Understanding Multi-View Transformers

Julien Gaubil[1*] **Michal Stary**[2*] **Louis Martinez**[3]
Andreas Geiger[4,5] **Maks Ovsjanikov**[1†] **Ayush Tewari**[6†] **Vincent Sitzmann**[7†]
[1]LIX, École Polytechnique, IPP [2]TUM [3]Lamarck Labs
[4]KE:SAI [5]Tübingen AI Center, University of Tübingen
[6]Cambridge University [7]MIT CSAIL

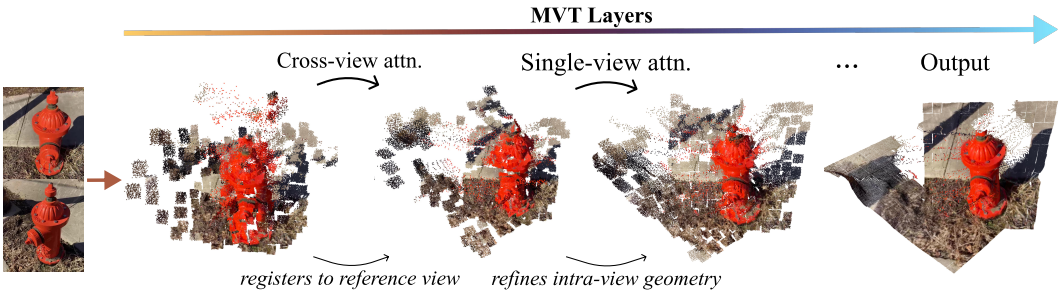


Figure 1: **Probing multi-view transformers.** We train layer-wise pointmap probes which decode intermediate tokens into dense 3D reconstructions and demonstrate that multi-view transformers reconstruct 3D scenes through progressive refinement. Across layers, cross-view attention improves multi-view consistency, while single-view attention refines geometry within the non-reference views.

Abstract

Multi-view transformers solve 3D vision tasks in a single feed-forward pass, yet their internal mechanisms remain poorly understood. We view their residual layers as progressively updating a latent state, and study how this state evolves during 3D reconstruction. To make the state geometrically interpretable, we train layer-wise probes that decode intermediate tokens into 3D pointmaps, yielding a 3D reconstruction at every layer of the network. We show that multi-view transformers reconstruct 3D scenes progressively, with most layers refining the reconstruction, even though they operate in a single forward pass. Reconstruction speed varies with sample difficulty: easy samples converge within a few layers, whereas difficult ones require more updates. We further analyze these updates and find consistent, complementary roles for single-view and cross-view attention. We demonstrate that cross-view attention improves multi-view alignment, while single-view attention refines intra-view geometry. Finally, we identify cross-view attention heads that specialize in correspondence matching. We find that correspondences produced by these layers are refined jointly with the reconstructed geometry, and that encoder choice strongly affects correspondence accuracy. Overall, our analysis provides a direct view of the reconstruction process inside multi-view transformers, and suggests similarities with traditional Structure-from-Motion pipelines. The code for the analysis is available at https://github.com/JulienGaubil/und3rstand.

*Equal contribution. Work partially done while visiting students at MIT.
†Equal supervision.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

1 Introduction

Reconstructing 3D scenes from images has traditionally relied on explicit, iterative optimization pipelines such as COLMAP [1]. Models in this family estimate correspondences, camera poses, and scene geometry through a structured sequence of updates. Recent multi-view transformers (MVTs) depart from this paradigm. Starting with DUSt3R [2], such approaches replace this expert-designed pipeline with a generic transformer that processes two or more views and predicts scene geometry in a single forward pass. This design has rapidly become a leading approach for 3D reconstruction [3–8] and 3D-aware tasks such as novel view synthesis [9–14] or correspondence matching [15]. Compared to conventional optimization-based methods, feed-forward models are fast and can solve underdetermined reconstruction problems. Yet their internal computation remains poorly understood, beyond evidence that geometric structures such as correspondences [16] and the fundamental matrix [17] can emerge in their representations. Understanding this computation can inform useful inductive biases [18, 19], or support zero-shot extensions to dynamic scenes [20, 21].

Prior work has studied residual networks — transformers in particular — through an iterative lens, casting inference as a process that carries and updates a state through each layer's skip connection [22–28]. Inspired by this view, we study how spatial geometry emerges in the residual state of multi-view transformers. Specifically, we train small probes [22] to decode individual patch tokens into explicit 3D pointmaps after each residual update. Unlike transformer analyses in vocabulary or language embedding space [24–27], 3D has a well-studied geometric structure that lets us inspect the layer-by-layer sequence of decoded states. Moreover, by probing multi-view transformers on their native reconstruction task, we directly study how these models solve multi-view 3D reconstruction.

Using this framework, we examine recent multi-view transformers: DUSt3R [2], VGGT [3], and Depth Anything 3 [8]. We demonstrate that, although they reconstruct 3D scenes in a single forward pass, their intermediate reconstructions *gradually* move toward the final prediction. Most layers improve the reconstruction on average. Beyond identifying the geometric information present in intermediate features, our probes reveal *how* this information changes across layers. We find consistent and complementary roles for single-view and cross-view attention. Cross-view attention increases multi-view consistency by aligning predictions to the reference view, while single-view attention improves intra-view geometry in non-reference frames. Together, these updates progressively register the views into a locally and globally coherent reconstruction. Finally, we show that correspondences emerging in MVT cross-view attention maps [16, 17] are refined jointly with internal geometry.

Overall, multi-view transformers trained on large 3D datasets show similarities to traditional pipelines such as Structure-from-Motion [1]: network depth acts as a refinement budget, geometry progressively improves, and correspondences emerge and are used during reconstruction. We further retrain MVTs from scratch in a unified codebase to study how design affects these properties. In small-scale experiments with matched compute budgets and parameter counts, attention-only MVTs outperform MVTs that include MLP layers, which lack a clear geometric interpretation in our framework. Additionally, models trained with CroCoV2 [29] encoders exhibit significantly higher correspondence accuracy in their cross-view attention maps than models trained with DINOv2 [30] encoders.

Taken together, our work provides a systematic picture of the learned reconstruction process implemented by multi-view transformers. Our contributions are **1)** a layer-wise pointmap probing framework applied to study the 3D reconstruction process of several recent multi-view transformers; **2)** evidence that feed-forward reconstruction is internally progressive, with complementary roles for single-view and cross-view attention in this refinement; and **3)** an analysis of the correspondences that emerge in cross-view attention maps and are refined across layers alongside the geometry.

2 Related Work and Preliminaries

3D Reconstruction from Images. Reconstructing 3D scenes from images is a long-standing problem [31]. Classical Structure-from-Motion (SfM) systems such as COLMAP [1] solve unposed reconstruction through an explicit pipeline of correspondence matching, relative pose estimation, triangulation, and joint optimization of camera poses and geometry. Learning-based methods have replaced or augmented parts of this pipeline with learned components [32–34]. DUSt3R [2] marks a more direct departure: it uses a vision transformer (ViT) [35, 36] to jointly process two views and predict scene geometry in a single forward pass. This *multi-view transformer* design quickly

Table 1: **Design choices of the studied multi-view transformers.** Models differ in encoder and MVT initialization, cross-view attention parameterization (CVA), and output 3D parameterization.

	DUSt3R [2]	**DA3** [8]	**VGGT** [3]
Encoder init.	CroCoV2 [29]	DINOv2 [30]†	DINOv2 [30]
MVT init.	CroCoV2 [29]	DINOv2 [30]†	Random
MVT weights	Per-view	View-shared	View-shared
CVA param.	Cross-attention	Global self-attention	Global self-attention
Output param.	Pointmap	Depth + pose + ray map	Pointmap + depth + pose + FoV

† DA3 initializes its encoder and MVT using the first third and remaining two thirds of DINOv2, respectively.

became a de facto choice for 3D reconstruction tasks [3–8, 37–39]. Other approaches process views sequentially while updating a latent spatial memory [40–42], which naturally supports long image sequences. Replacing handcrafted SfM updates with generic attention blocks improves speed and robustness, but makes reconstruction less transparent. In this paper, we show that multi-view transformers learn interpretable operations that resemble traditional optimization pipelines.

Multi-View Transformers. Beyond reconstruction, multi-view transformers have been adopted for 3D-aware tasks such as correspondence matching [15] and novel view synthesis [9–14]. Their architecture is illustrated in Fig. 2(a): a shared encoder first processes each view separately, then a multi-view transformer jointly processes all views before task-specific heads predict the final outputs. This design has several advantages: it can reuse pretrained ViT representations, supports various forms of cross-view communication through attention (see Fig. 2(b)), and adapts to many downstream tasks. Our study focuses on DUSt3R, VGGT, and Depth Anything 3 (DA3) [2, 3, 8], whose main design differences in terms of initialization, cross-view attention mechanism, and output parameterization are summarized in Table 1.

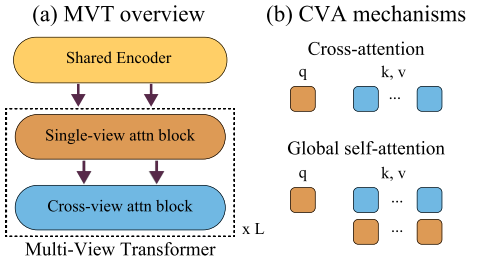


Figure 2: **Multi-view transformers overview. (a)** A shared encoder processes each image before the multi-view transformer processes all frames jointly. We omit output heads. **(b)** Cross-view attention (CVA) can be implemented with cross-attention to tokens from other views, or global self-attention over tokens from all views.

Analyzing Iterative Inference in Transformers. Residual networks [43] can be interpreted as iterative inference systems: each residual layer updates a latent state, and training encourages these updates to reduce the loss [23]. This view has been used to study how transformer representations evolve across depth in language models [24–27]. Such analyses often rely on probing [22], which makes intermediate states inspectable by training lightweight heads on frozen representations. In 3D vision, prior probing work has mostly studied monocular encoders at one or a few layers [44–50]. We instead adopt the iterative perspective and probe the multi-view transformer itself, layer by layer, to analyze how it exchanges information across views and progressively reconstructs 3D geometry.

Understanding and Improving Multi-View Transformers. Existing analyses of multi-view transformers mostly focus on attention maps [16, 20, 21, 51]. Closest to our work, Bratulić et al. [17] probe MVT layers for the fundamental matrix and study the role of correspondences and learned priors in reconstruction. We instead probe for 3D reconstructions directly from intermediate states. This reveals how multi-view transformers progressively refine geometry despite operating in a single forward pass, and clarifies each layer's role in the reconstruction process.

3 Understanding Multi-View Transformers

3.1 Probing Residual Pointmaps

To study how geometry evolves in the residual state of a multi-view transformer, we train probes on patch features extracted after each residual update.

Probe 3D parameterization. Our probes predict pointmaps: per-pixel 3D points expressed in a coordinate frame shared across views. Unlike local parameterizations such as depth, pointmaps place all views at a common scale and pose, facilitating joint visualization and evaluation. This makes them well suited for inspecting the 3D structure encoded in the latent state. We express all pointmaps in the coordinate frame of the first, reference view, mirroring the training setup of the evaluated models.

Multi-view setup. We mainly focus on two-view reconstruction to study how MVTs solve this central multi-view geometry problem. Two-view reconstruction can be extended to n views with iterative registration procedures, and it lets us compare DUSt3R, which takes two views as input, with VGGT and DA3, which can process longer sequences. For VGGT and DA3, we conduct additional four-view experiments to study feed-forward reconstruction beyond the two-view case.

Probe design and training. We train two probes, one for the reference view and one shared across all remaining views. This reflects the asymmetry of the prediction tasks: reconstructing the reference-view pointmap requires estimating depth and intrinsics, while reconstructing a non-reference-view pointmap also requires estimating that view's pose relative to the reference. We limit the probe capacity to ensure that it faithfully reflects the 3D information encoded in the state: each probe maps individual patch features to 3D points without communication across patches. This prevents the probe from solving global reasoning tasks on its own, such as relative pose estimation or joint geometry reconstruction. We train probes with the confidence-aware DUSt3R loss [2], additionally regressing per-view confidence maps. Appendix A.4 illustrates the probe mechanism and details the loss.

3.2 Evaluating Pointmap Probes

Pointmap metrics. We evaluate predicted pointmaps with three errors. First, e_{point} is the absolute-relative 3D pointmap error used in [7], computed after scale normalization. We also report a rigidly aligned variant, e_{proc}, obtained by applying a per-view Procrustes alignment between predicted and ground-truth pointmaps before computing the same error. This metric evaluates intra-view geometry independently of relative pose error. Finally, e_{mv} measures multi-view 3D consistency as the discrepancy between predictions for corresponding 3D points in different views. Metrics are computed per sample and view, then averaged over test samples.

Difficulty subsets. We categorize test samples by reconstruction difficulty. For each model, we compute e_{point} for the final pretrained reconstruction and obtain one error score per sample. We then define per-model easy and hard subsets as the bottom and top 20% of samples, respectively.

Convergence analysis. To understand how MVT layers progressively refine the reconstruction toward a final estimate, we study how intermediate probe reconstructions approach the final probe reconstruction across depth. We measure e_{point} between each intermediate pointmap and the final probe reconstruction, rather than between the intermediate pointmap and ground truth. This yields per-layer error sequences that approach zero regardless of the final reconstruction error, allowing comparison across difficulty subsets and models. We summarize convergence with the inlier ratio at an e_{point} threshold of 0.05, averaged across samples.

Measuring layer contributions. We quantify how different MVT layer types contribute to reducing a pointmap error e. Let e_k denote the dataset-averaged value of e for pointmaps probed after layer k, and let $\mathcal{K}_{\mathcal{F}}$ be the set of layer indices with type $\mathcal{F}$, where $\mathcal{F}$ is single-view attention, cross-view attention, or MLP. We define the Accumulated Layer Contribution Percentage (ALCP) as:

$$\text{ALCP}(e, \mathcal{F}) = 100 \cdot \frac{\sum_{k \in \mathcal{K}_{\mathcal{F}}} (e_{k-1} - e_k)}{e_{\text{enc}} - e_{\text{mvt}}}, \tag{1}$$

where e_{enc} and e_{mvt} are the dataset-averaged errors e at the encoder and MVT output, respectively. A positive $\text{ALCP}(e, \mathcal{F})$ indicates that layers of type $\mathcal{F}$ reduce the error e on average across the MVT.

We detail metrics, alignment procedures, and difficulty subset definitions in Appendix Section A.5.

3.3 Additional Zero-shot Evaluations

To complement the pointmap probes, we use two zero-shot analyses to study how latent states and reconstructions evolve across multi-view transformer layers.

Representation similarity measure. To relate pointmap error evolution to changes in the latent state, we measure the calibrated linear Centered Kernel Alignment (CKA) [52, 53] between MVT block outputs. We compute the similarity between patch tokens extracted after the last residual connection of each block.

Evaluating emergent correspondences. We also evaluate the emergence of correspondences, a core cue in SfM pipelines, during reconstruction. Following [16], we extract patch-level zero-shot correspondences from cross-view attention (CVA) maps. Let (a, b) be a pair of views, and let p^a be a query patch in view a with a ground-truth correspondence in view b ("stereo patch"). For each CVA layer ℓ and head h, we predict its match as the target patch p^b with the maximum attention score:

$$\hat{p}^b_{\ell,h} = \arg\max_{p^b} A^{a\to b}_{\ell,h}\left(p^a, p^b\right). \tag{2}$$

For VGGT and DA3, we decompose each global attention map into within-view and cross-view maps, and use the cross-view map. For DUSt3R, we use the cross-attention maps directly and follow the heuristic from [16], suppressing the first sequence token when it dominates the attention ranking. We report per-head correspondence accuracy over all stereo query patches and summarize each layer by its best-performing head. We obtain ground-truth patch correspondences by aggregating pixel correspondences derived from reprojected ground-truth 3D geometry.

The CKA and correspondence evaluation procedures are detailed in Appendix A.6 and Appendix A.7.

3.4 Multi-View Transformer Ablations

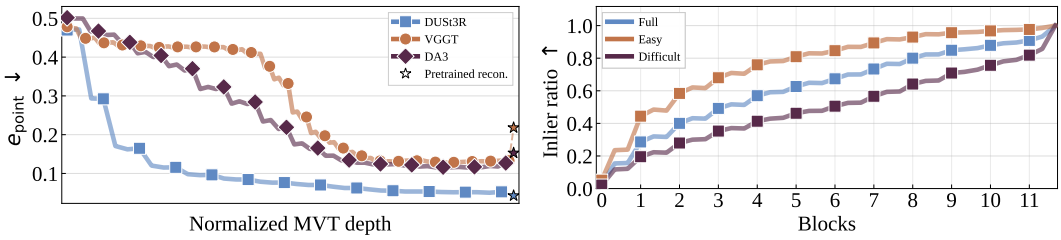
To study how MVT design choices affect 3D state evolution and emergent correspondences, we implement multi-view transformers in a unified codebase that allows us to vary three components from Table 1: the encoder, cross-view attention mechanism, and weight sharing. We otherwise use a VGGT-style MVT with alternating single-view and cross-view attention. Following MUSt3R [42], we omit RoPE from cross-view attention. We use DUSt3R's pointmap output parameterization. The architecture is detailed in Appendix Section A.8.

4 Experiments

Through our experiments, we investigate three questions related to how multi-view transformers address 3D reconstruction:

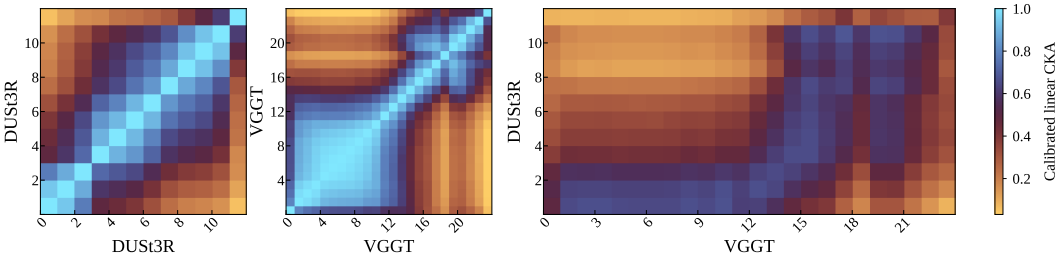
1. How does the internal state evolve across the multi-view transformer?
2. What is the role of individual layers?
3. Does the network utilize correspondences?

Datasets. We primarily train probes on 100,000 rendered images from the HM3D dataset [54] and evaluate the main metrics on 500 image pairs from held-out scenes. We additionally train and evaluate probes on BlendedMVS [55]. For qualitative evaluation, we use real-world examples from RealEstate10K and CO3Dv2 [56, 57], both within the models' training distribution. We additionally evaluate zero-shot correspondence matching on object-centric data, using GSO [58] and DTC [59] objects rendered against procedurally generated backgrounds in Blender; see appendix for details.



(a) **Layer-wise reconstruction error.** e_{point} evaluated after each residual update.

(b) **Convergence speed vs. difficulty (DUSt3R).** Inlier ratio within 5% of the final probe reconstruction.



(c) **Representation similarity.** Calibrated centered kernel alignment (CKA) across MVT block outputs.

Figure 3: **Second-view error and state evolution on HM3D. (a)** Pointmap test error relative to ground truth after each residual update; markers indicate block inputs. **(b)** Inlier ratio relative to the final probe reconstruction, grouped by sample difficulty. **(c)** Calibrated CKA similarity between second-view MVT block outputs. These results highlight the inherently progressive nature of the reconstruction process, where convergence speed varies with sample difficulty and phase transitions in error align with internal representation dynamics.

Training and implementation details. We parametrize each probe as a five-layer MLP with ReLU activations and hidden dimension 512, and train all probes for four epochs. We evaluate the official pretrained checkpoints of DUSt3R, VGGT, and Depth Anything 3 (DA3). Our default MVT implementation uses 12 alternating-attention blocks and a linear output head. The MVT is trained from scratch, while the pretrained encoder remains frozen. See Appendices A.1 and A.8 for details.

4.1 How Does the Internal State Evolve Across Multi-View Transformer Layers?

Feed-forward reconstruction is progressive. We first analyze the evolution of the pointmap error e_{point}. On average, most MVT layers reduce the error (Fig. 3a), consistent with prior analyses of residual networks and transformers [22–27]. This layer-wise improvement has a direct geometric interpretation: although multi-view transformers reconstruct 3D scenes in a single forward pass, they *progressively refine* geometry through several updates rather than in one step. This progression is analogous to iterative refinement, but differs from classical optimization in that each layer applies a *distinct* learned update rather than repeatedly applying a fixed rule to optimize an explicit objective.

Convergence speed depends on sample difficulty. Figure 4 illustrates this progressive refinement. It shows that reconstructions of easy and difficult image pairs improve at different rates. At the output of the shared encoder, the network already provides a strong geometric estimate of the first view, suggesting that monocular depth and camera intrinsics are largely encoded before multi-view interaction. In the simple example, the rotation is resolved after the first MVT block, and later blocks align scale and refine translation. In the difficult example, the correct pose emerges only after several blocks, as second-view pointmaps gradually move toward the final estimate. This indicates that MVT depth acts as an iteration budget in optimization: additional layers let difficult samples keep refining when early updates are insufficient. Quantitatively, the DUSt3R inlier ratio with respect to the final probe estimate rises from 5 to nearly 50% after the first MVT block on the easy subset. It remains less than half as high on the difficult subset and at least 20% below the easy subset until the last block (Fig. 3b). As observed in [60], convergence speed depends on sample difficulty. Easy samples converge in one or a few blocks, whereas harder samples may require more blocks to converge.

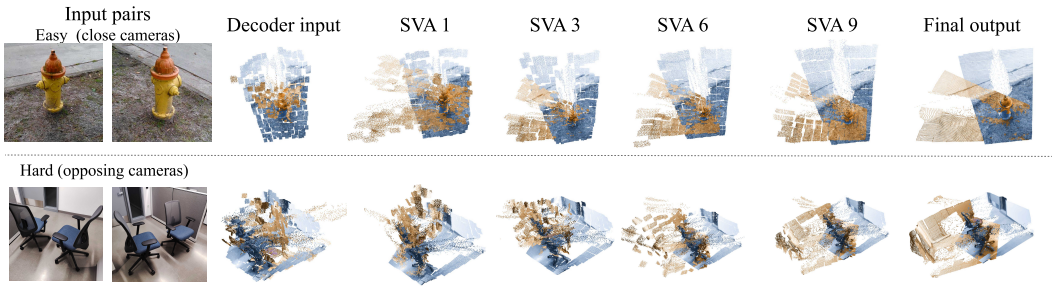


Figure 4: **Pointmap visualizations for easy vs. hard samples (DUSt3R).** We visualize pointmap predictions across MVT blocks for an easy example (top) and a challenging one (bottom). The first view is shown in blue and the second in yellow. The yellow-tinted patches progressively move through 3D space toward their final positions in a semi-rigid manner. While early layers already yield a reasonable reconstruction for simple cases, harder samples require refinement across more layers.

Error and state dynamics. We observe in Fig. 3a that error profiles differ across models. DUSt3R's error drops sharply when multi-view information enters the MVT, DA3 improves more gradually, and VGGT remains nearly flat before an abrupt decrease. To relate these regimes to the evolution of the latent state, we analyze CKA similarities between blocks. For the pointmap-based transformers DUSt3R and VGGT, CKA similarity exhibits a phase structure [52, 61, 62] that mirrors the phase structure observed in the pointmap error curves (Fig. 3a and 3c). This confirms that the probes track meaningful changes in the models' internal 3D state. This structure also appears in cross-model similarity between DUSt3R and VGGT: blocks within the same error regime, before or after the sharp error decrease, are more similar than blocks from different regimes. This suggests that the two models may follow similar reconstruction stages despite different convergence dynamics.

Explaining fast and slow dynamics. Output head design, more than MVT architecture, explains differences in error reduction speed across depth. To test this, we train MVTs from scratch on HM3D with either a final-layer or a multilayer head. Across architectures, final-layer heads yield rapid error reduction, whereas multilayer heads yield slower refinement (Appendix Section B.2, Fig. 13). Models such as DUSt3R, trained first with a final-layer head and then with a multilayer head, retain their rapid initial reduction. This explains DUSt3R's faster convergence relative to VGGT and DA3, which are trained from scratch with multilayer heads. We hypothesize that shortcuts to the output head redistribute reconstruction across MVT layers compared with standard residual networks [23, 43].

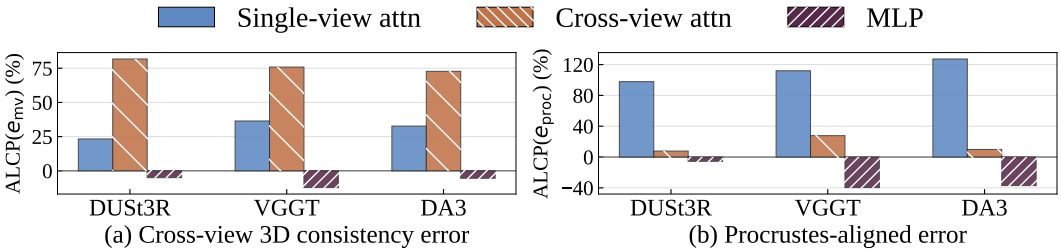
Additional results. Appendix B shows that probing after residual updates and using nonlinear probes are essential for obtaining stable and interpretable layer-wise trajectories. It also reports detailed per-model error and convergence curves, CKA analyses, and output head ablations.

Takeaway. *Multi-view transformers refine geometry progressively despite using a single forward pass. The final reconstruction emerges within the first few layers for easy pairs, while difficult pairs require further refinement.*

4.2 What Is the Role of Individual Layers?

To understand the role of individual layers, we inspect their action on probed intermediate pointmaps.

Cross-view attention improves multi-view consistency. In Fig. 4, the reference-view pointmap remains stable, while the non-reference pointmap is progressively registered into the reference coordinate frame. Within this process, cross-view attention layers improve multi-view consistency by bringing predictions of corresponding points across views closer together. Figure 20 illustrates this behavior. We quantify this effect with the accumulated layer contribution ALCP($e_{\text{mv}}, \mathcal{F}$), which measures how much each layer type $\mathcal{F}$ reduces the multi-view 3D consistency error. As shown in Fig. 5(a) on HM3D probes, cross-view attention accounts for most of this reduction, contributing about 70% of the decrease across the MVT for DA3 and about 80% for DUSt3R and VGGT.



(a) Cross-view 3D consistency error (b) Procrustes-aligned error

Figure 5: **Second-view error analysis by layer type on HM3D.** Contribution of each layer type to second-view cumulative error reduction across the MVT for **(a)** e_{mv} and **(b)** e_{proc}. Cross-view attention reduces e_{mv} by bringing corresponding point predictions closer, while single-view attention reduces e_{proc} by regularizing intra-view geometry.

Single-view attention improves non-reference view geometry. After cross-view attention registers non-reference views to the reference frame, single-view attention layers improve their intra-view geometry. Figure 20 illustrates this behavior. We quantify this effect using the accumulated layer contribution for the Procrustes-aligned pointmap error, ALCP($e_{\text{proc}}, \mathcal{F}$), shown in Fig. 5(b) on HM3D probes. Across all models, single-view attention accounts for most of the net reduction across the multi-view transformer, contributing close to or above 100% of the reduction because other layer types can partially increase this error.

Role of MLP layers. Unlike cross-view and single-view attention, MLP layers show no clear geometric role in our probes: on HM3D, they increase the probed reconstruction error on average across all three models (Fig. 16). To test their necessity, we train VGGT-style MVTs on HM3D from scratch with and without MLP layers, and evaluate on held-out scenes. Removing MLP layers degrades performance by only 14.6% while reducing the trainable parameter count by a factor of three (see Table 2). A deeper and wider attention-only MVT with approximately the same parameter count as the baseline outperforms it by 5.6%. It also performs better than pretrained DA3 and VGGT on HM3D, but worse than DUSt3R. These results suggest that MLP layers are not essential for effective 3D reconstruction in this setting, and that reallocating their parameters to attention layers can even improve performance.

Table 2: **MLP ablation on HM3D.** Mean two-view error of the output 3D reconstruction. Parentheses report relative error change from the VGGT† baseline (negative values indicate improvement). Parameter counts and FLOPs exclude the frozen encoder and reconstruction head. FLOPs count matrix operations in the MVT blocks for two 224×224 views.

Variant	Blocks	Width	Parameters	FLOPs	$e_{\text{point}} \downarrow$
VGGT†	12	1024	302.4M	334.0G	0.1202
VGGT† – MLP-free	12	1024	100.8M	124.6G	0.1378 (+14.6%)
VGGT† – MLP-free, scaled	24	1216	285.6M	342.6G	**0.1135** (−5.6%)

† Our VGGT-style re-implementation trained from scratch.

Additional results. Appendix Section B.3 provides full per-layer curves for all metrics and evaluation settings. It shows that the same layer-role trends persist for DUSt3R probes trained and evaluated on BlendedMVS and extend to a four-view VGGT setting. It includes a perturbation analysis that supports the identified role of cross-view attention, and details the MLP ablations and results.

Takeaway. *Multi-view transformers progressively reconstruct 3D scenes by alternating cross-view alignment to the reference view with single-view regularization of non-reference geometry. MLPs have no clear geometric role in our probes; attention-only models match or outperform models with MLPs at comparable parameter counts in small-scale experiments.*

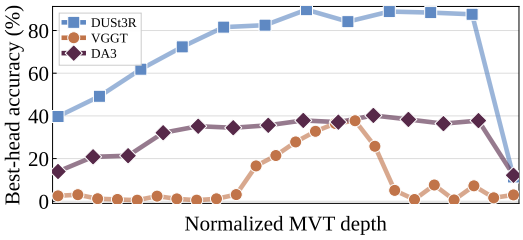


Figure 7: **Correspondence accuracy on HM3D.** Top-1 accuracy of the best-performing head in each second-view cross-view attention layer.

Table 3: **Cross-dataset correspondence accuracy.** Top-1 accuracy of the best-performing second-view cross-view attention head across scene- and object-centric datasets.

	Scene-level	Object-level	
Model	**HM3D**	**GSO**	**DTC**
DUSt3R	89.77%	70.31%	55.06%
VGGT	37.73%	59.33%	47.00%
DA3	40.22%	52.16%	42.77%

4.3 Do Multi-View Transformers Utilize Correspondences?

After establishing that cross-view attention (CVA) aligns points across views, we examine whether image correspondences emerge alongside this alignment. We visualize individual attention heads and find, across all models, heads that consistently attend to candidate matching patches in the other view. These have also been observed in [17]. We refer to these heads as "correspondence heads" and provide detailed DUSt3R visualizations in Appendix Section B.5.

Correspondences turn from semantic to geometric. We first qualitatively examine correspondence heads using the example in Fig. 6. In early DUSt3R MVT blocks, attention maps are consistent with *semantic* correspondences [63]: they match patches with similar appearance or category. After several blocks, DUSt3R refines these matches into *geometric* correspondences [44], attending to patches that observe the same 3D point.

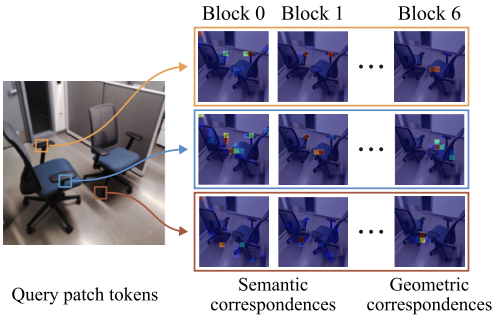


Figure 6: **Semantic-to-geometric refinement in DUSt3R.** Cross-view attention for three query patches shifts from semantically similar regions to geometrically corresponding patches.

Measuring the refinement. We quantify this refinement using the zero-shot correspondence estimator described in Section 3.3. Figure 7 reports top-1 correspondence accuracy on HM3D for the best-performing head of each second-view CVA layer. All models improve through the first half of the MVT, most strongly DUSt3R: its accuracy rises from 40% at the MVT input to nearly 90% after six blocks. VGGT and DA3 start lower and reach nearly 40%. As shown in Table 3, correspondence accuracy in CVA layers also varies with scene type. DUSt3R's advantage over VGGT and DA3 narrows significantly on object-centric datasets: DUSt3R achieves 2.4× VGGT's accuracy on HM3D, compared with only 1.2× on object-centric datasets.

Correspondence ablation. To understand the performance gap between DA3/VGGT and DUSt3R, we study how the pretrained encoder and cross-view attention mechanism affect correspondence accuracy. We train MVTs from scratch following the setup described in Section 3.4. For scene-level evaluation, we train on HM3D and evaluate on held-out scenes. For object-centric evaluation, we train on DTC and evaluate on held-out objects, as well as on GSO objects. The results in Appendix Table 5 show that the pretrained encoder substantially affects zero-shot correspondence accuracy. The CVA mechanism also has a clear, although smaller, impact. In particular, CroCoV2-based models substantially outperform their DINOv2 counterparts, and cross-attention outperforms global attention in all but one setting. Overall, the model most similar to DUSt3R achieves the highest correspondence accuracy across all datasets, whereas the model most similar to VGGT and DA3 achieves the lowest.

Takeaway. *Emergent correspondences are refined across MVT cross-view attention layers. Encoder choice significantly affects this behavior, with CroCoV2 promoting strong correspondences.*

Additional results. Appendix Section B.4 provides correspondence accuracy for all views and heads, as well as visualizations of DUSt3R attention maps. We also include preliminary experiments replacing the attention maps of correspondence heads with ground-truth correspondences.

5 Discussion

Iterative refinement analogy. Our results show that multi-view transformers resemble iterative SfM methods in several respects: geometry progressively improves, network depth acts as a refinement budget, and correspondences emerge and are used during reconstruction. However, MVT updates are computed by separate layers with *distinct* parameters, unlike classical iterative optimization methods that repeatedly apply the *same* update rule. Despite the similarities highlighted in our work, we therefore do not claim that an MVT's forward pass constitutes an iterative optimization procedure. Investigating the extent to which MVT updates can be understood as iterative refinement is a promising direction for future work, particularly in light of recent MVTs that achieve competitive reconstruction performance by reusing the same set of layers across multiple iterations [64].

Implications for future model design. While this work focuses on analysis rather than new model design, our results suggest several concrete directions for improving multi-view transformers. First, our small-scale experiments suggest that MLP layers may not be necessary and that deeper and wider attention-only MVTs can even improve performance. Second, our iterative view of MVTs is closely related to recent parameter-efficient approaches that repeatedly apply a fixed set of layers [64]. Finally, finding ways to use emergent, interpretable behaviors such as correspondences to enhance generic MVTs with explicit inductive biases after training is an interesting direction.

6 Conclusion

We introduced a layer-wise pointmap probing framework for studying how multi-view transformers encode and update 3D geometry during feed-forward reconstruction. Applying this framework to DUSt3R, VGGT, and Depth Anything 3, we found that decoded intermediate reconstructions improve over several layers rather than appearing only at the output. This progression varies across architectures and sample difficulty, suggesting that transformer depth acts partly as a learned refinement budget. Our analysis further shows that this refinement is structured. Cross-view attention primarily improves multi-view consistency by aligning stereo points, while single-view attention restores and refines within-view geometry, especially in regions visible from only one view. We also find that a subset of cross-view attention heads behaves like correspondence matchers, with correspondences becoming more geometric as reconstruction improves. Together, these results show similarities between multi-view transformers and traditional SfM pipelines. They provide a direct view of reconstruction dynamics inside multi-view transformers and suggest that pointmap probing can serve as a practical tool for analyzing future feed-forward 3D and 4D reconstruction models.

Acknowledgments The authors are grateful to Lahav Lipson and Duc-Hai Pham for helpful discussions and for reviewing the manuscript.

VS was supported by the National Science Foundation under Grant No. 2211259, by the Singapore DSTA under DST00OECI20300823 (New Representations for Vision, 3D Self-Supervised Learning for Label-Efficient Vision), by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior/ Interior Business Center (DOI/IBC) under 140D0423C0075, by the Amazon Science Hub, and by the MIT-Google Program for Computing Innovation. AT acknowledges the NVIDIAAcademic Grant Program. MO was supported by the ERC Consolidator Grant 101087347 (VEGA), as well as gifts from Ansys Inc., and Adobe Research. This work was granted access to the HPC resources of IDRIS under the allocation 2025-AD011017075 made by GENCI.

References

[1] Johannes Lutz Schönberger and Jan-Michael Frahm. "Structure-from-Motion Revisited". In: *CVPR*. 2016.

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

[3] Jianyuan Wang et al. "VGGT: Visual Geometry Grounded Transformer". In: *CVPR*. 2025.

[4] Linyi Jin et al. "Stereo4D: Learning How Things Move in 3D from Internet Stereo Videos". In: *CVPR*. 2025.

[5] Junyi Zhang et al. "MonST3R: A Simple Approach for Estimating Geometry in the Presence of Motion". In: *ICLR*. 2025.

[6] Yifan Wang et al. "π^3: Permutation-Equivariant Visual Geometry Learning". In: *ICLR*. 2026.

[7] Nikhil Keetha et al. "MapAnything: Universal Feed-Forward Metric 3D Reconstruction". In: *3DV*. 2026.

[8] Haotong Lin et al. "Depth Anything 3: recovering the visual space from any views". In: *ICLR*. 2026.

[9] Mehdi S. M. Sajjadi et al. "Scene Representation Transformer: Geometry-Free Novel View Synthesis Through Set-Latent Scene Representations". In: *CVPR*. 2022.

[10] Mehdi S. M. Sajjadi et al. "RUST: Latent Neural Scene Representations from Unposed Imagery". In: *CVPR*. 2023.

[11] Hanwen Jiang et al. "RayZer: A Self-supervised Large View Synthesis Model". In: *ICCV*. 2025.

[12] Haian Jin et al. "LVSM: A Large View Synthesis Model with Minimal 3D Inductive Bias". In: *ICLR*. 2025.

[13] Thomas W. Mitchel, Hyunwoo Ryu, and Vincent Sitzmann. "True Self-Supervised Novel View Synthesis is Transferable". In: *ICLR*. 2026.

[14] Evan Kim, Hyunwoo Ryu, Thomas W. Mitchel, and Vincent Sitzmann. "Scaling View Synthesis Transformers". In: *CVPR*. 2026.

[15] Vincent Leroy, Yohann Cabon, and Jerome Revaud. "Grounding Image Matching in 3D with MASt3R". In: *ECCV*. 2024.

[16] Honggyu An et al. "Cross-View Completion Models are Zero-shot Correspondence Estimators". In: *CVPR*. 2025.

[17] Jelena Bratulić, Sudhanshu Mittal, Thomas Brox, and Christian Rupprecht. "On Geometric Understanding and Learned Data Priors in VGGT". In: *ECCV*. 2026.

[18] Takeru Miyato, Bernhard Jaeger, Max Welling, and Andreas Geiger. "GTA: A Geometry-Aware Attention Mechanism for Multi-View Transformers". In: *ICLR*. 2024.

[19] Ruilong Li et al. "Cameras as Relative Positional Encoding". In: *NeurIPS*. 2025.

[20] Xingyu Chen, Yue Chen, Yuliang Xiu, Andreas Geiger, and Anpei Chen. "Easi3R: Estimating Disentangled Motion from DUSt3R Without Training". In: *ICCV*. 2025.

[21] Jisang Han et al. "D^ 2USt3R: Enhancing 3D Reconstruction with 4D Pointmaps for Dynamic Scenes". In: *NeurIPS*. 2025.

[22] Guillaume Alain and Yoshua Bengio. "Understanding intermediate layers using linear classifier probes". In: *arXiv* (2018).

[23] Stanisław Jastrzebski et al. "Residual Connections Encourage Iterative Inference". In: *ICLR*. 2018.

[24] Mor Geva, Avi Caciularu, Kevin Ro Wang, and Yoav Goldberg. "Transformer Feed-Forward Layers Build Predictions by Promoting Concepts in the Vocabulary Space". In: *EMNLP*. 2022.

[25] Guy Dar, Mor Geva, Ankit Gupta, and Jonathan Berant. "Analyzing Transformers in Embedding Space". In: *arXiv* (2023). eprint: 2209.02535.

[26] Raul Molina. "Traveling Words: A Geometric Interpretation of Transformers". In: *arXiv* (2023). eprint: 2309.07315.

[27] Nora Belrose et al. "Eliciting Latent Predictions from Transformers with the Tuned Lens". In: *arXiv* (2023). eprint: 2303.08112.

[28] Nelson Elhage et al. *A Mathematical Framework for Transformer Circuits*. https://transformer-circuits.pub/2021/framework/index.html. Accessed: 2025-09-06. Transformer Circuits Thread. 2021.

[29] Philippe Weinzaepfel et al. "CroCo v2: Improved Cross-view Completion Pre-training for Stereo Matching and Optical Flow". In: *ICCV*. 2023.

[30] Maxime Oquab et al. "DINOv2: Learning Robust Visual Features without Supervision". In: *TMLR* (2024).

[31] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Second. Cambridge University Press, 2003.

[32] Cameron Smith, David Charatan, Ayush Tewari, and Vincent Sitzmann. "FlowMap: High-Quality Camera Poses, Intrinsics, and Depth via Gradient Descent". In: *3DV*. 2025.

[33] Eric Brachmann et al. "Scene Coordinate Reconstruction: Posing of Image Collections via Incremental Learning of a Relocalizer". In: *ECCV*. 2024.

[34] Jianyuan Wang, Nikita Karaev, Christian Rupprecht, and David Novotny. "VGGSfM: Visual Geometry Grounded Deep Structure From Motion". In: *CVPR*. 2024.

[35] Ashish Vaswani et al. "Attention is All you Need". In: *NeurIPS*. 2017.

[36] Alexey Dosovitskiy et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *ICLR*. 2021.

[37] Zhenggang Tang et al. "MV-DUSt3R+: Single-Stage Scene Reconstruction from Sparse Views In 2 Seconds". In: *CVPR*. 2025.

[38] Jianing Yang et al. "Fast3R: Towards 3D Reconstruction of 1000+ Images in One Forward Pass". In: *CVPR*. 2025.

[39] Wonbong Jang, Philippe Weinzaepfel, Vincent Leroy, Lourdes Agapito, and Jérôme Revaud. "Pow3R: empowering unconstrained 3D reconstruction with camera and scene priors". In: *CVPR*. 2025.

[40] Hengyi Wang and Lourdes Agapito. "3D Reconstruction with Spatial Memory". In: *3DV*. 2025.

[41] Qianqian Wang, Yifei Zhang, Aleksander Holynski, Alexei A. Efros, and Angjoo Kanazawa. "Continuous 3D Perception Model with Persistent State". In: *CVPR*. 2025.

[42] Yohann Cabon et al. "MUSt3R: Multi-view Network for Stereo 3D Reconstruction". In: *CVPR*. 2025.

[43] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition". In: *CVPR*. 2016.

[44] Mohamed El Banani et al. "Probing the 3D Awareness of Visual Foundation Models". In: *CVPR*. 2024.

[45] Yunze Man et al. "Lexicon3D: Probing Visual Foundation Models for Complex 3D Scene Understanding". In: *NeurIPS*. 2024.

[46] Guanqi Zhan, Chuanxia Zheng, Weidi Xie, and Andrew Zisserman. "A General Protocol to Probe Large Vision Models for 3D Physical Understanding". In: *NeurIPS*. 2024.

[47] Duolikun Danier, Mehmet Aygün, Changjian Li, Hakan Bilen, and Oisin Mac Aodha. "DepthCues: Evaluating Monocular Depth Perception in Large Vision Models". In: *CVPR*. 2025.

[48] Tyler Bonnen et al. "Evaluating Multiview Object Consistency in Humans and Image Models". In: *NeurIPS*. 2024.

[49] Mateusz Michalkiewicz et al. "GIQ: Benchmarking 3D Geometric Reasoning of Vision Foundation Models with Simulated and Real Polyhedra". In: *ICLR*. 2026.

[50] Yue Chen, Xingyu Chen, Anpei Chen, Gerard Pons-Moll, and Yuliang Xiu. "Feat2gs: Probing visual foundation models with gaussian splatting". In: *CVPR*. 2025.

[51] Jisang Han et al. "Emergent Outlier View Rejection in Visual Geometry Grounded Transformers". In: *CVPR*. 2026.

[52] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. "Similarity of neural network representations revisited". In: *ICML*. 2019.

[53] Fabian Gröger, Shuo Wen, and Maria Brbić. "Revisiting the Platonic Representation Hypothesis: An Aristotelian View". In: *arXiv preprint* (2026).

[54] Santhosh Kumar Ramakrishnan et al. "Habitat-Matterport 3D Dataset (HM3D): 1000 Large-scale 3D Environments for Embodied AI". In: *NeurIPS*. 2021.

[55] Yao Yao et al. "BlendedMVS: A Large-scale Dataset for Generalized Multi-view Stereo Networks". In: *CVPR*. 2020.

[56] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. "Stereo Magnification: Learning View Synthesis using Multiplane Images". In: *SIGGRAPH*. 2018.

[57] Jeremy Reizenstein et al. "Common Objects in 3D: Large-Scale Learning and Evaluation of Real-life 3D Category Reconstruction". In: *ICCV*. 2021.

[58] Laura Downs et al. "Google scanned objects: A high-quality dataset of 3d scanned household items". In: *ICRA*. 2022.

[59] Zhao Dong et al. "Digital Twin Catalog: A Large-Scale Photorealistic 3D Object Digital Twin Dataset". In: *CVPR*. 2025.

[60] Philipp Lindenberger, Paul-Edouard Sarlin, and Marc Pollefeys. "Lightglue: Local feature matching at light speed". In: *ICCV*. 2023.

[61] Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. "Do vision transformers see like convolutional neural networks?" In: *NeurIPS*. 2021.

[62] Mozes Jacobs et al. "Block Recurrent Dynamics in Vision Transformers". In: *ICLR*. 2026.

[63] Alexander C. Berg, Tamara L. Berg, and Jitendra Malik. "Shape matching and object recognition using low distortion correspondences". In: *CVPR*. 2005.

[64] Alessandro Burzio et al. "Déjà View: Looping Transformers for Multi-View 3D Reconstruction". In: *NeurIPS*. 2026.

[65] Maximilian Denninger et al. "BlenderProc2: A Procedural Pipeline for Photorealistic Rendering". In: *Journal of Open Source Software* 8.82 (2023), p. 4901. DOI: 10.21105/joss.04901. URL: https://doi.org/10.21105/joss.04901.

[66] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. "Layer normalization". In: *arXiv preprint arXiv:1607.06450* (2016).

[67] Ilya Loshchilov and Frank Hutter. "Decoupled Weight Decay Regularization". In: *ICLR*. 2019.

[68] Peter H Schönemann. "A generalized solution of the orthogonal procrustes problem". In: *Psychometrika* 31 (1966).

[69] Olga Sorkine-Hornung and Michael Rabinovich. "Least-squares rigid motion using svd". In: *Computing* (2017).

[70] Thao Nguyen, Maithra Raghu, and Simon Kornblith. "Do Wide and Deep Networks Learn the Same Things? Uncovering How Neural Network Representations Vary with Width and Depth". In: *ICLR*. 2021.

[71] Dan Hendrycks and Kevin Gimpel. *Gaussian Error Linear Units (GELUs)*. 2016.

[72] Alex Henry, Prudhvi Raj Dachapally, Shubham Shantaram Pawar, and Yuxuan Chen. "Query-key normalization for transformers". In: *Findings of EMNLP*. 2020.

[73] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. "Going deeper with image transformers". In: *ICCV*. 2021.

[74] Jianlin Su et al. "Roformer: Enhanced transformer with rotary position embedding". In: *Neurocomputing* 568 (2024).

[75] Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. "Vision Transformers Need Registers". In: *ICLR*. 2024.

[76] Jianyuan Wang et al. "VGGT-Ω". In: *CVPR*. 2026.

[77] Wenshan Wang et al. "Tartanair: A dataset to push the limits of visual slam". In: *IROS*. 2020.

[78] Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. "ScanNet++: A High-Fidelity Dataset of 3D Indoor Scenes". In: *ICCV*. 2023.

[79] Thomas Schöps et al. "A Multi-View Stereo Benchmark with High-Resolution Images and Multi-Camera Videos". In: *CVPR*. 2017.

Appendix

Overview of the Appendix

This appendix provides implementation and evaluation details, additional experiments, and qualitative examples supporting the analyses in the main paper.

Section A: Training and evaluation details. We specify the datasets, probe architecture, training procedure, and pointmap metrics (Sections A.2 to A.5). We detail zero-shot representation similarity measure and correspondence evaluation (Sections A.6 and A.7). We finally describe the shared MVT architecture and training setup used in our design ablations (Section A.8).

Section B: Additional results. We first examine probe capacity and probing granularity (Section B.1), then analyze internal-state evolution, convergence by sample difficulty, and the effect of output head training (Section B.2). We provide further quantitative and qualitative evidence for individual layer roles, including cross-dataset and four-view evaluations and the MLP ablation (Section B.3). Finally, we report correspondence analyses and ablations (Section B.4), followed by additional pointmap and attention-map visualizations (Section B.5).

A Training and Evaluation Details.

A.1 Models Studied

We probe models using their official checkpoints, available on HuggingFace:

- DUSt3R: `naver/DUSt3R_ViTLarge_BaseDecoder_512_dpt`
- VGGT: `facebook/VGGT-1B`
- Depth Anything 3 (ViT-G): `depth-anything/DA3-GIANT-1.1`

A.2 Datasets

Figure 8 shows example pairs from each dataset.

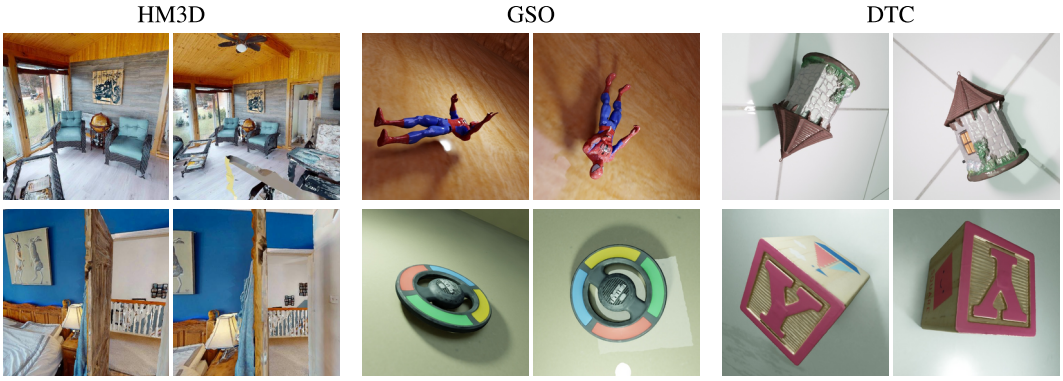


Figure 8: **Example image pairs.** Test pairs from the scene-level HM3D dataset and the object-centric GSO and DTC datasets.

HM3D. We train probes and evaluate the main metrics on image pairs rendered from HM3D scenes [54], using held-out scenes for evaluation.

Object-centric datasets. We render GSO [58] and DTC [59] objects with the same BlenderProc [65] pipeline. Each scene contains a single object placed in a room, rescaled to a bounding-box diagonal of 2.7 scene units. We randomize the textures of the room surfaces, as well as the specularity, roughness, and metallic parameters of all materials. We place an emissive ceiling of random color

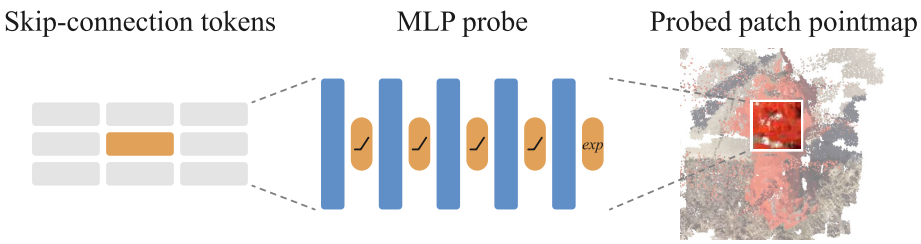


Figure 9: **Probing mechanism.** We train our probe to regress the patch of a pointmap from the corresponding vision transformer patch token. The same probe is shared across all patches in a particular layer of the network.

and intensity and a randomly placed point light. For each object, we render two sequences of five 720×540 views with a fixed focal length of 500 pixels. Cameras lie on a sphere of radius 3 scene units around the object, at elevations between $30°$ and $89°$, and point toward the object with a random roll. We split GSO by object into 540/60/151 train/val/test objects, and DTC by category into 1353/143/360 train/val/test objects.

A.3 Probing Mechanism

Our probing mechanism is illustrated in Fig. 9. Each probe is parameterized as a 5-layer MLP with ReLU activations and input LayerNorm [66].

A.4 Training Details

Training loss. The probes are supervised via a confidence-weighted 3D regression loss introduced in DUSt3R [2]. For a given view, let $\bar{X}$ denote the ground-truth pointmap provided over a subset of valid pixels

$$D \subseteq \{1, \ldots, W\} \times \{1, \ldots, H\}.$$

We first normalize both the predicted and ground-truth pointmaps by their respective scale factors:

$$z = \text{norm}(X) \quad \text{and} \quad \bar{z} = \text{norm}(\bar{X}),$$

where

$$\text{norm}(X) = \frac{1}{|D|} \sum_{(i,j) \in D} \|X_{i,j}\|_2.$$

A per-pixel regression loss is then defined as

$$\ell_{\text{regr}}(i,j) = \left\| \frac{X_{i,j}}{z} - \frac{\bar{X}_{i,j}}{\bar{z}} \right\|_2. \tag{3}$$

The final loss, computed over both views expressed in first-view coordinate system, is given by:

$$L = \sum_{v \in \{1,2\}} \sum_{(i,j) \in D^{(v)}} \left[C_{i,j}^{(v)} \, \ell_{\text{regr}}^{(v)}(i,j) - \alpha \log \left(C_{i,j}^{(v)} \right) \right], \tag{4}$$

where α is a weighting hyperparameter that regularizes the confidence, and $D^{(v)}$ denotes the valid pixels in view v. We use $\alpha = 0.2$ in our experiments. Confidence scores are learned without explicit supervision. Each pixel's confidence value $C_{i,j}^{(v)}$ modulates the loss to reduce the impact of unreliable or ambiguous predictions (e.g., sky or translucent regions).

Optimizer. We use the AdamW optimizer [67] with a learning rate of 5e-4 and a weight decay of 0.01.

Training details. We train our probes for 4 epochs on each dataset on four Nvidia H100 GPUs.

A.5 Pointmap Evaluation

Relative pointmap error We measure pointmap accuracy with an absolute relative 3D error. Let $X_n^{(v)}, \bar{X}_n^{(v)} \in \mathbb{R}^{H\times W\times 3}$ denote the predicted and ground-truth pointmaps for sample n and view $v \in \{1, 2\}$, expressed in the first-view coordinate frame, and let $D_n^{(v)}$ denote the valid pixels. Following the MapAnything evaluation protocol, we first normalize predicted and ground-truth pointmaps by their average valid 3D distance. The per-pixel pointmap error for pixel coordinates $(i,\ j)$ is

$$r_{n,v}(i,j) = \frac{\left\| X_n^{(v)}(i,j) - \bar{X}_n^{(v)}(i,j) \right\|_2}{\left\| \bar{X}_n^{(v)}(i,j) \right\|_2}. \tag{5}$$

We first average over valid pixels within each sample and view,

$$e_{\text{point}}(n,v) = \frac{1}{|D_n^{(v)}|} \sum_{(i,j)\in D_n^{(v)}} r_{n,v}(i,j), \tag{6}$$

and then average these per-sample errors over the dataset. When reporting a global two-view error, we average the two view errors for each sample before averaging over samples.

Procrustes alignment To evaluate pointmaps up to a rigid transformation ambiguity, we align predicted and ground-truth pointmaps $\bar{X}, X$ using a Procrustes alignment [68], following [32, 69]. This corresponds to solving the following weighted Procrustes problem:

$$\arg\min_{P\in SE(3)} \left\| W^{1/2}\big(X - P\,\bar{X}\big) \right\|_F^2, \tag{7}$$

where W is a diagonal weight matrix. We use uniform weights, as we do not observe a significant improvement when using the confidence weights predicted by our probes. We report the resulting aligned error as e_{proc}.

Difficulty subsets We define sample difficulty from the final reconstruction error of each pretrained model. For each test image pair, we first evaluate the final pretrained reconstruction and compute the per-sample pointmap error e_{point} against ground truth. Since evaluation uses symmetrized image pairs, we compute this error in both directions and average the two values to obtain one scalar difficulty score per sample. We then rank test samples by this score and define the easy and hard subsets as the bottom and top 20% of the error distribution, respectively. On our 500-pair test set, each subset contains 100 samples.

A.6 Measuring Representation Similarity

We measure the linear Centered Kernel Alignment (CKA) [52] by treating each patch token in the dataset as an individual sample. Instead of reporting a debiased per-batch CKA estimate as in previous work [70], we compute the exact CKA over the entire dataset via online accumulation across batches. We report the CKA after the calibration procedure described in [53] to remove the confounding effect of feature dimensionality, which can induce spurious correlations.

To do so, we first compute the uncalibrated CKA score s by accumulation over the entire dataset. During this process, for every pair of MVT layers, we randomly sample and store 500 patch tokens for calibration. For a given pair, we then use these samples to compute a calibration threshold $\tau_{\alpha=0.5}$, and compute the calibrated score following their procedure as:

$$s_{\text{cal}} = \max\left(\frac{s - \tau_{\alpha=0.5}}{1 - \tau_{\alpha=0.5}},\ 0\right)$$

A.7 Zero-shot Correspondence Extraction and Evaluation

Following [16], we evaluate whether MVT cross-view attention maps can serve as zero-shot correspondence estimators. For each cross-view attention layer ℓ, attention head h, and direction $a \to b$, we denote the corresponding attention map by $A_{\ell,h}^{a\to b}$. For a source patch p^a, the top-1 predicted match is the target patch with maximum attention:

$$\hat{p}_{\ell,h}^{b} = \arg\max_{p^b} A_{\ell,h}^{a\to b}(p^a, p^b).$$

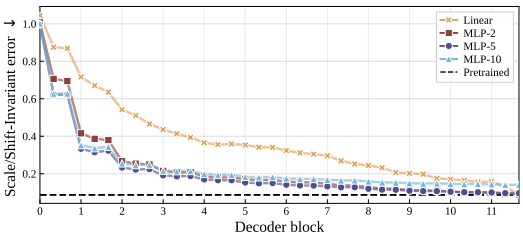


Figure 10: **Ablation study on probing head strength.** We measure e_{point} at the outputs of the MVT blocks for the second view, comparing different probing head architectures and capacities.

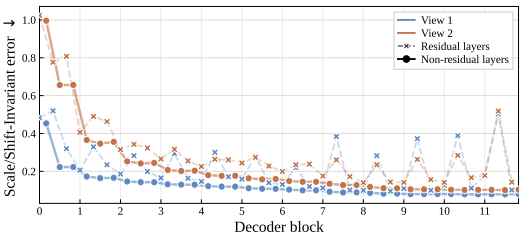


Figure 11: **Probing granularity ablation.** We measure e_{point} on HM3D. We plot the error at the output of residual layers (dashed) and after skip connections (solid). The error decreases almost monotonically after skip connections, while other operations exhibit more irregular variations.

Ground-truth patch correspondences. Ground-truth pointmaps are projected across views using the camera poses and intrinsics. We then compute symmetric mutual correspondence masks with a tolerance of $\tau = 5$ pixels, since we find smaller thresholds to be too aggressive on HM3D, possibly due to rendering artifacts. We convert these pixel-level correspondences to patch-level correspondences by voting. For each source patch, all pixels in that patch vote for the target patch containing their projected correspondence. Any target patch with at least three votes is considered as a valid match. This one-to-many mapping reflects the ambiguity of the patch-level correspondence matching task, with several patches forming potentially valid matches.

Correspondence evaluation. For each valid source patch, layer, head, and direction, we measure top-k matching accuracy: a prediction is correct if one of the ground-truth target patches appears among the k most-attended target patches. We favor this metric over distance threshold metrics, since we evaluate coarse correspondences at the patch level. We evaluate $k \in \{1, 3, 5\}$ and aggregate counts over all valid query patches in the evaluation set. For DUSt3R, we apply the first-token heuristic of [16]: if the first target token is the most-attended token for a query, its attention score is set to zero before computing the top-k ranking. For VGGT and DA3, we first split the global attention map into within-view and cross-view blocks, and evaluate only the cross-view block. We report accuracies separately for both directions, yielding one accuracy score per layer and head. We summarize each layer by reporting its best-performing head.

A.8 MVT Design and Ablations

Shared architecture. We implement the MVT design space ablation described in Section 3.4 in a unified codebase, varying the main components listed in Table 1: pretrained encoder, cross-view attention mechanism, and view-wise weight sharing. The standard configuration closely follows VGGT but uses a smaller architecture. It uses 12 alternating-attention blocks of width 1024 with 16 attention heads. Each block contains single-view attention followed by cross-view attention, with each attention operation followed by a two-layer MLP with GELU activations [71]. We use pre-LayerNorm [66], query/key normalization [72], and LayerScale [73] initialized to 0.01. Rotary position embeddings [74] are applied to single-view attention but omitted from cross-view attention, following MUSt3R [42]. Our implementation naturally supports any number of views.

Encoders and registers. We use pretrained CroCoV2 or DINOv2 ViT-L encoders and keep their weights frozen. Four learned register tokens [75] are concatenated to each view's token sequence, with separate parameters for the reference and non-reference views even when MVT weights are shared. As noted in VGGT, these distinct register tokens act as a reference view encoding and enable to identify the reference view even when MVT and output head weights are shared.

Output head and 3D parameterization. Unless otherwise specified, we use a linear output head applied to the final block output. We also implement a dense multilayer convolutional head adapted from VGGT-Omega [76]. It aggregates block outputs at relative depths $0.25, 0.5, 0.75, 1.0$, rather than using only the final state. By default, we use DUSt3R's 3D output parameterization and predict

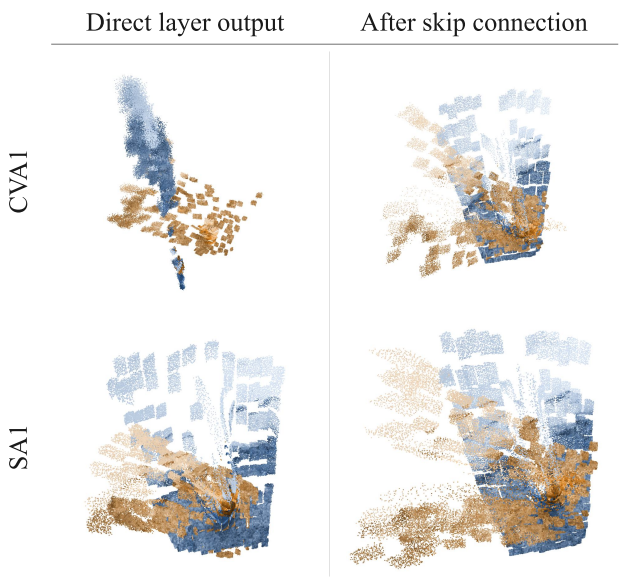


Figure 12: **Visual comparison of pointmaps** probed after skip connections vs direct layer outputs. We observe that probing after skip connections yields clearer and more interpretable geometry.

pointmaps in the first-view coordinate frame and their associated confidence maps. We also implement VGGT's output parameterization with pointmaps, depth, and poses.

Training details. We train at a resolution of 224×224, using a batch size of 16. We use DUSt3R's confidence-weighted pointmap regression loss described in Eq. (4). We also implement VGGT's loss and support training with an arbitrary number of views. The MVT and output head are randomly initialized and trained with the AdamW [67] optimizer, using a base learning rate of 10^{-4}, weight decay of 0.01, and gradient-norm clipping at 1. Each stage lasts 10 epochs, with half an epoch of linear warmup followed by cosine learning-rate decay to 10^{-6}.

B Additional Results

B.1 Ablations

Probe capacity. We find that using non-linear probes yields considerably cleaner and more interpretable pointmaps compared to a linear head. This also results in a faster decrease of e_{point} over the blocks, as reported in Figure 10. Furthermore, we find that using more than 5 layers does not affect the results.

Probing granularity. We verify the hypothesis of skip connections as a state carrier by probing the direct outputs of every individual layer before skip connections. Figure 11 shows the comparison of e_{point} when probed from the skip connection versus when probed directly at the output of the layer. The error after skip connections is clearly more stable. Our visualization in Figure 12 also shows that the direct outputs are less interpretable by pointmap probing. This indicates that the individual layers alter the state in a way that does not necessarily have clear geometric meaning, while the state update from the skip connection does.

B.2 Evolution of the internal state

Full pointmap error curves. In Section B.3, we provide the full layer-wise pointmap curves used to analyze individual layer roles. We report the HM3D results used in the main text, DUSt3R probes trained and evaluated on BlendedMVS, HM3D-trained probes evaluated zero-shot on the MapAnything benchmark datasets (TartanAirv2 [77], ScanNet++ [78], and ETH3D [79]), and a

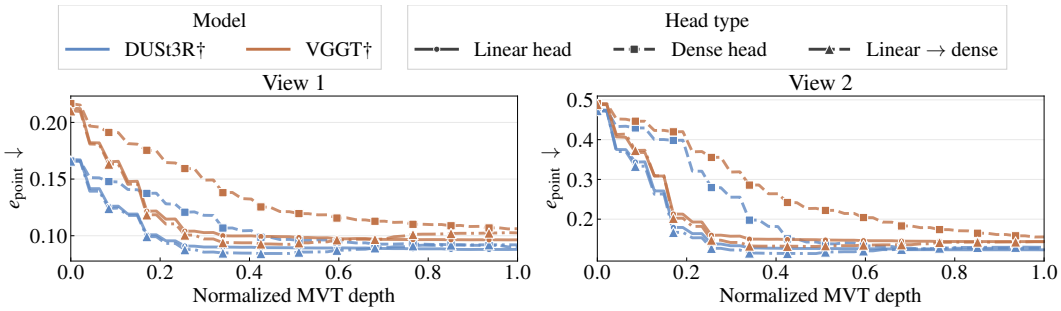


Figure 13: **Output head design affects reconstruction dynamics.** Probed pointmap error e_{point} on HM3D. Linear-head training and the linear-to-dense curriculum yield fast error reduction, particularly in non-reference view 2. Meanwhile, training with a dense head froms scratch leads to slower reduction in reconstruction error. † denotes our re-implementations with MVTs trained from scratch.

four-view VGGT setting trained and evaluated on HM3D. These additional settings test whether the same layer-wise trends persist across datasets and beyond the two-view setup.

Centered kernel alignment measure. In Figure 14, we report Centered Kernel Alignment (CKA) scores between MVT blocks for several multi-view transformers, computed over the validation set after the calibration procedure described in Section A.6. As computing similarities between all pairs of layers is expensive, we visualize similarities at the block-level by measuring similarity between the last layer of each block.

Convergence by sample difficulty. Figure 15 compares convergence across blocks toward the final probe reconstruction for DUSt3R, VGGT, and DA3, using all samples and the easiest and hardest 20% defined in Section A.5. Across models and views, easy samples converge faster on average. VGGT and DA3 do not converge before the final layer at the 5% error threshold, with second-view inlier ratios below 60%.

Effect of output head on error dynamics. We study how output head design and training curriculum affect layer-wise reconstruction dynamics using the DUSt3R-style and VGGT-style MVTs. The DUSt3R-style variant combines CroCoV2, cross-attention, and binocular MVTs with separate per-view attention weights and output heads. The VGGT-style variant combines DINOv2, global attention, and shared attention weights and output heads. Apart from these differences, both use the same alternating-attention implementation described in Section A.8. We compare training with a linear head applied to the final layer, training from scratch with a dense head that aggregates multiple layers, and a linear-to-dense curriculum (Section A.8). In the curriculum, we retain the MVT weights obtained at the end of the first stage, discard the final-layer head, and initialize a fresh multilayer head. We then train for 10 additional epochs with a fresh optimizer and learning-rate schedule, while keeping the encoder frozen. We freeze each of the six trained checkpoints and train probes using the protocol described in the main text. We train MVTs and probes on 100,000 HM3D image pairs at a resolution of 224×224, and evaluate on held-out HM3D scenes.

Figure 13 shows that, across both architectures, training with a final-layer head yields a faster reduction in probed pointmap error across MVT depth, especially for the non-reference view. When training with a final-layer head, the error approaches its final value within the first few MVT blocks. In contrast, training with a multilayer head from scratch yields a slower decrease, particularly for the VGGT-style variant. Training first with a final-layer head and then with a multilayer head preserves the rapid error reduction across MVT depth. These results explain the differences in error reduction speed across MVT depth observed between DUSt3R and VGGT/DA3 in Section 4.1.

B.3 Role of Individual Layers

Quantitative analysis. We present detailed results for the layer-role analysis in the main text. For each setting, we report three errors: e_{point}, the per-pixel relative pointmap error defined in Equation (5); e_{proc}, the Procrustes-aligned variant described in Section A.5; and e_{mv}, the cross-view error

Table 4: **Attention perturbation in DUSt3R.** Increase in multi-view consistency error e_{mv} on HM3D relative to the unperturbed model when the attention maps of a single layer are mixed with a uniform map of strength α. We report the mean and the maximum increase across blocks for each layer type.

	Layer type	$\alpha = 0.05$	$\alpha = 0.1$	$\alpha = 0.25$	$\alpha = 0.5$	$\alpha = 1$
Mean	Cross-view attention	+3.1%	+9.1%	+31.2%	+75.9%	+448.4%
	Single-view attention	+0.4%	+1.4%	+6.5%	+19.8%	+68.3%
Max	Cross-view attention	+28.1%	+75.0%	+227.2%	+463.3%	+3497.7%
	Single-view attention	+2.6%	+8.8%	+35.4%	+92.1%	+264.0%

computed between predictions from different views corresponding to the same underlying 3D point. For each metric $e \in \{e_{\text{point}},\ e_{\text{proc}},\ e_{\text{mv}}\}$, we report the error across layers for all models, and the Accumulated Layer Contribution Percentage aggregated by layer type (Eq. (1)), separately for the reference view and the non-reference view(s).

We use the same layout for four evaluation settings. First, Figure 16 reports the main two-view HM3D setting: probes for DUSt3R, VGGT, and DA3 are trained on HM3D and evaluated on held-out HM3D scenes. This is the setting used for the main layer-role claims. Second, Figure 17 reports DUSt3R probes trained and evaluated on BlendedMVS, which tests whether the same decomposition of layer roles appears on a real multi-view stereo dataset. Third, Figure 18 reports the HM3D-trained probes evaluated zero-shot on the MapAnything benchmark datasets. Finally, Figure 19 reports VGGT probes trained and evaluated on HM3D with four input views, where the reference-view probe is applied to view 1 and the non-reference probe is shared across views 2–4. This setting checks whether the two-view observations persist when a feed-forward MVT receives more views.

Attention perturbation. To further isolate the roles of single-view and cross-view attention in DUSt3R, we perturb one attention layer at a time during inference. Each attention map of the perturbed layer is mixed with a uniform attention map with strength α: $\alpha = 0$ leaves the model unchanged, while $\alpha = 1$ fully replaces the attention map and removes the layer's ability to select tokens. We then measure the increase in multi-view consistency error e_{mv} on HM3D relative to the unperturbed model. Table 4 reports this increase averaged across blocks and its maximum increase.

Across all perturbation strengths, perturbing cross-view attention causes substantially greater degradation than perturbing single-view attention, with effects 5.03 to 13.25 times larger on average. Even a small perturbation of $\alpha = 0.05$ increases the error by up to 28.1% for a cross-view layer, compared with at most 2.6% for a single-view layer. At larger perturbations, the most influential cross-view layer is in block 0, consistent with our probing results identifying it as important for multi-view alignment. These interventions provide evidence that multi-view consistency depends on selective cross-view attention: disrupting its ability to retrieve informative tokens from the other view increases e_{mv} far more than an equivalent single-view attention perturbation.

Effect of MLP layers on reconstruction. We test whether the MVT MLPs are necessary for reconstruction using the VGGT-style architecture described in Section A.8. We compare this baseline with a variant that removes the MLP layers after both single-view and cross-view attention while retaining the same depth and width. Excluding the frozen encoder and reconstruction head, this reduces the parameter count from 302.4M to 100.8M. We additionally train a a MLP-free variant that matches the baseline parameter budget approximately by increasing MVT depth from 12 to 24 MVT blocks and embedding dimension from 1024 to 1216 channels. All three models are trained from scratch on 100,000 HM3D image pairs at 224×224 resolution with the setup described in Section A.8. We evaluate their output 3D reconstructions on held-out HM3D scenes.

Table 2 shows that removing MLPs at fixed depth and width increases pointmap error by only 14.6% while using one-third as many trainable parameters. Increasing the depth and width of the MLP-free model to approximate the baseline parameter count improves error by 5.6% relative to the baseline. This suggests that reallocating MLP capacity to attention can improve reconstruction in this small-scale setting.

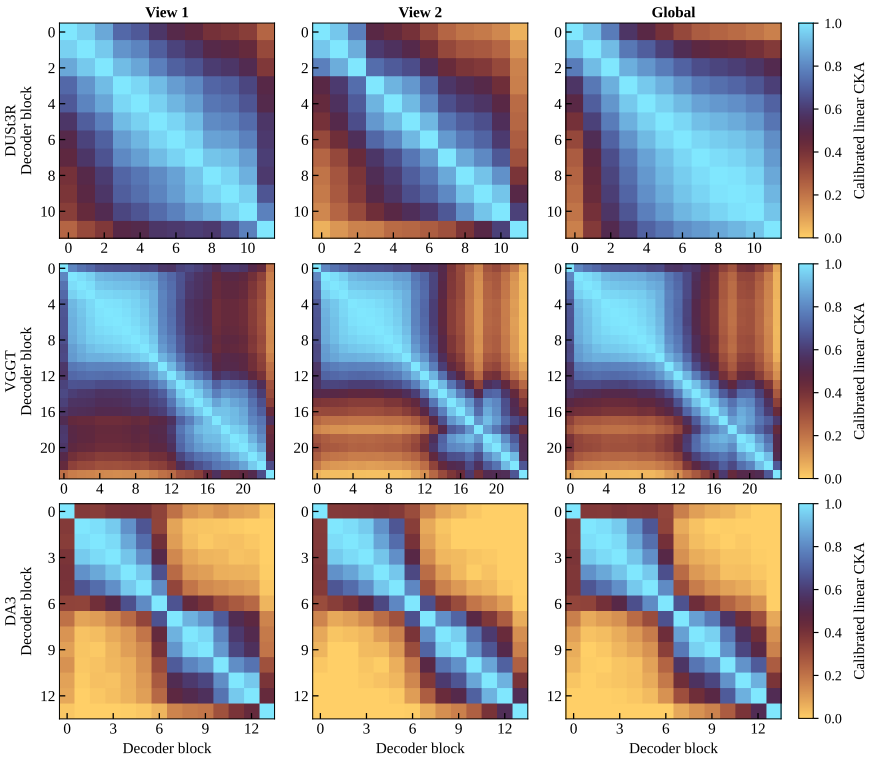


Figure 14: **Centered Kernel Alignment (CKA) between MVT blocks.** Pairwise calibrated CKA scores between the last layer of each MVT block. Results are computed over the validation set and reported separately for features from view 1 and view 2, as well as jointly over both views ("Global"), where features from both views are concatenated prior to CKA computation.

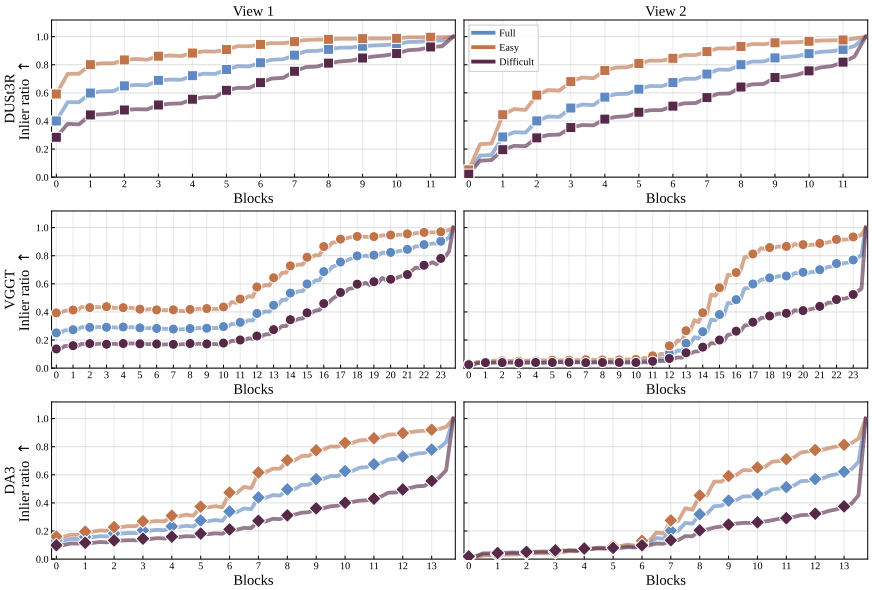


Figure 15: **Convergence by sample difficulty.** Inlier ratio within 5% relative error of the final probe reconstruction, averaged over the full dataset or each model's easiest and hardest 20% of test samples.

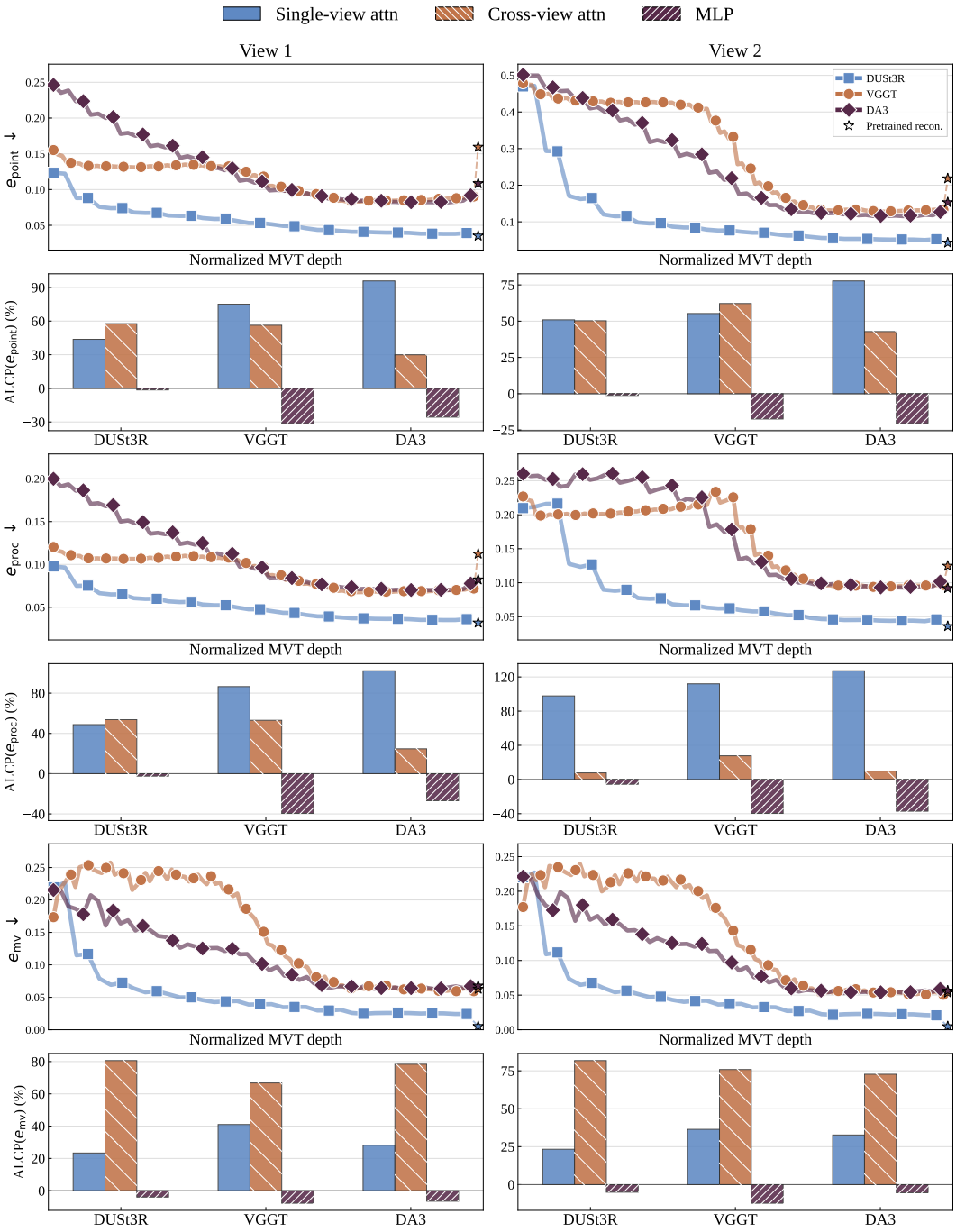


Figure 16: **Layer-role analysis on HM3D.** Probes for DUSt3R, VGGT, and DA3 are trained and evaluated on HM3D. Rows pair, for each metric, the error across MVT layers for all models with the Accumulated Layer Contribution Percentage aggregated by layer type.

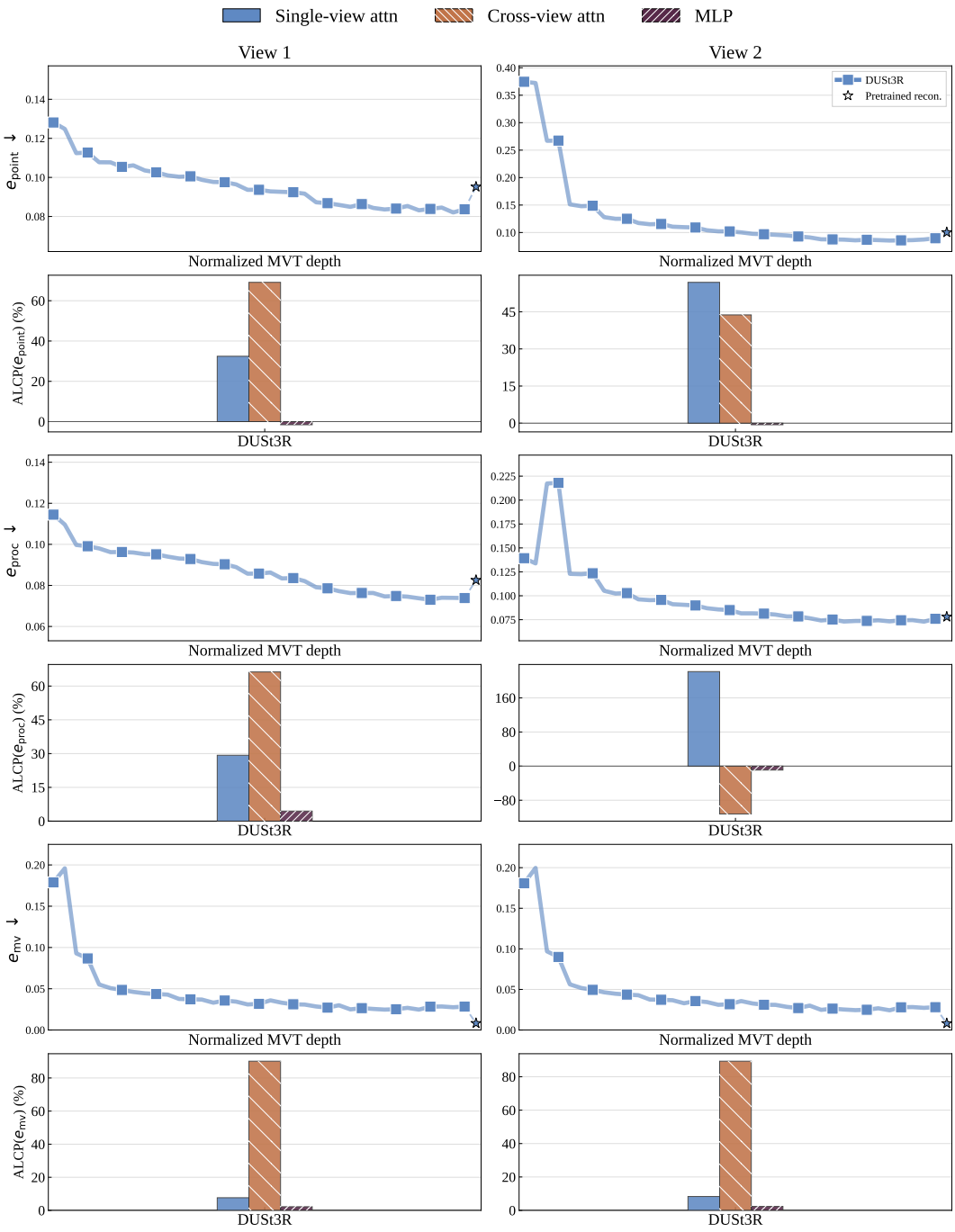


Figure 17: **Layer-role analysis on BlendedMVS.** DUSt3R probes are trained and evaluated on BlendedMVS. Rows pair, for each metric, the error across MVT layers for all models with the Accumulated Layer Contribution Percentage aggregated by layer type.

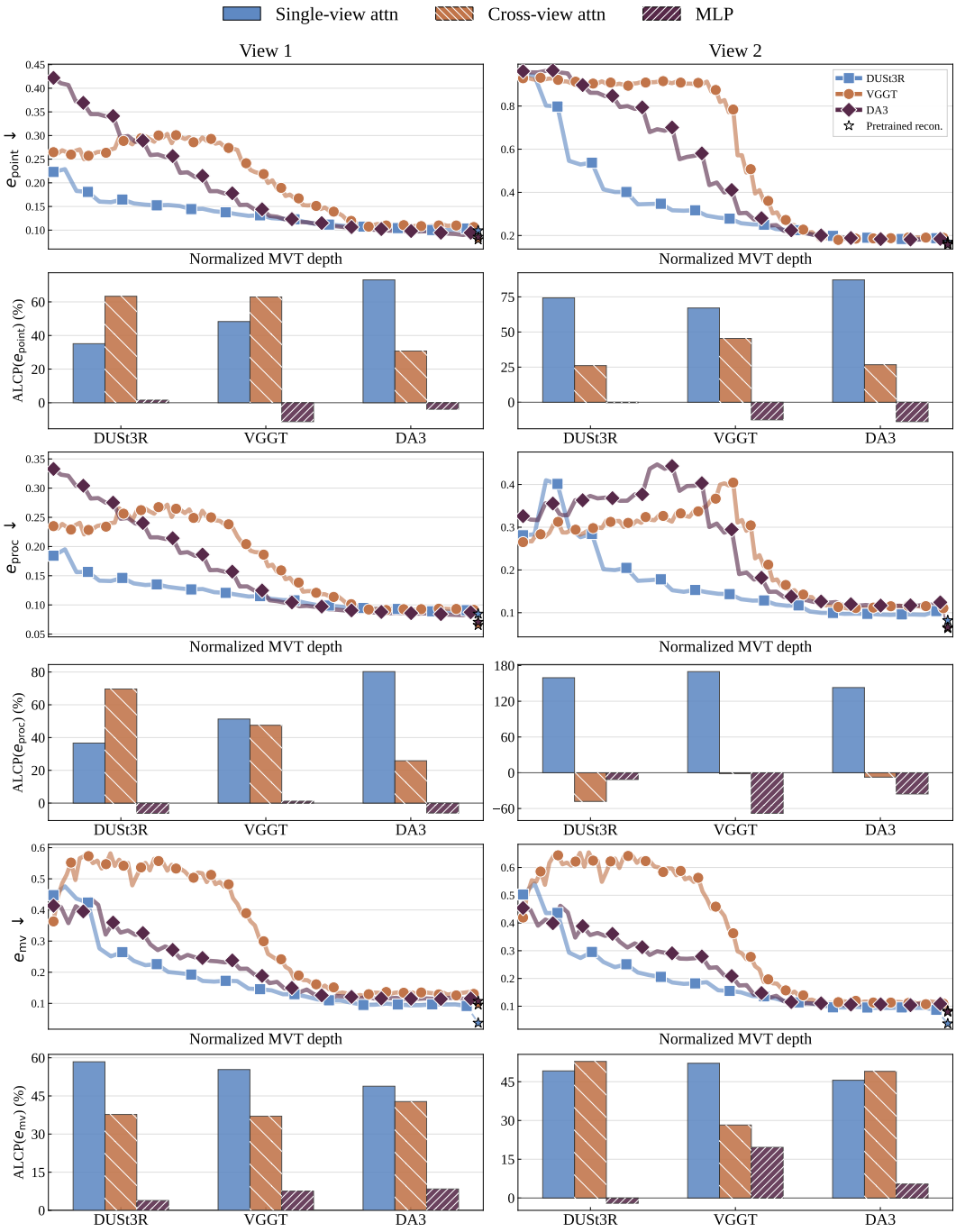


Figure 18: **Zero-shot layer-role analysis on the MapAnything benchmark.** Probes trained on HM3D are evaluated zero-shot on the MapAnything benchmark datasets (TartanAirv2 [77], ScanNet++ [78], and ETH3D [79]). Rows pair, for each metric, the error across MVT layers for all models with the Accumulated Layer Contribution Percentage aggregated by layer type.

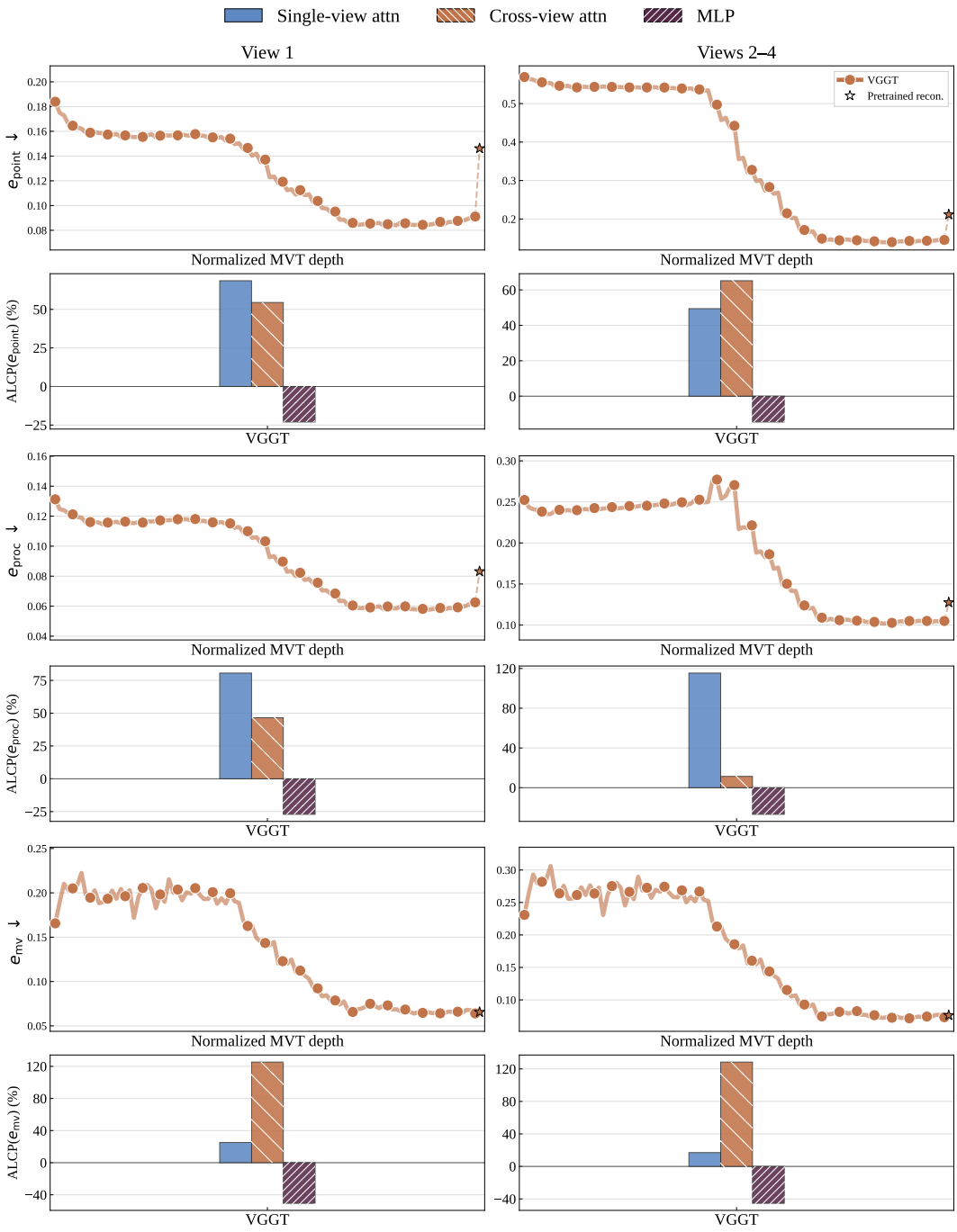


Figure 19: **Layer-role analysis for four-view VGGT on HM3D.** VGGT probes are trained and evaluated with four HM3D views; the right column averages views 2–4. Rows pair, for each metric, the error across MVT layers for all models with the Accumulated Layer Contribution Percentage aggregated by layer type.

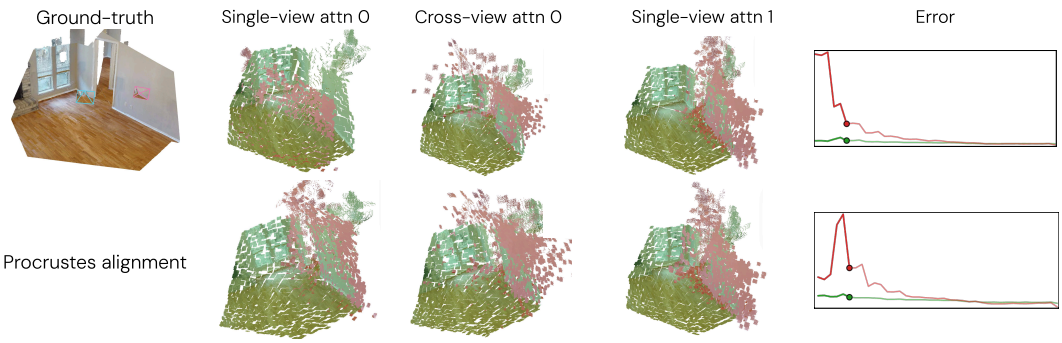


Figure 20: **Probed pointmaps in DUSt3R early MVT blocks.** We visualize pointmaps predicted after the first single-view and cross-view attention operations. Results are shown both before and after Procrustes alignment. Points from view 1 are colored in green, and points from view 2 in red. Cross-view attention aligns the two views but disrupts the geometric consistency within view 2. Subsequent single-view attention restores intra-view geometric structure.

Qualitative analysis In Figure 20, we illustrate the role of cross-view and single-view attention with a visualization of an example probed from DUSt3R. We show intermediate reconstructions from early MVT layers, starting before any cross-view communication occurs. For each layer, we display two versions of the predicted pointmap: the raw probe output and the version after Procrustes alignment with the ground truth. As described in Section 3.1, this alignment procedure allows us to evaluate the geometric consistency of the second view independently of the relative pose error introduced by imperfect alignment with the first view during early reconstruction stages. We also report the L2 error before Procrustes alignment for each view with respect to the ground truth for each pointmap sequence.

The initial reconstruction of the first view is already accurate and exhibits low error. Probe outputs indicate that this reconstruction remains largely unchanged across subsequent layers and effectively acts as a stable anchor. In contrast, the initial reconstruction of the second view is not aligned with the coordinate system of the first view. However, the Procrustes-aligned sequence reveals that the second-view pointmap already captures the scene geometry (e.g., the rigidity of the wall). The first cross-view attention layer then registers the second view with the first, substantially reducing the error in the direct probe output. Examining the same transformation in the Procrustes-aligned sequence shows that it temporarily disrupts geometric consistency in the second view (for instance, the rigidity of the wall) resulting in a sharp increase in Procrustes-aligned error. The subsequent single-view attention operation restores the internal geometric structure of the second view, leading to a decrease in Procrustes-aligned error.

These observations illustrate the complementary roles played by the two attention mechanisms. Cross-view attention aligns the second view with the first through non-rigid updates, while single-view attention subsequently improves internal geometric consistency.

B.4 Role of correspondences

Per-head correspondences. For every model, we report the per-head correspondence accuracy Figure 21, following the methodology described in the main text. Accuracy is aggregated over all valid query patches in the evaluation set. We find that correspondences for the second view improve across the first half of the MVT for all models. In particular, the correspondence accuracy is significantly higher for DUSt3R than for other models. We confirm this trend and report the best-performing head in Figure 22 for varying top-k thresholds.

Encoder ablation.

Replacing attention heads with GT correspondence maps. In DUSt3R, we replace cross-view attention maps by the ground-truth correspondence map for patch tokens with a valid correspondence in the other view. We apply this only for attention heads that perform correspondence search. To automatically find these heads, we use the per-head correspondence accuracy Fig. 21 and modify

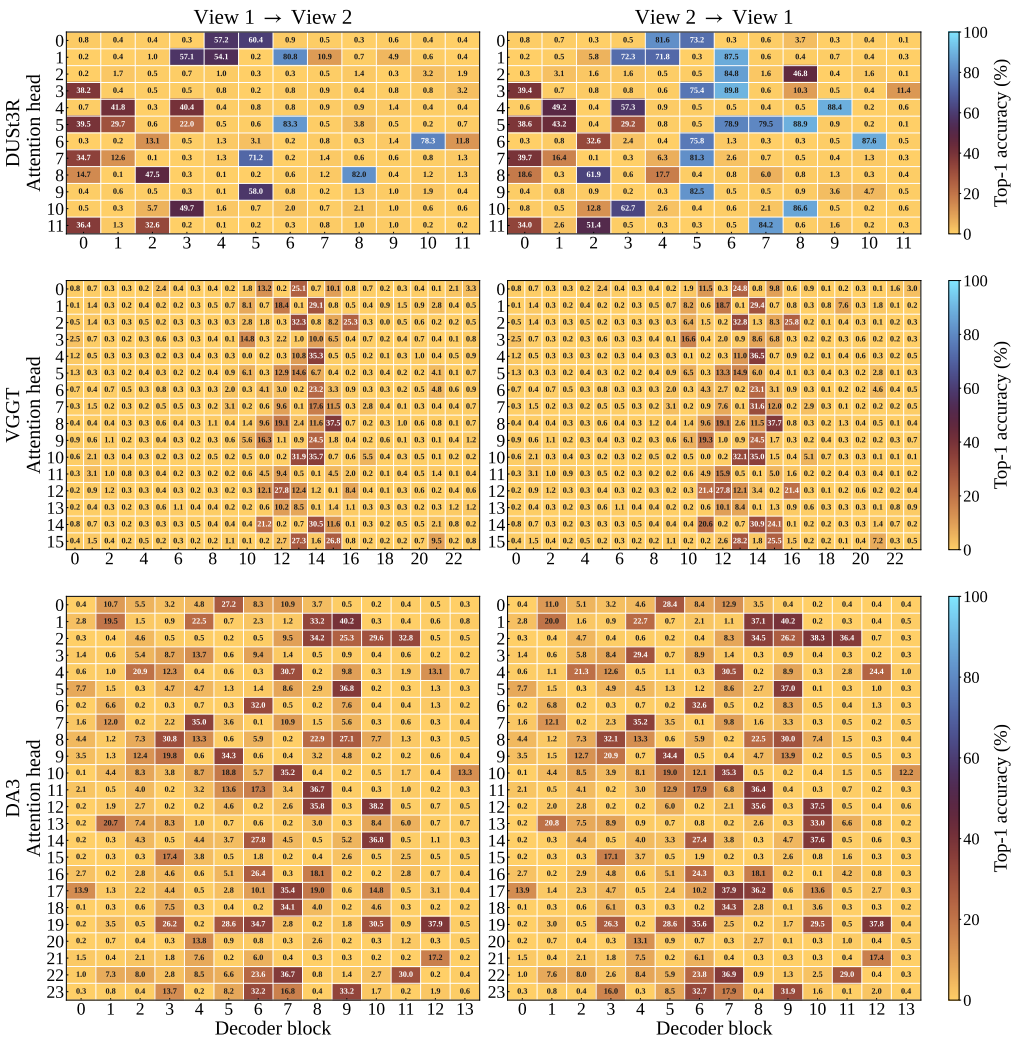


Figure 21: **Per-head correspondence accuracy.** Top-1 correspondence accuracy (%) of every cross-view attention head on HM3D, for both attention directions. Correspondences improve across MVT blocks until the middle of the MVT for all models. DUSt3R presents the highest accuracy across all models.

Table 5: **MVT correspondence ablation.** Top-1 View 2 → View 1 accuracy of the best-performing attention head. CA and GA denote cross- and global attention.

	Scene-level	Object-level	
Model	**HM3D**	**GSO**	**DTC**
CroCoV2 + CA	**73.93%**	**55.53%**	**52.74%**
CroCoV2 + GA	67.92%	50.48%	50.13%
DINOv2 + CA	40.06%	35.45%	32.54%
DINOv2 + GA	43.40%	21.89%	21.97%

the heads whose accuracy is higher than a threshold. We choose 2% as the threshold. We then evaluate the modified model and report the results compared to the original DUSt3R Fig. 23. We explore replacing heads in both view streams, as well as in each view stream separately. We find that replacing the heads in either view stream separately does not affect the result significantly. However, a slight decrease in performance is observed when replacing both.

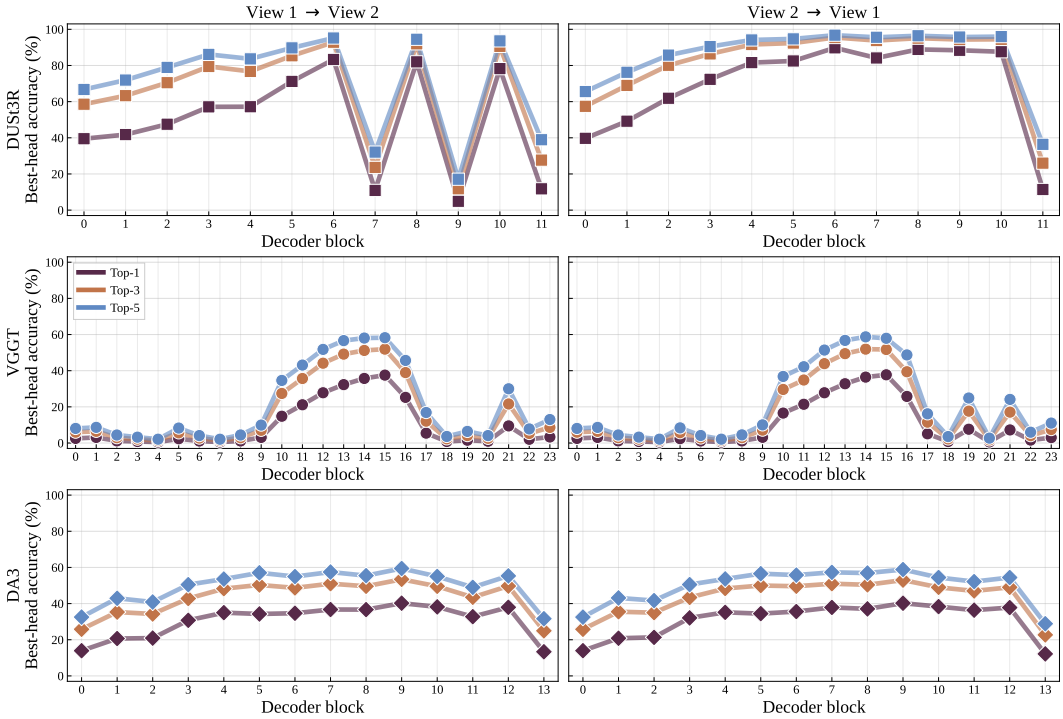


Figure 22: **Best correspondence accuracy.** Top-1, top-3, and top-5 correspondence accuracy of the best cross-view attention head in each decoder block on HM3D, for both attention directions.

B.5 Additional Visualizations

Pointmap visualizations. Additional pointmap visualizations for early MVT blocks are provided in Figure 24, along with colored refinement examples in Figure 25.

Self-attention maps visualizations We visualize self-attention maps in the second MVT block of the second view for the `hydrant` example in Figure 26. We selected this layer because it contributed most to reducing the second-view pointmap error after Procrustes alignment. We identify heads that mainly attend to neighboring patches of the query and others that consistently attend to a fixed subset of patch tokens across all queries. We refer to these heads as *register heads* and these subsets of tokens as *register tokens*, which likely correspond to the high-norm tokens identified in [75]. Note that our use of this term differs from their paper, where it refers to additional tokens introduced to overcome this phenomenon. This behavior repeats across samples and can also be observed in other MVT blocks; however, we show only one block on a single example for brevity.

Cross-attention maps visualizations We visualize cross-attention maps in the first MVT block for the `hydrant` example in Figure 27. Similar to self-attention, the attention maps indicate a specialization of individual heads for particular tasks. Noticeably, heads 3, 5, 7, 8, and 11 attend to candidate corresponding patches for the query, whenever correspondences exist. The behavior of these correspondence heads is consistent across samples. We also observe heads attending to *register tokens*, as in the self-attention map. Besides, we notice that cross attention maps tend to be significantly sharper than self-attention ones.

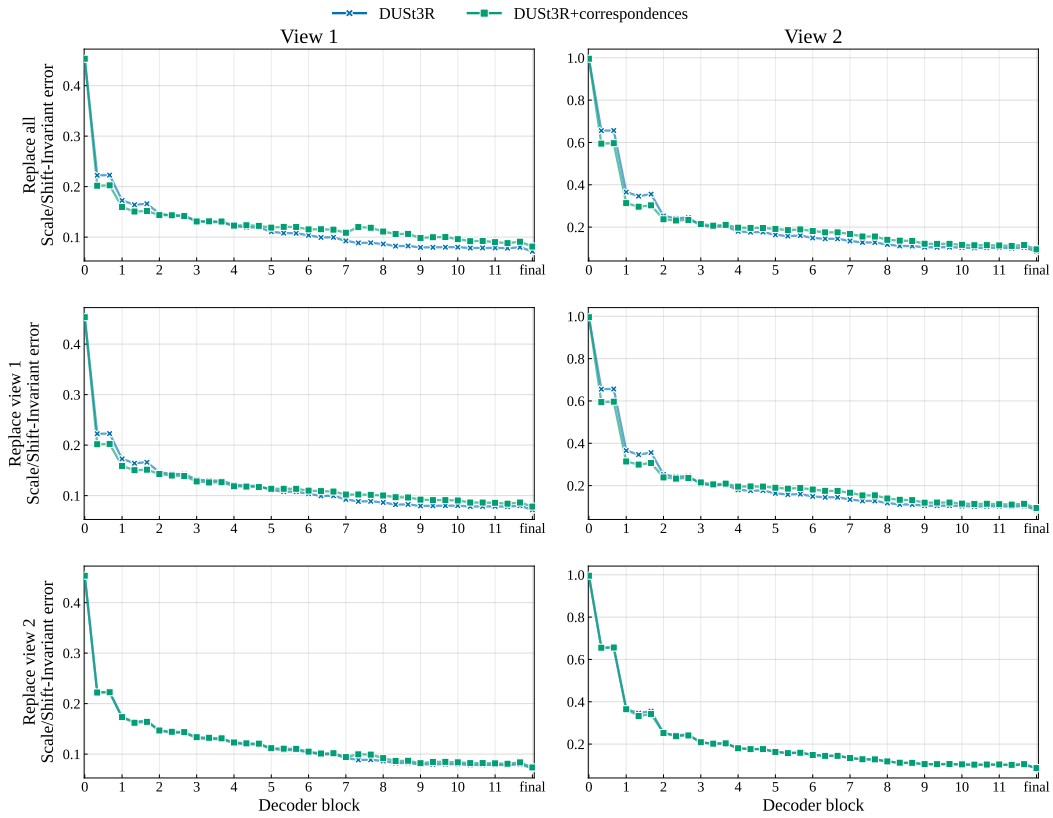


Figure 23: **Replacing cross-view attention with correspondences.** We replace DUSt3R cross-view attention maps with ground-truth correspondences for heads that perform correspondence search.

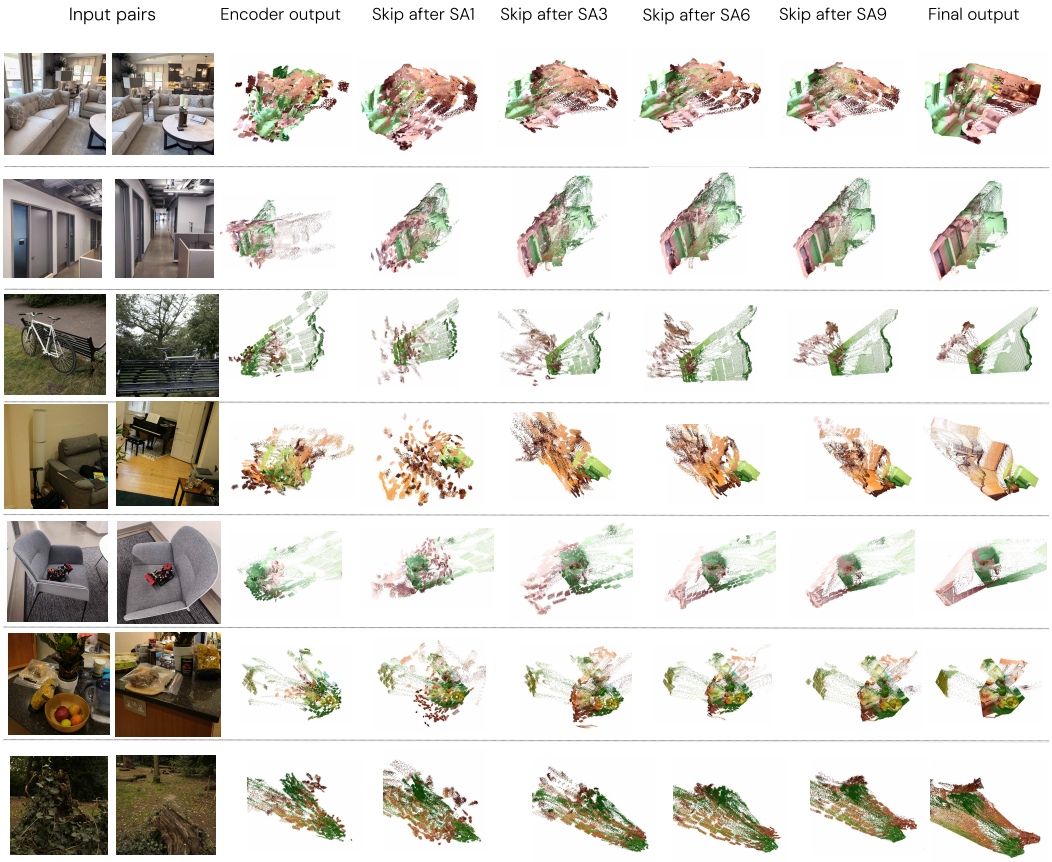


Figure 24: **Additional pointmap visualizations.** The two simpler top scenes (room and corridor) converge after a single cross- and self-attention iteration. The four more challenging bottom cases (bicycle, couch corner, chair with Lego, counter) require 3 to 6 iterations to correctly align the second-view patches (red). In the final opposing-view example (stump), reconstruction fails, likely due to early convergence to a local minimum.

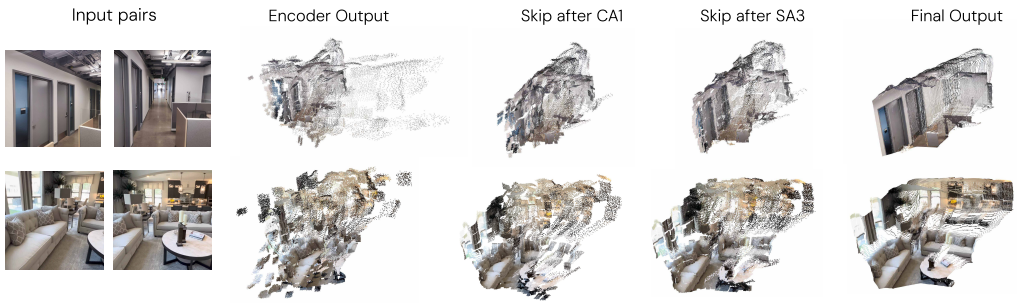


Figure 25: **Additional visual examples with RGB colors**.

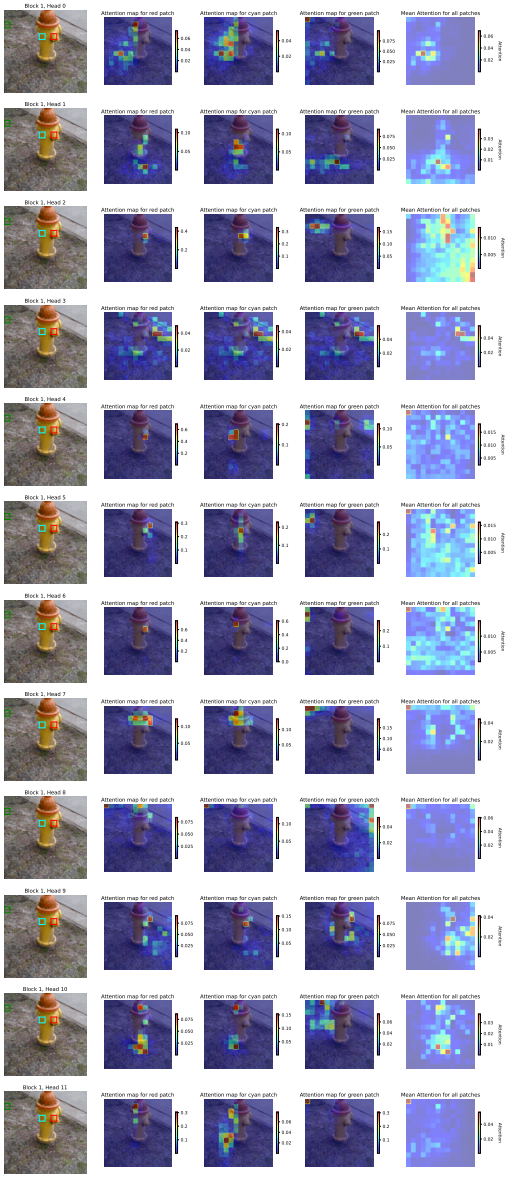


Figure 26: **Visualization of DUSt3R self-attention maps** in the second MVT block of the second view. We visualize the attention maps for three query patches shown on the query image. Heads 1, 2, 4, 5, 6, and 11 mainly attend to local patches around the query; heads 0, 3, 8, and 9 are identified as attending to *register tokens*. The locations of these registers vary across examples.

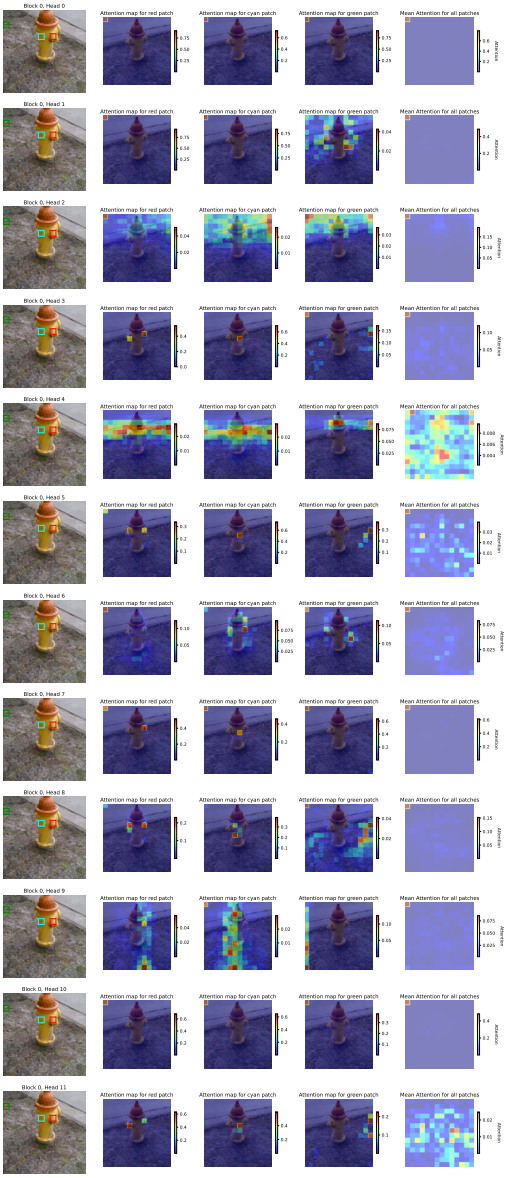


Figure 27: **Visualization of DUSt3R cross-attention maps** in the first MVT block of the second view. We visualize the attention maps for three query patches shown on the query image. Heads 3, 5, 7, 8, and 11 act as correspondence heads, while heads 0, 3, 8, and 10 consistently attend to the top-left *register token*. Heads 9 and 4 exhibit spatial behavior, performing vertical and horizontal searches, respectively.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the paper's scope, contributions, assumptions, and empirical findings, which are supported by the experiments and limitations discussed in the paper.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We include a discussion and limitations section at the end of the paper.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide detailed explanations of our evaluation setup in the main paper and appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code will be made open source upon publication.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: We rely on standard datasets and explain their sources. We provide details for our optimizers and hyperparameters.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We averaged our results across three runs, but did not observe meaningful variation across runs. We believe this is due to probing frozen pretrained networks with dense objectives on a large number of samples, which smooths differences.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We have provided approximate execution times as well as type and number of compute workers used for our experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: We have reviewed the code of ethics and conformed to it to the best of our knowledge.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: We see no direct malicious applications for our work.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We do not release new models or data.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We own our code and have used and credited models and data according to their licenses and terms of use.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The released code is well documented for easy use and adoption.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs as important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.