linear layer with a sigmoid outputs the score. The scorer has 0.53 M parameters at $D = 1024$ and 0.66 M at $D = 1536$ (Table A1).

Table A1: Training configuration and cost per backbone model. Peak memory is measured during training. Trainable parameters are the Token Scorer plus one restoration module per tapped stream.

	VGGT	π^3	**DA3**	**MapAnything**
Epochs (stage 1 / stage 2)	19 / 8	19 / 12	19 / 12	19 / 12
Optimizer steps (stage 1 / 2)	399 / 664	190 / 516	190 / 516	190 / 516
Clips per optimizer step	24	24	24	24
Training clip length (frames)	2–8	2–4	2–4	2–4
Precision	fp16	bf16	bf16	bf16
GPU-hours (stage 1 / stage 2)	0.11 / 3.84	0.19 / 2.11	0.16 / 2.37	0.15 / 2.04
Peak training memory (GiB)	8.4 / 32.3	6.6 / 23.9	9.2 / 28.9	9.1 / 33.6
Token Scorer params	0.53 M	0.53 M	0.66 M	0.66 M
Restoration params (taps)	4.74 M (4)	1.58 M (1)	7.12 M (4)	3.16 M (3)
Trainable total	**5.27 M**	**2.11 M**	**7.77 M**	**3.82 M**
Frozen backbone params	1257 M	959 M	1356 M	1228 M
Trainable share	0.42 %	0.22 %	0.57 %	0.31 %

B.1 Restoration loss

Let $\mathcal{T}_{\text{tap}}$ be the set of tapped streams, G^t_{dense} the restored output at tap t, and G^t_{full} the output of the unpruned model at the same tap. The restoration loss is the mean squared error between the two, divided by the variance of G^t_{full}, which receives no gradient, and averaged over the taps:

$$\mathcal{L}_{\text{restore}} = \frac{1}{|\mathcal{T}_{\text{tap}}|} \sum_{t \in \mathcal{T}_{\text{tap}}} \frac{\text{MSE}(G^t_{\text{dense}}, G^t_{\text{full}})}{\text{Var}[G^t_{\text{full}}]}. \tag{4}$$

The number of tapped streams is set by the dense head of each backbone model: four on VGGT and Depth Anything 3, three on MapAnything, and one on π^3 (Section D).

B.2 Task terms of stage 2

The term $\mathcal{L}_{\text{downstream}}$ of Section 3.5 is the sum of a prediction-consistency term and three ground-truth depth terms, $\mathcal{L}_{\text{downstream}} = \mathcal{L}_{\text{consist}} + \mathcal{L}_{\text{si}} + \mathcal{L}_{\text{tail}} + \mathcal{L}_{\text{range}}$, each with weight 1.0 on all four backbone models.

Prediction consistency. $\mathcal{L}_{\text{consist}}$ compares the outputs of the pruned model with those of the frozen unpruned model on the same clip:

- VGGT (Wang et al., 2025b): the camera pose encoding (Huber), the depth and world-point maps (after scale-and-shift alignment), and their two confidence maps (relative Huber).
- π^3 (Wang et al., 2026d): the camera rotation (Huber) and translation (relative Huber), the local point maps (after scale-and-shift alignment), and the confidence map (relative Huber).
- MapAnything (Keetha et al., 2026): the 3D point maps (after scale-and-shift alignment), the camera rotation from sign-aligned quaternions, the camera translation, and the confidence map.
- Depth Anything 3 (Lin et al., 2025): the camera rotation and translation from the world-to-camera extrinsics, the depth maps (after scale-and-shift alignment), and the intrinsics (relative Huber, in pixels).

The dense terms exclude pixels at which the prediction of the frozen model is an outlier, with a threshold of 0.5, and the pose terms are not masked.

Ground-truth depth. Three terms compare the predicted depth with the ScanNetv2 ground truth after one alignment offset shared by the whole clip. Let D_{gt} be the ground-truth depth map, $d = \log D_{\text{gt}}$

at valid pixels, and $\tilde{d}$ the aligned log prediction. Then

$$\mathcal{L}_{\text{si}} = \text{scale-invariant loss on } (\tilde{d} - d), \tag{5}$$

$$\mathcal{L}_{\text{tail}} = \text{mean}_{\text{worst 1\%}} \ |\tilde{d} - d| \text{ per frame}, \tag{6}$$

$$\mathcal{L}_{\text{range}} = \text{mean relu}\Big(\tilde{d} - \log(1.5 \cdot \max_{\text{valid}} D_{\text{gt}})\Big). \tag{7}$$

The tail term averages the worst 1 % of pixels in each frame, because the per-scene error on this benchmark is dominated by a few badly predicted regions. The range term penalizes depth predicted beyond 1.5 times the farthest valid ground truth. It is averaged over all pixels, including those without ground truth, because restored regions that are predicted far outside the scene tend to lie where the ground truth has holes.

VGGT ground-truth objective. On VGGT, $\mathcal{L}_{\text{downstream}}$ also includes, with weight 0.1, VGGT's own training objective on camera, depth and point map. Each part is an aleatoric regression $\|\Sigma \odot (\hat{X} - X)\| + \|\Sigma \odot (\nabla\hat{X} - \nabla X)\| - \beta_{\text{unc}} \log \Sigma$ with $\beta_{\text{unc}} = 0.2$, following DUSt3R (Wang et al., 2024), where $\hat{X}$ and X are the predicted and ground-truth quantities, Σ is the predicted confidence, ∇ takes spatial gradients, and $\odot$ is elementwise multiplication. The other three backbone models use the consistency and depth terms only. No backbone model is trained with a tracking loss, and the VGGT tracking head is evaluated with the modules trained as above (Section F.3).

C Token Routing

Let $\mathcal{K}_f$ be the $K_f = \lceil Pr \rceil$ kept tokens of frame f (Section 3.4) and $\mathcal{N}_f = \{i \in f : i \notin \mathcal{K}_f\}$ its other patch tokens. The mean score of the non-kept tokens,

$$\phi_f \ = \ \frac{1}{|\mathcal{N}_f|} \sum_{i \in \mathcal{N}_f} S_i, \tag{8}$$

measures how much scored content a frame has outside its keep set. The clip-level merge budget $B_M = \min\Big(\lfloor \gamma N P \rfloor, \sum_f |\mathcal{N}_f|\Big)$ is split across frames in proportion to ϕ_f,

$$M_f \ = \ \text{round}\left(B_M \cdot \frac{\phi_f}{\sum_{g=1}^{N} \phi_g} \right), \qquad M_f \le |\mathcal{N}_f|, \tag{9}$$

with the cap applied to each frame. Largest-remainder rounding makes the shares sum to B_M, and when every ϕ_g is zero the budget is split uniformly under the same cap and rounding. As a result, a frame with high scores outside its keep set merges more of its non-kept tokens, and a frame with low scores prunes more of them.

Each merge token m joins the kept token of the same frame whose encoder feature is most similar,

$$\text{dst}(m) \ = \ \arg\max_{k \in \mathcal{K}_f} \ \frac{\langle F_m, F_k \rangle}{\|F_m\| \, \|F_k\|}, \tag{10}$$

where F_m and F_k are the encoder features of m and of the kept token k. Each kept token is then replaced by the score-weighted average of itself and the tokens that join it,

$$\hat{F}_k \ = \ \frac{F_k + \sum_{m:\, \text{dst}(m)=k} S_m \, F_m}{1 + \sum_{m:\, \text{dst}(m)=k} S_m}. \tag{11}$$

The kept token has a fixed weight of 1, so it stays the reference of its group.

D Backbone Instantiation

Table A2 summarizes how PrePARE is set up on each backbone model, and the paragraphs below give the details.

VGGT (Wang et al., 2025b). The encoder is the frozen DINOv2 ViT-L/14 (Oquab et al., 2023). At 518×392 input, each frame has $P = 1036$ patch tokens of width $D = 1024$, and the interface is the

Table A2: Instantiation of PrePARE on four backbone models.

	VGGT	π^3	**MapAnything**	**Depth Anything 3**
Encoder E	frozen DINOv2 ViT-L/14; 518×392, $P = 1036$, $D = 1024$	frozen DINOv2 (VGGT init.); same P, D	DINOv2 ViT-G layer-24 features, $D = 1536$, max side 518; frozen here	blocks $0 \ldots 8$ of one 40-block ViT (giant, $D = 1536$); 504^2, $P = 1296$
AA stack A; interface	24 blocks (frame + global attention); cut at encoder output $\rightarrow$ block 1	36 layers (view + global attention); cut at encoder output $\rightarrow$ layer 1	16 layers, 24 heads, $D = 1536$; cut at encoder output $\rightarrow$ layer 1; images-only (no geometric side inputs)	blocks $9 \ldots 39$, cross-view on odd blocks from 13; cut after block 8, four blocks upstream of the first cross-view block
Special tokens $\mathcal{T}$	1 camera + 4 register tokens per frame	5 decoder register tokens per frame (the encoder's own class and register tokens never reach the interface)	1 global scale token	1 camera token per view
Anchor set $\mathcal{A}_f$; α	$\{[\text{CAM}]_f\}$; $\alpha = 0.25$	$\emptyset$; $\alpha = 0$	the scale token shared by all frames; $\alpha = 0$ (no per-frame anchor)	$\{[\text{CAM}]_f\}$; $\alpha = 0.25$
Dense head; restoration at	DPT depth and point heads on blocks 4, 11, 17, 23 (0-based); one module per tap, Q and K shared	per-view decoders reading a concatenation of the last two layers; one module on that concatenation	DPT head on two information-sharing taps plus the final layer; three modules	Dual-DPT on blocks $[19, 27, 33, 39]$; each tap concatenates within- and cross-view states ($2D = 3072$); one module per tap

encoder output entering the first of the 24 AA blocks, each a pair of frame-wise and global attention. The camera token and the four register tokens of each frame skip routing, and the anchor set is the frame's camera token, with $\alpha = 0.25$. The DPT depth and point heads (Ranftl et al., 2021) read AA blocks 4, 11, 17 and 23 (0-based), each a 2048-wide concatenation of the frame-wise and global states. One restoration module is attached to each of these four streams, and together they hold the 4.74 M parameters of Table A1. Unless stated otherwise, VGGT results leave the first frame unrouted: its patch tokens pass through all 24 AA blocks in full, and routing applies to the other frames.

π^3 (Wang et al., 2026d). The encoder is the frozen DINOv2 of π^3, initialized from VGGT's, with the same P and D, and the interface is its output entering the first of 36 alternating view-wise and global layers. π^3 is permutation-equivariant and has no camera or reference token, so the anchor set is empty, $\alpha = 0$, and $S^\star = \mathrm{Norm}_f(S_{\mathrm{match}})$. The encoder discards its own class and register tokens, so the interface carries patch tokens only. The decoder then adds five register tokens to each frame, and these skip routing and restoration like the special tokens of the other backbone models. At keep ratio r, each frame therefore enters the AA stack with $5 + \lceil 1036\, r \rceil$ tokens. The heads of π^3 read one concatenation of the last two AA layers, so a single restoration module runs on that concatenation.

MapAnything (Keetha et al., 2026). The encoder is the DINOv2 ViT-G of MapAnything, with normalized layer-24 features ($D = 1536$, patch size 14, longest image side 518), fine-tuned by MapAnything and kept frozen here. The interface sits between this encoder and the first of the 16 AA layers. We use the images-only input mode, in which each view is given as an image without intrinsics, depth, rays or poses, so no geometric input is added to the encoder features before the interface. The single scale token skips routing, and the reference-view embedding, which is added to the tokens of view 1, travels with the kept tokens. Because the scale token is shared by all frames, MapAnything has no per-frame anchor and uses $\alpha = 0$. Restoration is attached to three streams: the two intermediate taps that the dense head reads and the output of the last layer. The scale head reads the scale token directly.

Depth Anything 3 (Lin et al., 2025). We use the giant checkpoint, with 40 blocks, $D = 1536$, 24 heads and 1.36 B frozen parameters, whose first cross-view block is block 13. At 504×504 input, each view has $P = 1296$ patch tokens, and DA3 has no register tokens. The interface is placed after block 8. The scorer reads the residual stream after block 8, and blocks 9 to 39, which include all 14 cross-view blocks, run on the reduced sequence. DA3 therefore runs nine blocks on the full grid before routing, which is one reason its peak-memory saving at $r = 0.8$ (20 %) is smaller than that of VGGT at $r = 0.4$ (80 %, Table 2). A camera token for each view is inserted at block 13 in place of the class token. It skips routing and serves as the anchor, with $\alpha = 0.25$. The Dual-DPT head reads blocks 19, 27, 33 and 39, and each tap concatenates the last within-view state with the current

cross-view state, giving $2D = 3072$ channels. Each of the four restoration modules takes the full 3072-channel tap, with one normalization, a $3072 \to 256$ value projection and a $256 \to 3072$ output projection, and uses the 1536-wide patch stream only to compute the attention. After restoration, the tap is split into its two halves so that the frozen normalization of the backbone can be reapplied to the cross-view half. At evaluation, DA3 processes 16 frames per inference chunk, so restoration runs once per tap and chunk.

E Evaluation Protocol

Datasets. A ScanNetv2 clip of N frames takes frames of the scene at a fixed integer stride, starting from the first frame. Of the 50 test scenes, 49 have enough frames for $N = 500$ and 39 for $N = 1000$, and the results at these lengths are averaged over those scenes. 7-Scenes has 18 test clips: the 16 sequences of 1000 frames give 334 frames at stride 3 and 100 at stride 10, and the two 500-frame stairs sequences give 167 and 50. NRGBD has 9 scenes, which give 40 to 168 frames at stride 10. The 7-Scenes and NRGBD clips follow the FastVGGT loader without a frame cap.

Baselines. Each baseline runs from its released code at its recommended setting: FastVGGT (Shen et al., 2026) and LiteVGGT (Shu et al., 2025) at merge ratio 0.9, Co-Me (Chen et al., 2025) at $p = 0.6$ on VGGT and $g = 4$ on the other backbone models, and SAF3R (Deng et al., 2026a) at its released head-sparsity profile. FastVGGT and LiteVGGT are evaluated on VGGT, SAF3R on π^3 and Depth Anything 3, and Co-Me on all four backbone models. On 7-Scenes and NRGBD, LiteVGGT evaluates the longest FP8-compatible prefix of each clip, which on 7-Scenes is 96 or 48 frames at stride 10 and 328 or 160 frames at stride 3, and it reads its points from the depth and camera heads.

3D reconstruction. On ScanNetv2, the predicted point cloud is aligned to the ground truth by matching the centers and diagonals of their bounding boxes. Per-direction distances are clamped at 0.5 m, and the Chamfer distance is the sum of the two directions over 100k sampled points at a voxel size of 0.05, with sampling seed 33. The depth-confidence threshold is 1.0, so every predicted point is kept. Fail counts the scenes whose Chamfer distance exceeds 0.5. On 7-Scenes and NRGBD, Acc, Comp and NC are computed after scale-and-shift alignment.

Depth. Depth is read from the frozen depth head at the model grid and compared with the ground-truth depth after one median scale per clip. AbsRel and $\delta{<}1.25$ are averaged over the pixels whose ground truth lies in (0.1, 10) m, with no confidence mask.

Point tracking. Only VGGT has a tracking head. The frozen VGGT tracker runs on every frame of the 30 TAP-Vid DAVIS clips (Doersch et al., 2022) with first-frame-visible queries, and AJ, δ_{avg} and OA are computed at TAP-Vid's 256-pixel resolution. With PrePARE, the tracker reads the restored feature taps.

Peak memory. Peak GPU memory uses the same clip construction, resolution and precision for every method. The CUDA peak-memory counter is reset immediately before the measured forward pass and read after device synchronization. Tables report the maximum over the scenes of a run, in GiB, and OOM means that the forward pass did not complete on the 93 GiB card.

Timing. Each per-scene record stores the wall-clock time of the measured forward pass as the median of three timed runs after two warm-up runs, with CUDA synchronization. A Time cell is the median of this value over the scenes of the run.

Statistics. Parity with the unpruned model is tested with the two-sided exact paired sign test of Section 4.1. Table A3 also reports the two-sided paired Wilcoxon signed-rank test, and all p-values are uncorrected.

F Additional Results

F.1 Keep-ratio sweep

Table A3 lists the keep ratios behind the operating points of Table 2 on π^3, Depth Anything 3 and MapAnything, together with the VGGT operating point, whose sweep is in Table 6a. Each backbone model uses one trained checkpoint for every keep ratio, and only the number of kept tokens per frame changes. The test is made at $N = 300$ on π^3 and DA3 and at $N = 100$ on MapAnything, whose

unpruned model does not fit at $N = 300$. This rule gives $r = 0.6$ on π^3 and $r = 0.8$ on DA3 and MapAnything. On DA3 at $r = 0.6$, the sign test gives $p = 0.065$ and the Wilcoxon test $p < 0.01$, and we count this keep ratio as worse.

At $N = 500$, where the unpruned π^3 and DA3 still fit, the same operating points are better than the unpruned model on 27 of 49 scenes on π^3 (sign test $p = 0.57$, Wilcoxon $p = 0.43$) and on 35 of 49 on DA3 (sign test $p = 0.004$, Wilcoxon $p = 0.006$), and they save 33 and 29 % of peak memory.

Table A3: Keep-ratio sweep on ScanNetv2: CD, ATE (m), ARE (degrees), Mem (GiB), Fail (scenes with CD > 0.5), and Saved (peak-memory reduction against the block's unpruned row). Last column: paired sign test against that row (scenes where PrePARE is better, p) and paired Wilcoxon p.

Setting	CD↓	ATE↓	ARE↓	Mem↓	Fail	Saved	Sign / Wilcoxon
VGGT, $N = 300$ (other keep ratios in Table 6a)							
unpruned	0.4512	0.096	4.17	72.3	26		*reference*
$r = 0.4$	0.4187	0.092	3.99	14.4	24	80%	34/50, 0.015 / 0.014
π^3, $N = 300$							
unpruned	0.4301	0.053	2.48	15.7	21		*reference*
$r = 0.8$	0.4283	0.075	3.88	14.1	17	10%	26/50, 0.89 / 0.87
$r = 0.6$	0.4203	0.057	2.55	13.3	21	15%	25/50, 1.00 / 0.21
$r = 0.4$	0.4566	0.090	3.94	13.3	20	15%	17/50, 0.033 / 0.018
Depth Anything 3, $N = 300$							
unpruned	0.3929	0.083	4.24	40.2	18		*reference*
$r = 0.8$	0.3924	0.085	4.32	32.3	19	20%	26/50, 0.89 / 0.74
$r = 0.6$	0.4383	0.100	4.74	26.1	22	35%	18/50, 0.065 / < 0.01
$r = 0.4$	0.4374	0.117	5.26	20.6	17	49%	16/50, 0.015 / < 0.01
MapAnything, $N = 300$							
unpruned	out of memory on all 50 scenes						
$r = 0.8$	0.4236	0.211	9.39	27.5	18		*no unpruned reference*
$r = 0.4$	0.6668	0.800	56.49	20.5	36		*no unpruned reference*
π^3, $N = 100$							
unpruned	0.4161	0.054	2.36	7.8	20		*reference*
$r = 0.8$	0.4241	0.078	3.59	7.1	18	9%	22/50, 0.48 / 0.26
$r = 0.6$	0.4228	0.060	2.51	6.8	17	13%	26/50, 0.89 / 0.31
$r = 0.4$	0.4365	0.091	3.74	6.8	18	13%	23/50, 0.67 / 0.097
Depth Anything 3, $N = 100$							
unpruned	0.3882	0.084	4.05	17.6	18		*reference*
$r = 0.8$	0.3900	0.086	4.05	14.2	17	19%	24/50, 0.89 / 0.31
$r = 0.6$	0.3998	0.100	4.61	12.4	18	30%	23/50, 0.67 / 0.34
$r = 0.4$	0.4085	0.120	5.44	11.3	16	36%	14/50, < 0.01 / 0.014
MapAnything, $N = 100$							
unpruned	0.3995	0.127	5.58	52.9	18		*reference*
$r = 0.8$	0.4195	0.219	8.16	16.0	16	70%	25/50, 1.00 / 0.60
$r = 0.6$	0.5305	0.523	30.87	15.5	25	71%	16/50, 0.015 / < 0.01
$r = 0.4$	0.5992	0.734	41.14	15.1	28	71%	8/50, < 0.01 / < 0.01

F.2 DEPTH ESTIMATION

Table A4 reports depth from the frozen VGGT depth head with PrePARE at $r = 0.4$, on ScanNetv2 at $N = 300$ and zero-shot on 7-Scenes and NRGBD, under the protocol of Section E. The zero-shot rows also give the unpruned model.

Table A4: **Depth estimation** with the frozen VGGT depth head: median-scaled AbsRel↓ and δ<1.25 ↑ over ground truth in (0.1, 10) m, Time (s) and Mem (GiB). ScanNetv2 at $N = 300$; 7-Scenes and NRGBD zero-shot with the ScanNetv2-trained modules (reconstruction in Table 4).

	Depth		Cost	
Method	**AbsRel**	δ_1	**Time**	**Mem**
ScanNetv2, $N = 300$				
PrePARE r=0.4	0.052	0.981	9.1	14.4
7-Scenes, stride 3				
unpruned	0.058	0.958	59.8	79.9
PrePARE r=0.4	0.076	0.967	21.5	17.8
7-Scenes, stride 10				
unpruned	0.057	0.958	6.9	27.2
PrePARE r=0.4	0.075	0.968	2.2	10.1
NRGBD, stride 10				
unpruned	0.013	0.993	9.7	42.5
PrePARE r=0.4	0.052	0.974	4.8	11.7

F.3 3D POINT TRACKING WITH THE VGGT TRACKING HEAD

Table A5 reports point tracking with the frozen VGGT tracker under the protocol of Section E, for the unpruned model and for PrePARE at $r = 0.4$. Four of the 30 unpruned runs did not complete, so the table covers the 26 clips on which both runs completed.

Table A5: **3D point tracking on VGGT.** The frozen VGGT tracker on the 26 TAP-Vid DAVIS clips both runs completed, every frame of each clip, first-frame-visible queries, metrics at 256 px (AJ, δ_{avg}, OA, all ↑); PrePARE hands it the restored taps. Time is wall-clock per clip (s), Mem peak GiB.

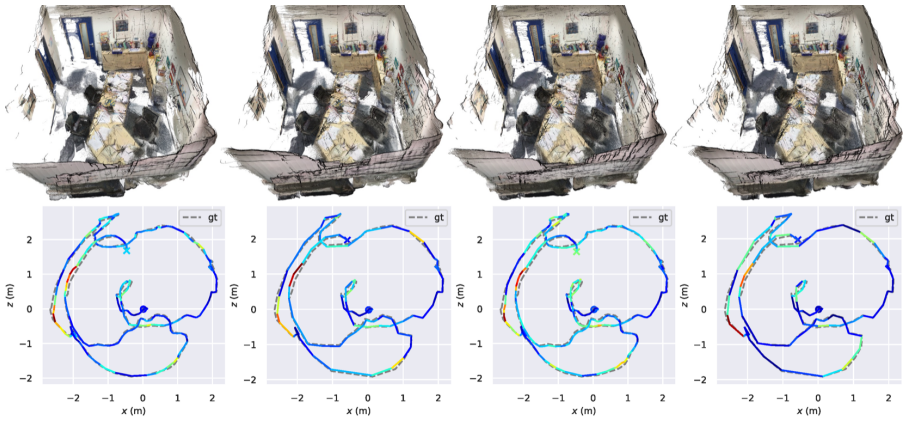
Method	**AJ**	δ_{avg}	**OA**	**Time**	**Mem**
unpruned	0.183	0.372	0.476	6.9	32.4
PrePARE r=0.4	0.130	0.310	0.454	3.6	18.2

F.4 QUALITATIVE RESULTS

Token-selection maps. Figure 3a and the overlap statistics below use one clip per ScanNetv2 scene, with 4 frames at stride 10 around the middle of the scene. The clips are short because the exact AA-usage target stores every global-attention map, which limits a clip to about five frames. Three score maps are each routed at $r = 0.40$ and $\gamma = 0.30$, with every frame routed: the target $S^\star$ of Equation (1) at its training setting ($\alpha = 0.25$, patch tokens only), the learned Token Scorer, and the encoder-similarity control S_{red}. Over the 50 clips, the keep sets of the scorer and the control contain 73.3 and 55.5 % of the tokens that the target keeps (standard errors 0.5 and 0.6 points), and a random keep set contains 40 %.

Long sequences. Figure 3b uses scene0353_02. Among the 39 scenes long enough for $N = 1000$, it is the scene whose Chamfer distances, summed over the four runs shown, deviate least from each run's per-scene median, and it is also the median scene of PrePARE on VGGT at $N = 1000$. Each panel is a one-scene run with the settings of Table 2. Its Chamfer distance and ATE match the per-scene values of the table run to within 10^{-4}, and the panel titles give the peak memory of the table runs. The points are the predicted cloud that the Chamfer protocol scores, after its bounding-box alignment, subsampled to 200,000 points, colored by height, and drawn top-down over the ground-truth mesh in gray. The 0.3 to 2.4 % of points that fall outside the footprint of the mesh are not drawn. Camera paths are the estimated camera centers after a Sim(3) alignment to the ground-truth centers, drawn in the frame of the mesh.

Point clouds and trajectories. Figure A1 shows reconstructed point clouds and camera trajectories for VGGT, FastVGGT, LiteVGGT and PrePARE.



VGGT FastVGGT LiteVGGT PrePARE

Figure A1: Qualitative results on VGGT. Top: point clouds on ScanNetv2 and 7-Scenes. Bottom: camera trajectories against ground truth on a ScanNetv2 sequence.

G Ablation Settings

The ablations of Table 6 run on VGGT over the 50 ScanNetv2 scenes at $N = 300$, with $r = 0.40$, $\gamma = 0.30$ and $\alpha = 0.25$ unless a row changes one of them, and with the first frame unrouted. Each row changes one component of the default and keeps the others.

Keep ratio. The rows of Table 6a with different r use the same trained checkpoint, and only the number of tokens kept per frame changes.

Restoration. Zero fill places the AA outputs of the kept tokens at their grid positions and sets every merged and pruned position to zero. Bilinear interpolation fills the merged and pruned positions from the kept positions of the grid. ToMe-style unmerging gives each merged position the output of the kept token it joined and fills each pruned position by inverse-distance weighting over its four nearest kept positions.

Training clips. The WS row trains on clips sampled at a wider frame stride than the default.

Merge fraction and blend. With $\gamma = 0$, every non-kept token is pruned, and with $\gamma = 1$, every non-kept token is merged into the keep set (Equation (9)). With $\alpha = 0$, the target is the matching score alone, and with $\alpha = 1$, it is the anchor score alone (Equation (1)).

Schedule. Stage 1 only trains the scorer on the AA-usage target and skips stage 2. Stage 2 only trains the scorer and the restoration modules from initialization under $\mathcal{L}_{\text{restore}} + \mathcal{L}_{\text{downstream}}$, without the AA-usage target. Joint trains both modules in a single stage from initialization under $\mathcal{L}_{\text{stage2}}$. Two-stage is the default schedule of Section 3.5.

Criterion. Encoder cosine replaces the learned scorer with the control score S_{red} of Section 3.3. The $S^\star$ oracle routes by the target $S^\star$ computed from an unpruned forward pass on each test clip, so it needs the unpruned model at inference.

Composition with an in-AA merger. In Table 5, the in-AA merger runs inside each global-attention block of VGGT on the tokens that PrePARE keeps. Query, key and value are merged after the rotary embedding, attention runs on the merged operands, and the output is unmerged after the output projection, as in FastVGGT, so the residual stream keeps the length set by PrePARE. The released mergers expect five special tokens at the start of each frame and a rectangular token grid, so the kept tokens of each frame are laid out as a rectangular grid (5×83 for the 415 tokens kept at $r = 0.4$), and the first frame is routed like the others. The matched baseline is PrePARE alone at the same keep ratio, also with the first frame routed. The scorer and the restoration modules are not retrained for the composition.