LUND WORKSHOP

Learning to Understand 3D

David Nordström

Chalmers University of Technology

LoMa · RoMa v2 · RoMa-Ω

MuM · Orochi

FRAMING

Language converged on one objective. Vision did not.

NLP

Next-token prediction. One objective, one recipe. Everything else is scale and data.

VISION, AND 3D IN PARTICULAR

No convergence. Competing paradigms, each with its own supervision, its own data, its own failure modes.

So: what is the right way to **learn to understand 3D**?

Right now there are three serious answers.

I'll walk through all three, and argue that the third is the only one that **scales**.

AGENDA

Three ways to get 3D out of images

01

Optimization pipelines with learned parts

COLMAP and friends. Keep the geometry, replace the hand-crafted pieces with learned ones.

LoMa · RoMa v2

02

Feed-forward reconstruction

Skip the pipeline. Regress geometry directly, trained on a mountain of 3D labels.

RoMa-Ω

03

Self-supervision, 3D as a by-product

No labels at all. Learn from raw image sequences and let 3D understanding emerge.

MuM · Orochi

Most of the time goes to **part three**, because I think self-supervision is our only scalable, general route to 3D.

PART 1

Optimization pipelines, with learned parts

The classical Structure-from-Motion pipeline is still the most accurate 3D reconstruction we have. The question is which of its pieces should be learned.

- COLMAP and the SfM pipeline
- LoMa: scaling local features
- RoMa v2: going dense

The SfM pipeline: matching is the learnable part



▲ **learned** the rest stays classical geometry and optimization

WHY KEEP THE PIPELINE

Geometric verification and bundle adjustment give guarantees a network does not. COLMAP is still the accuracy reference.

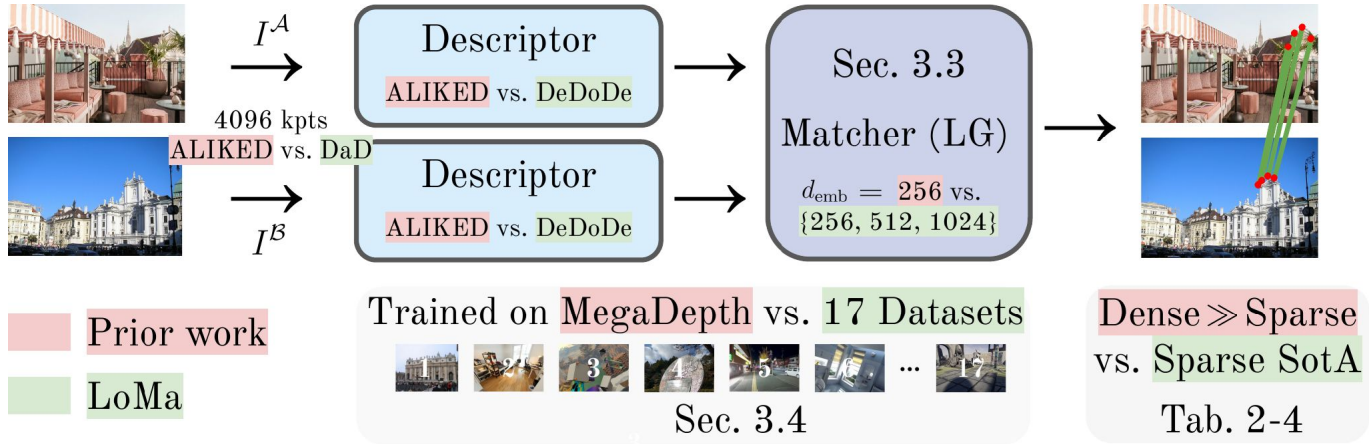
WHY IT'S WORTH IMPROVING

The front end, meaning detection, description and matching, is where the failures live. Everything downstream inherits them.

A better matcher makes every downstream stage better **without touching the geometry**.

LoMa: local features were written off too early

Detector-free and feed-forward methods overtook local features, but the comparison was never fair. They train on orders of magnitude more data.



DETECT + DESCRIBE

ALIKED → DaD + DeDoDe, retrained

MATCH

LightGlue-style, 3 sizes: B / L / G

DATA

MegaDepth → 17 datasets, our own MVS ground truth

LoMa: Local Feature Matching Revisited ECCV 2026 Short Oral

D. Nordström*, J. Edstedt*, G. Bökman, J. Astermark, A. Heyden, V. Larsson, M. Wadenbäck, M. Felsberg, F. Kahl

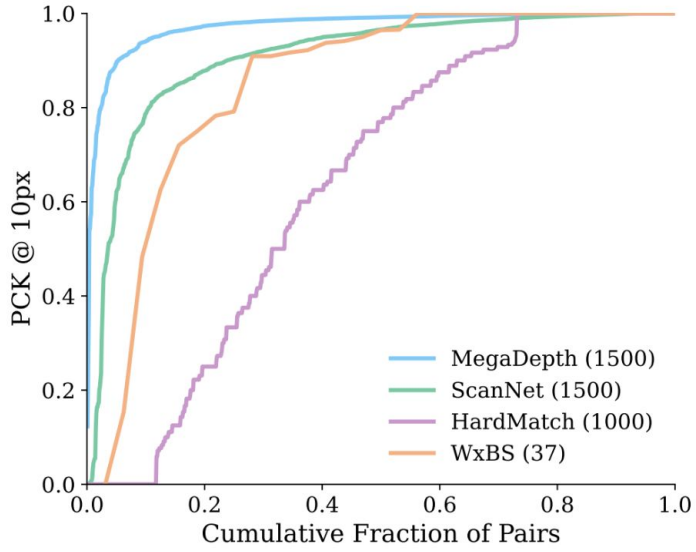
github.com/davnords/loma

HardMatch: super difficult matching

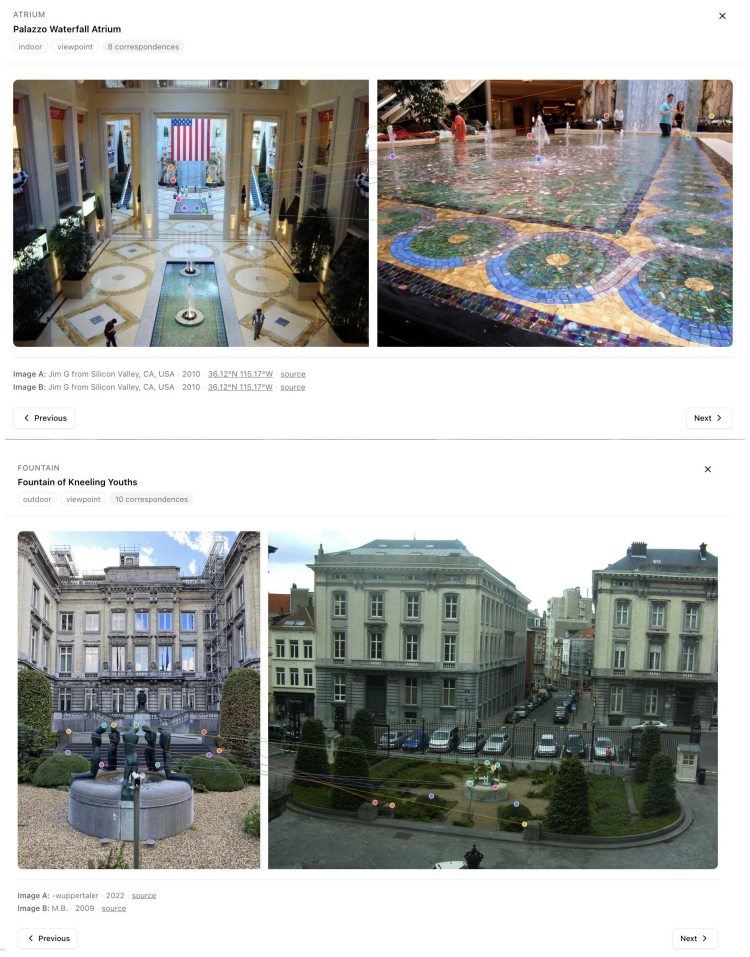
MegaDepth and ScanNet-1500 are saturated



(a) Matches on a pair from HardMatch.



(b) Distribution of pairs for different benchmarks successfully matched by SotA models.



LoMa: Local Feature Matching Revisited ECCV 2026 Short Oral

D. Nordström*, J. Edstedt*, G. Bökman, J. Astermark, A. Heyden, V. Larsson, M. Wadenbäck, M. Felsberg, F. Kahl

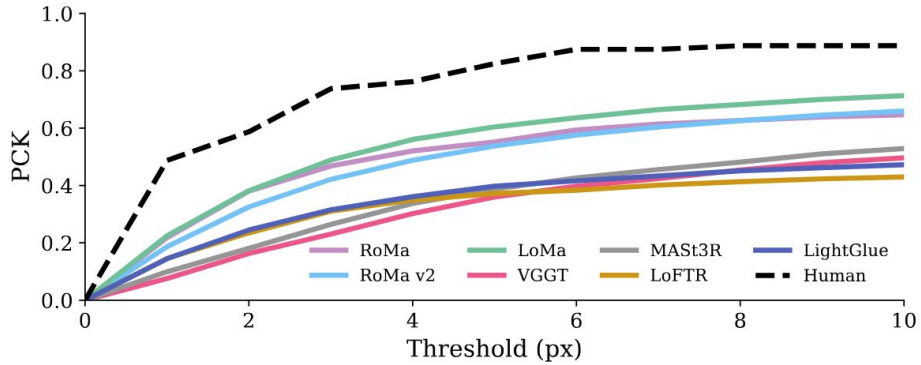
github.com/davnords/loma

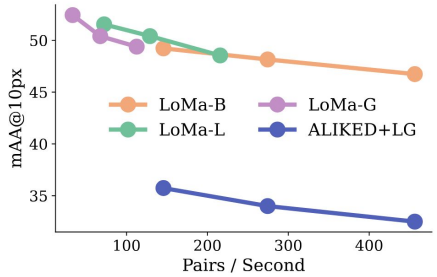
Scaled local features beat the dense matchers

+18.6
HardMatch
mAA vs LightGlue

73.4
WxBS
mAA, beats RoMa

89.3
IMC 2022
mAA, new SotA





HARDMATCH mAA@10PX, BY GROUP

A sparse matcher beating RoMa. 73.4 mAA on WxBS, the first time a local feature matcher has taken the hardest wide-baseline benchmark off the dense models.

And it drops straight into COLMAP. Same interface as SuperPoint+SuperGlue, so it plugs into HLoc and COLMAP unchanged, giving +14 to +20 points on InLoc, Map-free and Oxford Day-and-Night.

HardMatch. We also had to build a benchmark of 1,000 hand-annotated pairs, because MegaDepth is saturated.

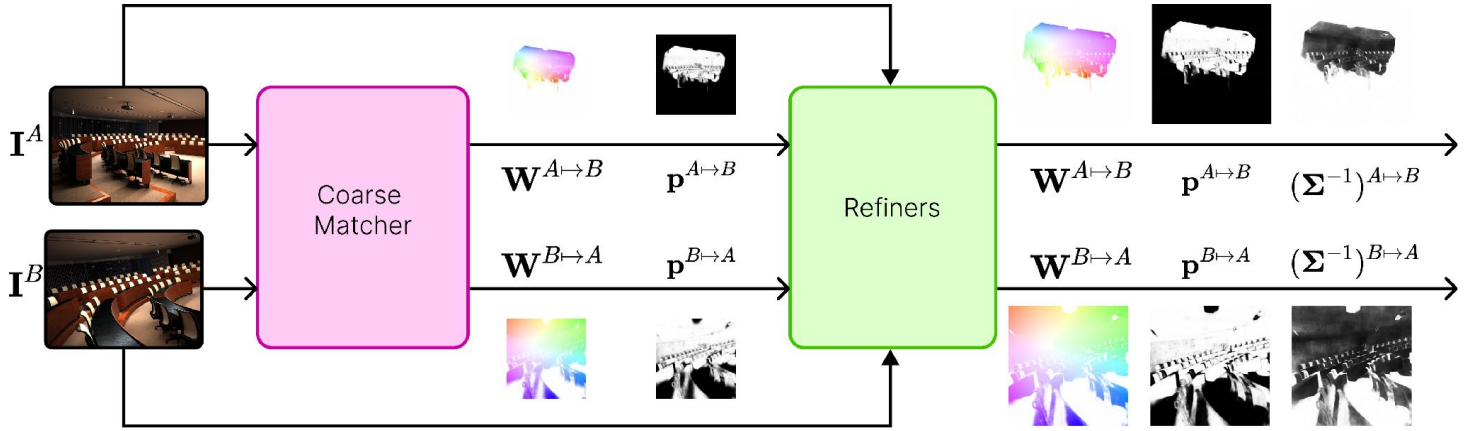
LoMa: Local Feature Matching Revisited ECCV 2026 Short Oral

D. Nordström*, J. Edstedt*, G. Bökman, J. Astermark, A. Heyden, V. Larsson, M. Wadenbäck, M. Felsberg, F. Kahl

github.com/davnords/loma

RoMa v2: drop the detector, match every pixel

One step further from the classical pipeline, with no keypoints at all. A coarse matcher predicts a dense warp, refiners sharpen it to sub-pixel.



COARSE

Frozen DINOv3 + transformer decoder + DPT head

REFINE

Convolutional refiners, custom CUDA kernel

OUTPUT

Dense warp + confidence, not a sparse match list

RoMa v2: Harder Better Faster Denser ECCV 2026 Short Oral

J. Edstedt, D. Nordström, Y. Zhang, G. Bökman, J. Astermark, A. Heyden, V. Larsson, F. Kahl, M. Wadenbäck, M. Felsberg

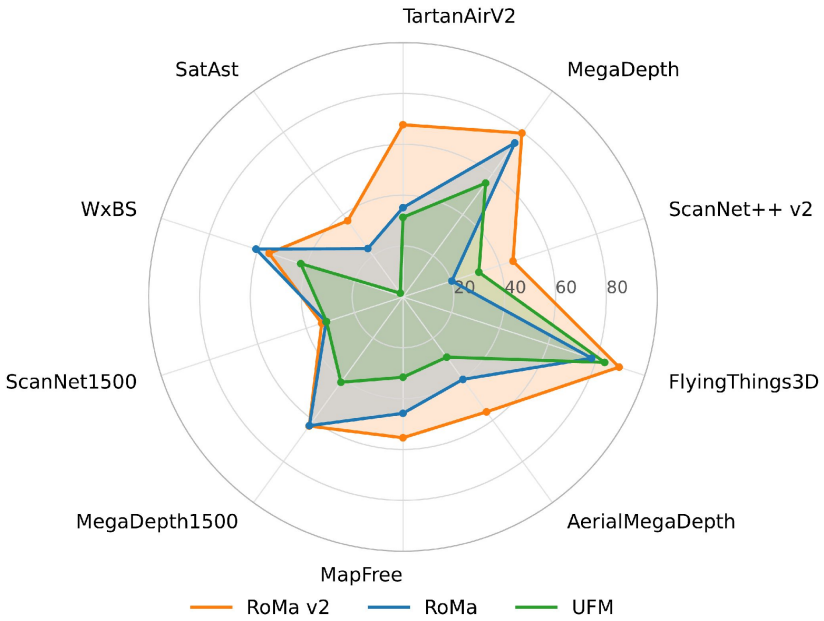
github.com/Parskatt/RoMaV2

Denser, more precise, and faster than its predecessor

Harder. SatAst 23.5 → 37.0 AUC, RUBIK 47.3 → 57.3% success.

Better. EPE on TartanAir-WB 60.6 → 13.8, and 1px accuracy 35 → 68.

Faster. Decoupled two-stage training and custom CUDA kernel, giving ~1.7× throughput vs. RoMa.



The one thing it lost: **robustness to extreme modality change**. RoMa still wins on WxBS. Hold that thought.

RoMa v2: Harder Better Faster Denser ECCV 2026 Short Oral

J. Edstedt, D. Nordström, Y. Zhang, G. Bökman, J. Astermark, A. Heyden, V. Larsson, F. Kahl, M. Wadenbäck, M. Felsberg

github.com/Parskatt/RoMaV2

PART 2

Feed-forward reconstruction

Throw the pipeline away. Train one network on an enormous pile of 3D labels and read the geometry straight out of a forward pass.

- DUSt3R → VGGT → VGGT-Ω
- RoMa-Ω: what do these models know about matching?
- The data loop problem

The pipeline, replaced by a single forward pass



DUSt3R → MASt3R → VGGT → VGGT-Ω, π³, DA3. **No detector, no matcher, no bundle adjustment.**

WHAT YOU GAIN

Speed, simplicity, and a model that keeps improving as you add data.

WHAT IT COSTS

Supervision. Every training sample needs depth maps and camera poses.

These models are trained for geometry. So: **what do they know about matching?**

Every state-of-the-art matcher freezes the same backbone

IMAGE MATCHERS

RoMa, RoMa v2 and LoMa are all built on frozen DINO features. Their robustness is routinely credited to DINO.

FEED-FORWARD 3D

VGGT, VGGT-Ω, π³ and DA3 are trained on geometry, at scale, and now with matching losses in the mix.

What do feed-forward 3D models know about **image matching**?

RoMa-Ω: What Feed-Forward 3D Models Know About Matching ECCV 2026 SfM-DL Workshop

D. Nordström, X. Zhang, T. Loiseau, V. Lepetit, F. Kahl

github.com/davnords/RoMa-Omega

What feed-forward 3D models know about image matching

01 **Raw similarity is misleading.** Later-layer features look collapsed under cosine similarity. However, a simple linear probe recovers strong correspondence information.

02 **Raw 3D predictions already match.** Just using VGGT-Ω's depth + pose predictions gives state-of-the-art performance on ScanNet-1500. However, falls apart under viewpoint and modality shift.

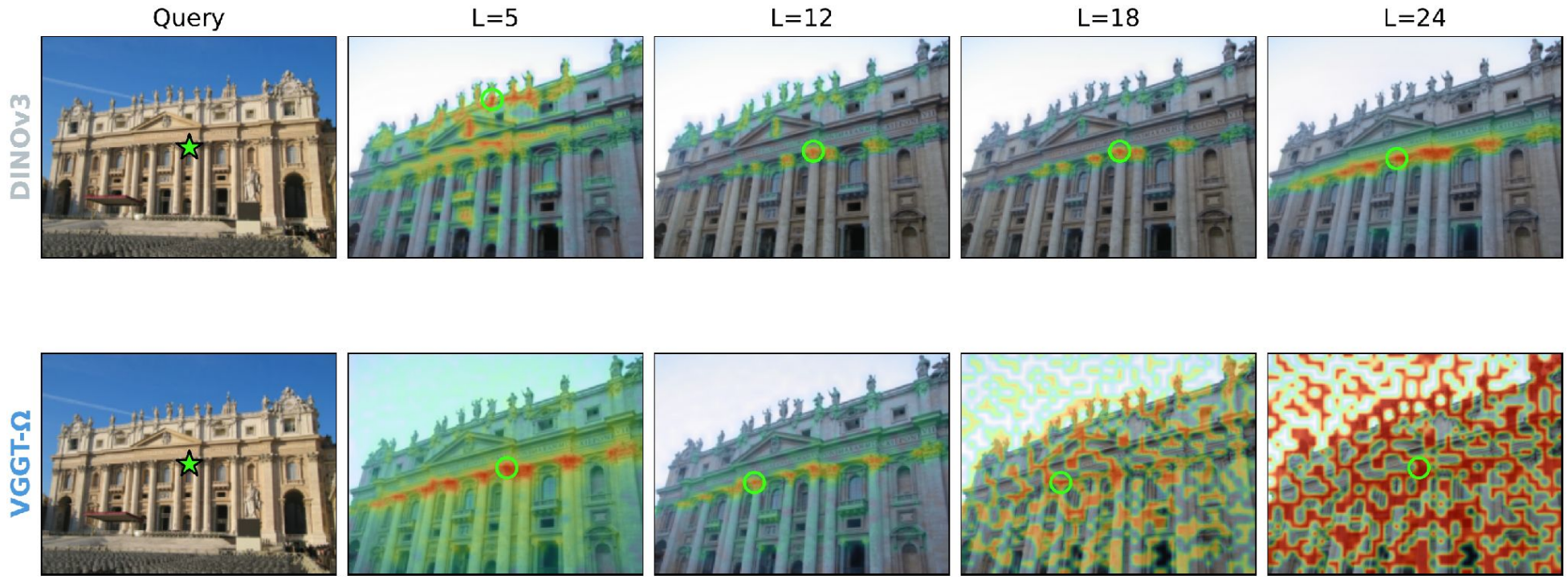
03 **So train a matcher on top.** DINOv3 → VGGT-Ω inside RoMa v2 gives state of the art on a wide range of benchmarks, e.g.: +8.1 WxBS, +3.5 HardMatch, +2.4 RUBIK.

RoMa-Ω: What Feed-Forward 3D Models Know About Matching ECCV 2026 SfM-DL Workshop

D. Nordström, X. Zhang, T. Loiseau, V. Lepetit, F. Kahl

github.com/davnords/RoMa-Omega

Reconstruction features degrade with depth



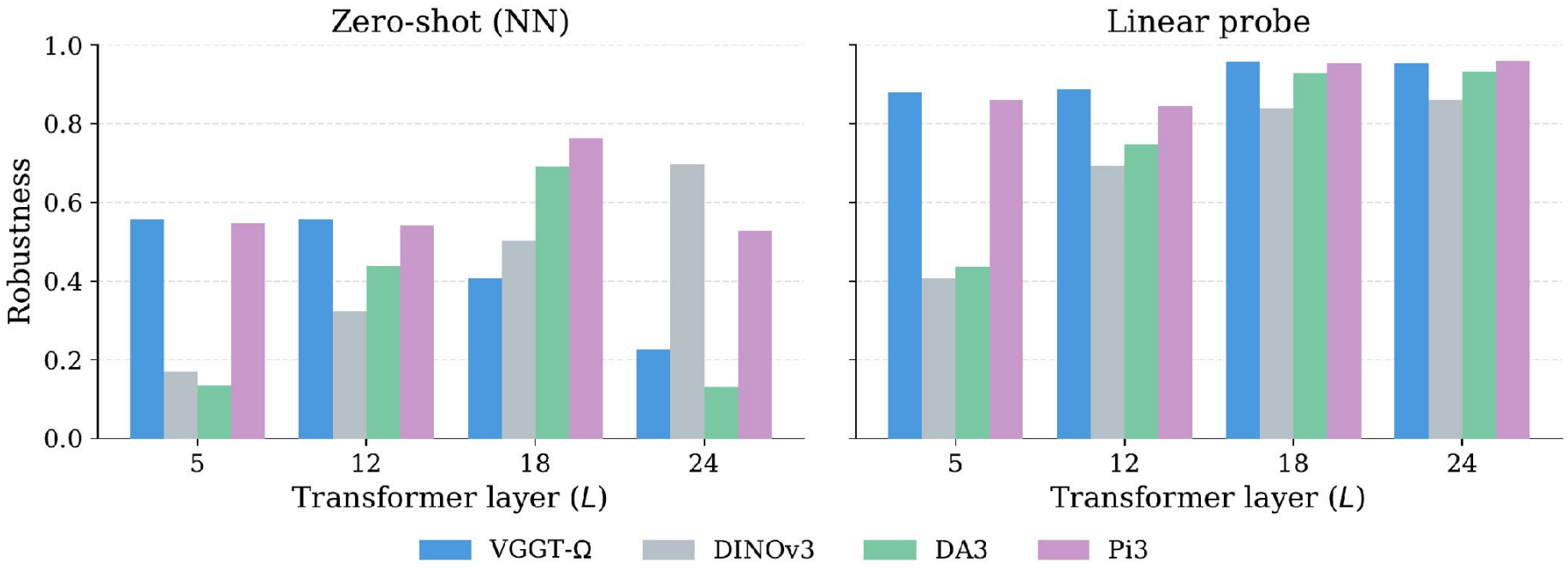
Correlation maps get **diffuse with depth**. DINOv3 sharpens instead. π³ and DA3 behave like VGGT-Ω.

RoMa-Ω: What Feed-Forward 3D Models Know About Matching ECCV 2026 SfM-DL Workshop

D. Nordström, X. Zhang, T. Loiseau, V. Lepetit, F. Kahl

github.com/davnords/RoMa-Omega

But the information is there, just not in a matching-friendly space



Zero-shot collapses at L=24. A **linear probe on the very same features beats DINOv3**.

RoMa-Ω: What Feed-Forward 3D Models Know About Matching ECCV 2026 SfM-DL Workshop

D. Nordström, X. Zhang, T. Loiseau, V. Lepetit, F. Kahl

github.com/davnords/RoMa-Omega

Matching from VGGT-Ω's raw predictions gives SotA

Relative-pose AUC on ScanNet-1500, with no matching-specific training. Correspondences are extracted by warping through the predicted depth.

Method	AUC@5°	AUC@10°	AUC@20°
VGGT-Ω raw predictions			
Depth warp	**34.2**	**57.6**	**74.8**
Mutual NN in 3D	22.7	43.0	59.5
Direct (from predicted cameras)	34.2	57.6	74.9
Trained matchers			
VGGT	33.9	55.2	73.4
RoMa v2	33.6	56.2	73.8

Competitive out of the box. However, **only 29.0 mAA on WxBS and 20.2 on HardMatch.** Collapses for difficult benchmarks.

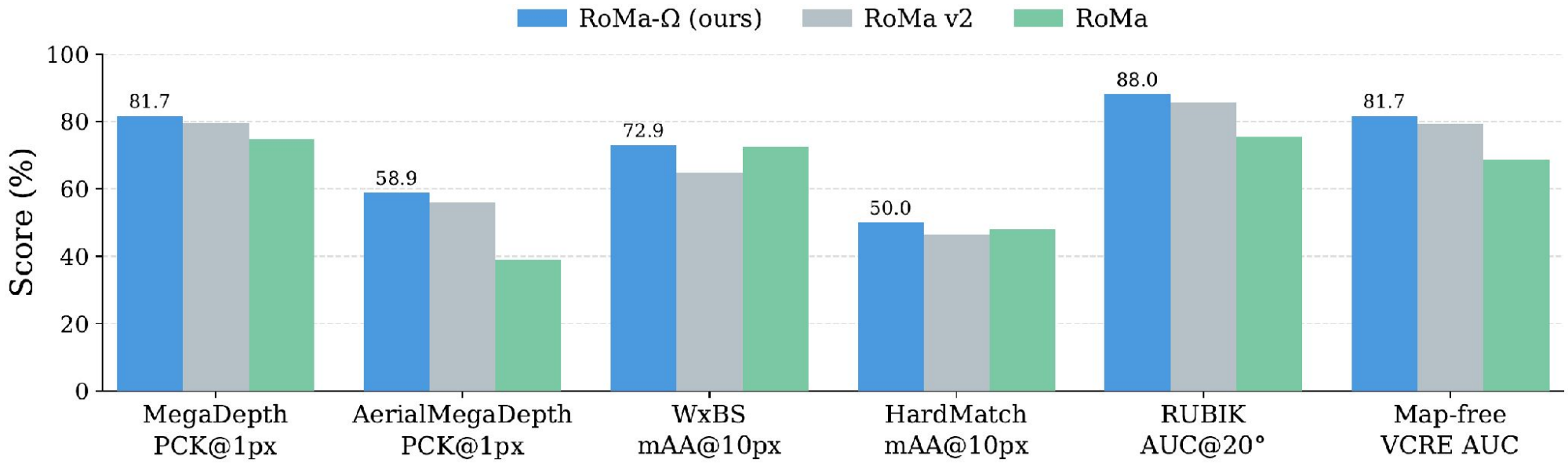
So swap the backbone: DINOv3 → VGGT-Ω

+8.1 **WxBS** mAA@10px

+3.5 **HardMatch** mAA@10px

+2.4 **RUBIK** AUC@20°

vs. RoMa v2



Remember RoMa v2 losing robustness? **RoMa-Ω gets it back**, giving RoMa's generalization with v2's sub-pixel accuracy.

RoMa-Ω: What Feed-Forward 3D Models Know About Matching ECCV 2026 SfM-DL Workshop

D. Nordström, X. Zhang, T. Loiseau, V. Lepetit, F. Kahl

github.com/davnords/RoMa-Omega

But where does all that supervision come from?



I talked to a strong lab that uses LoMa to generate the training data for their feed-forward reconstruction models.

We build a matcher, to build a pipeline, to make labels,
to train a model that **replaces the pipeline**.

The supervision ceiling is set by the pipeline we were trying to escape. **That can't be the end state.**

PART 3

Self-supervision: 3D as a by-product

No labels, no pseudo-labels, no pipeline. Learn from raw image sequences and let the 3D understanding emerge.

- Why SSL is the scalable bet
- MuM: masked modeling, many views
- Orochi: multi-view self-distillation

SSL works, but it has been optimized for semantics

DINO FAMILY

Self-distillation on single images. The strongest general-purpose features we have, tuned for semantics rather than geometry.

MAE

Reconstruct masked pixels. Stable, simple and scalable, but spends capacity on high-frequency detail.

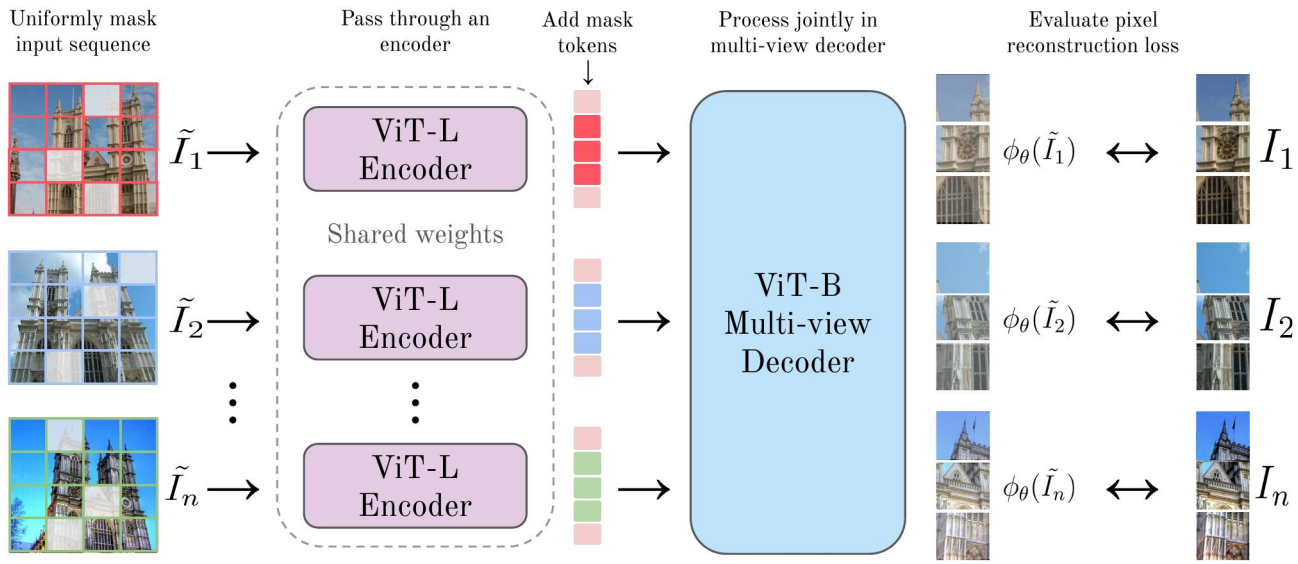
CROCO

Cross-view completion: MAE across two views. The first SSL objective aimed squarely at 3D.

A single image is the wrong unit for 3D. **Use the sequence.**

CroCo showed two views help. **Why stop at two?**

MuM: masked image modeling across many views



OBJECTIVE

Uniformly mask every view, reconstruct the pixels

ARCHITECTURE

Shared ViT-L encoder + light multi-view decoder

VS CROCO

Symmetric and n-view, so simpler and more scalable

MuM: Multi-View Masked Image Modeling CVPR 2026

D. Nordström, J. Edstedt, F. Kahl, G. Bökman

github.com/davnords/mum

No labels, less compute, and better 3D than DINOv3

AUC@30° ↑	CO3Dv2	RE10K	MegaDepth
CroCo v2	58.2	27.7	60.7
DINOv3	66.9	36.7	59.3
MuM	**71.5**	**50.8**	**73.0**

MULTI-VIEW CAMERA POSE, FROZEN ENCODER

Wins every multi-view 3D task. Pose estimation, point clouds, dense matching and relative pose. MuM beats both DINOv3 and CroCo v2.

Loses on semantics. Classification and segmentation still favor DINOv3. The objective buys geometry, and it costs semantics.

Trained on a fraction of DINOv3's compute and data. **Geometry is a data-efficiency story, not a scale story.**

MuM: Multi-View Masked Image Modeling CVPR 2026

D. Nordström, J. Edstedt, F. Kahl, G. Bökman

github.com/davnords/mum

But pixel reconstruction is the wrong objective

WHAT MAE COSTS YOU

Capacity goes into predicting high-frequency pixel detail nobody wants, and the raw features are poor for nearest-neighbor use.

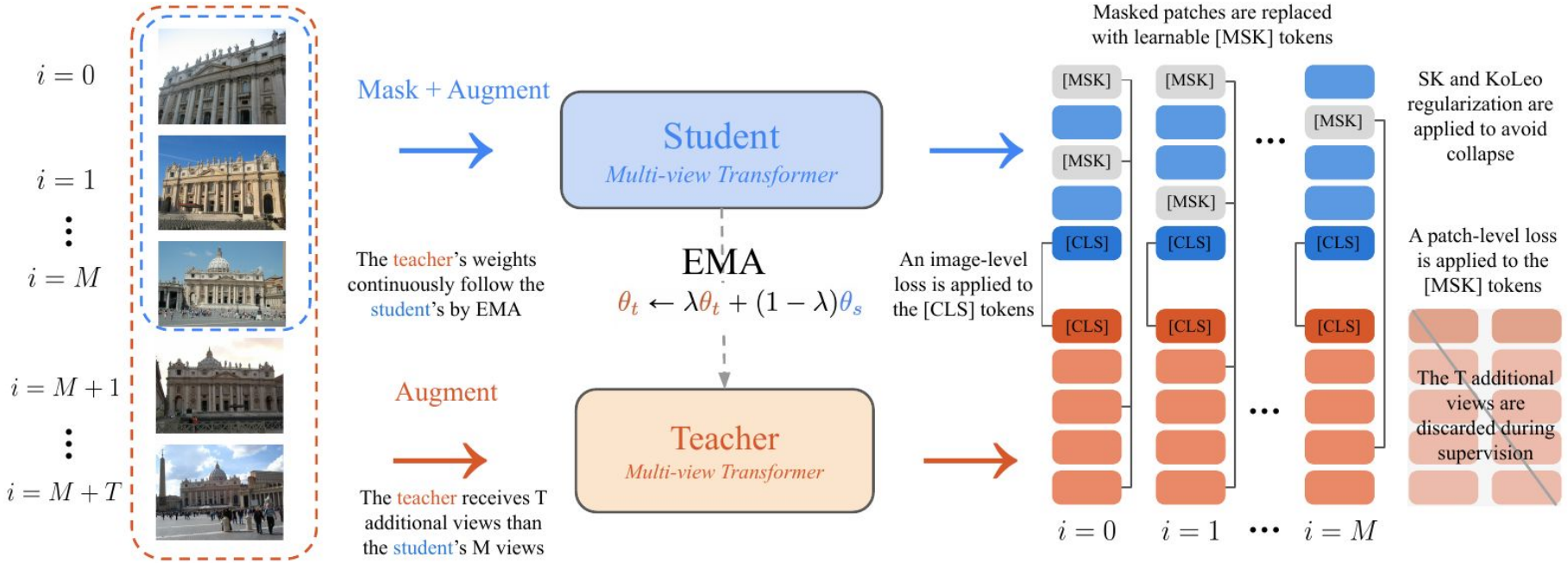
WHAT DINO GETS RIGHT

Self-distillation never touches pixels. It aligns representations, which is why DINOv3's frozen features are so much stronger.

Can self-distillation be trained **from scratch** across many views?

Short answer: yes, but **not by naively extending DINO**. That collapses.

Orochi: multi-view self-distillation



NO LOCAL CROPS

The single biggest fix, since local crops break multi-view alignment

IMAGE-LEVEL LOSS

A [CLS] objective alongside the masked-patch objective

TEACHER SEES MORE

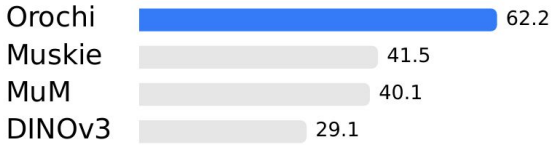
Teacher gets T extra views, student sees only M

Orochi: Learning 3D Without Supervision unpublished · ongoing work

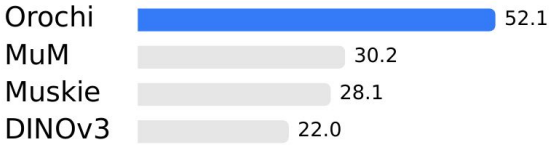
D. Nordström, T. Loiseau, G. Bourmaud, M. Felsberg, V. Lepetit, F. Kahl

State of the art across every 3D task we tried

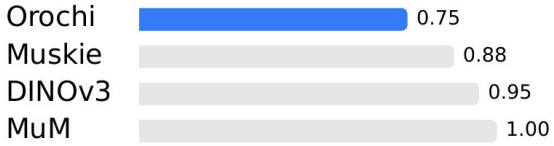
Pose Estimation RE10K (AUC@30°)



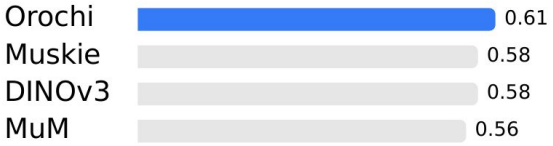
Pose Estimation SN++ (AUC@30°)



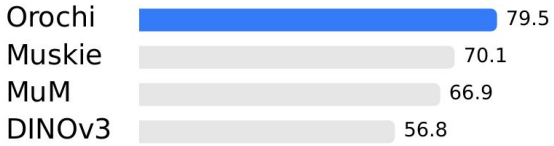
Point Cloud Error ETH3D (mm)



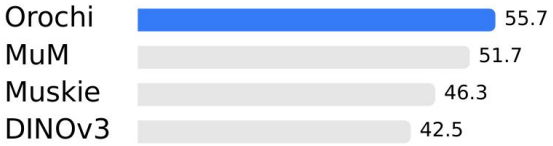
Normal Consistency DTU



MNN Matching SN (PCK@25)



SE(3) Decodability ($R^2 > 0$)



Beats every SSL baseline. DINOv3, MuM and Muskie, on pose, point clouds, matching and SE(3) structure.

And beats supervised 3D models. For correspondence, Orochi's features beat VGGT-Ω and π^3, which were trained with explicit 3D labels.

On a fraction of the compute. Matches DINOv3's zero-shot correspondence after under a day on one 8×H200 node. DINOv3 used ~320× more.

Orochi: Learning 3D Without Supervision unpublished · ongoing work

D. Nordström, T. Loiseau, G. Bourmaud, M. Felsberg, V. Lepetit, F. Kahl

Correspondence emerges without supervision

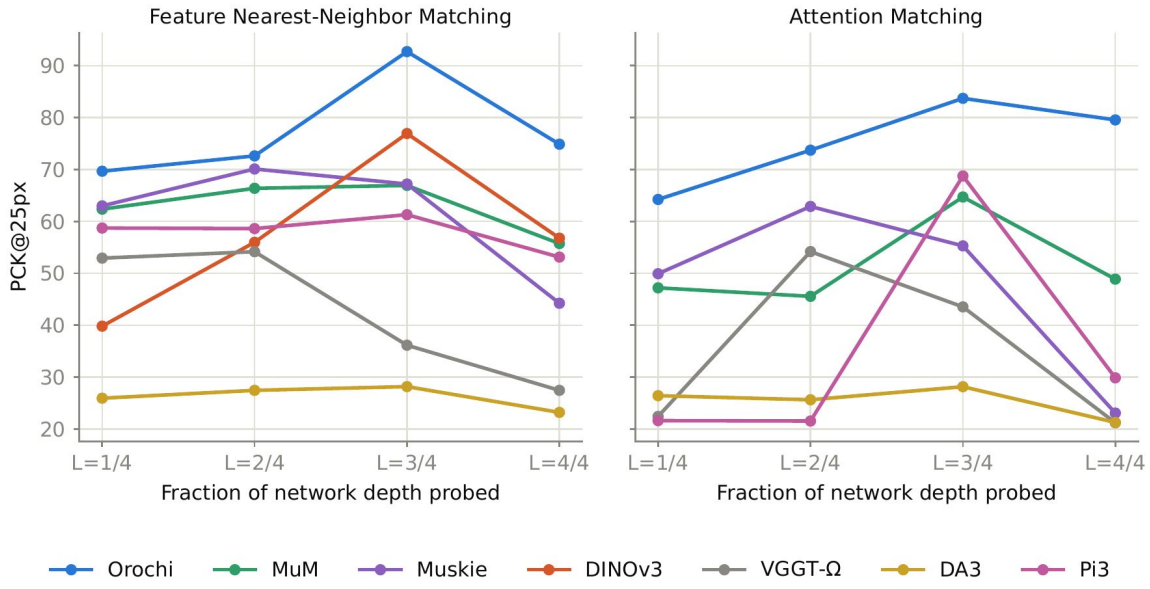


Tracks are the **arg-max of the attention map**. No labels, no attention supervision. It just falls out.

Orochi: Learning 3D Without Supervision unpublished · ongoing work

D. Nordström, T. Loiseau, G. Bourmaud, M. Felsberg, V. Lepetit, F. Kahl

And unlike the supervised models, it doesn't degrade with depth



Remember part two: VGGT-Ω's correlation maps collapsed in the later layers, and you needed a probe to get the information back out.

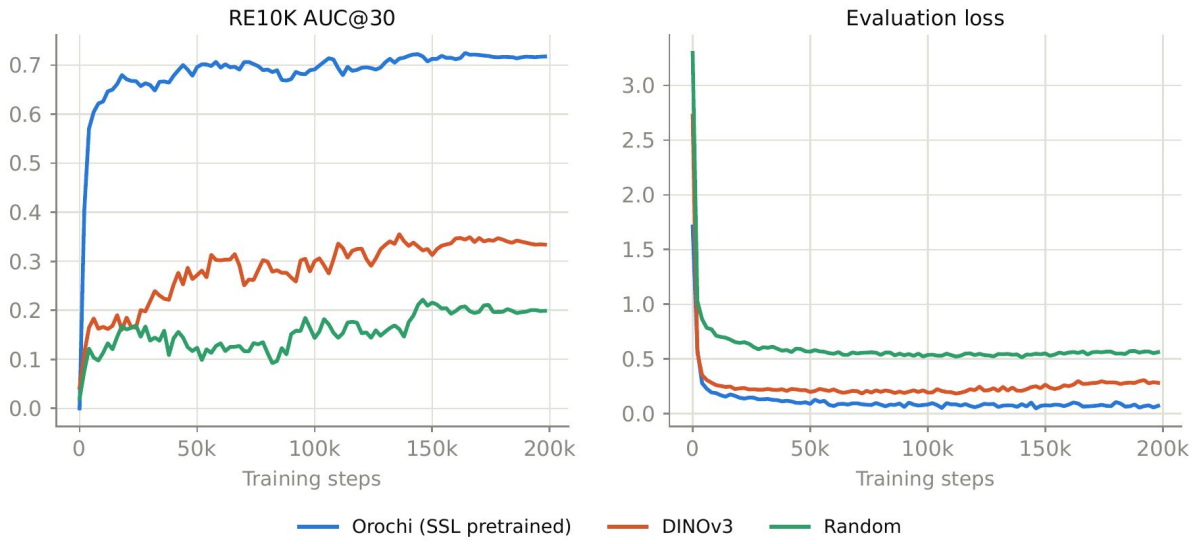
Orochi encodes multi-view geometry throughout the network, so the features are usable directly, at any depth.

Supervised 3D models bury the geometry. **Self-supervision keeps it on the surface.**

Orochi: Learning 3D Without Supervision **unpublished · ongoing work**

D. Nordström, T. Loiseau, G. Bourmaud, M. Felsberg, V. Lepetit, F. Kahl

And it makes supervised models easier to train



Training feed-forward reconstruction from scratch is hard. Initializing from DINOv3 barely helps.

Initializing from Orochi gets you to AUC@30° > 65 in 10K steps with a low learning rate.

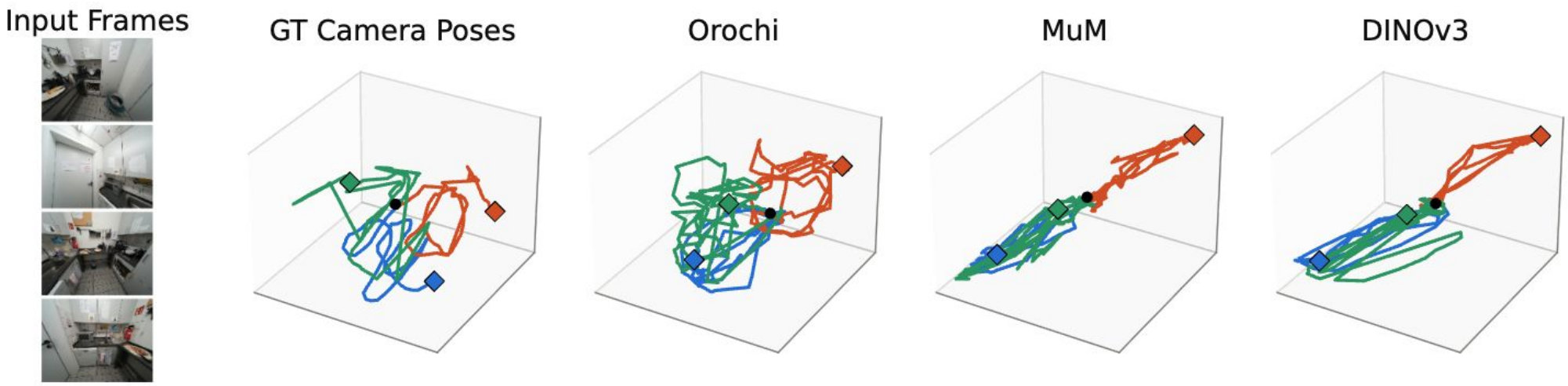
So the two paradigms aren't rivals: SSL is the pretraining, supervised 3D is the fine-tune.

The data loop breaks: **learn the representation from raw images**, then spend your scarce labels on the head.

Orochi: Learning 3D Without Supervision **unpublished · ongoing work**

D. Nordström, T. Loiseau, G. Bourmaud, M. Felsberg, V. Lepetit, F. Kahl

Understanding of camera trajectories also emerges



Visualize the alignment of visual features and **camera motion trajectories** using a lightweight network. Using features from Orochi , the adapter is able to unroll the feature space to closely follow the ground truth camera trajectories.

Orochi: Learning 3D Without Supervision **unpublished · ongoing work**

D. Nordström, T. Loiseau, G. Bourmaud, M. Felsberg, V. Lepetit, F. Kahl

WRAPPING UP

Learning to understand 3D

01 **Pipelines still work, and learned front-ends make them better.** LoMa put a scaled local matcher back ahead of the dense models, and it drops straight into COLMAP. RoMa v2 removes the detector entirely.

02 **Supervised 3D models really do learn geometry, but they bury it.** RoMa-Ω: VGGT-Ω's features look collapsed under cosine similarity, yet beat DINOv3 under a probe. Swapping the backbone gives state of the art.

03 **The labels come from the pipeline. Self-supervision is the way out.** MuM beats DINOv3 on 3D with far less compute. Orochi beats supervised 3D models at correspondence with no labels at all, and makes them faster to train.

NLP got one scalable objective. For 3D, I think **multi-view self-distillation** is the best candidate we have. Video models and next-view-prediction (**Atlas**) also promising.

Thank you

Questions very welcome.

CODE AND WEIGHTS ON GITHUB

LoMa
ECCV 2026 Short Oral
davnords/loma

RoMa v2
ECCV 2026 Short Oral
Parskatt/RoMaV2

RoMa-Ω
SfM-DL @ ECCV 2026
davnords/RoMa-Omega

MuM
CVPR 2026
davnords/mum

Orochi
unpublished
paper not out yet

David Nordström · Chalmers University of Technology

Work supported by WASP. Compute from NAISS (Berzelius, Alvis, C3SE).

APPENDIX

LoMa: relative pose, WxBS and HardMatch

AUC on MegaDepth-1500 and ScanNet-1500, and mAA@10px on WxBS and HardMatch (HM). 4096 keypoints.

Method	MD 5°	10°	20°	SN 5°	10°	20°	WxBS	HM
Feed-forward reconstruction								
MASt3R	42.4	61.5	76.9	33.6	56.8	74.1	34.5	33.6
VGGT	33.5	52.9	70.0	33.9	55.2	73.4	36.3	28.4
Dense matchers								
LoFTR	52.8	69.2	81.2	22.1	40.8	57.6	50.7	33.1
RoMa	62.6	76.7	86.3	31.8	53.4	70.9	72.6	48.1
UFM	41.5	57.9	72.4	31.3	54.1	72.0	53.3	33.9
RoMa v2	62.8	77.0	86.6	33.6	56.2	73.8	64.8	46.5
Sparse matchers, 4096 keypoints								
SP+SG	43.7	61.8	76.5	16.4	32.5	49.0	45.6	36.0
SP+LG	43.8	61.8	76.4	15.9	32.1	48.9	40.4	34.8
ALIKED+LG	48.1	65.7	79.3	14.5	28.9	43.5	43.9	35.7
LoMa-B (ours)	55.7	71.8	83.6	27.5	49.7	68.2	68.7	51.1
LoMa-L (ours)	56.5	72.7	84.3	29.3	51.9	70.3	70.6	53.5
LoMa-G (ours)	**56.1**	**72.2**	**84.0**	**29.3**	**51.7**	**70.0**	**73.4**	**54.3**

APPENDIX

LoMa: visual localization

Map-free (VCRE<90px), InLoc and Oxford Day-and-Night. The latter two report % localized within (0.25m, 2°) / (0.5m, 5°) / (1m, 10°).

Method	Map-free Prec.	AUC	InLoc DUC1	DUC2	Oxford Night
SP+SG	46.3	74.1	46.5/65.7/78.3	52.7/72.5/79.4	44.3/54.4/58.0
SP+LG	45.5	76.2	43.9/64.6/76.8	42.7/68.7/74.0	43.4/53.5/57.7
ALIKED+LG	47.2	79.5	41.4/64.6/79.8	35.9/64.1/67.9	42.8/53.9/58.9
LoMa-B (ours)	65.6	89.0	57.1/80.8/91.9	71.0/87.0/88.5	54.7/62.4/66.2
LoMa-L (ours)	67.6	89.4	59.1/80.8/91.9	71.0/84.0/87.8	56.0/64.6/69.2
LoMa-G (ours)	**68.9**	**90.3**	**55.6/80.3/91.4**	**73.3/87.8/89.3**	**58.9/66.0/69.7**

Also: RUBIK 73.2 / 79.9 AUC@10°/20° and IMC 2022 89.3 mAA, above the 2022 competition winner (86.3) and RoMa (88.0).

APPENDIX

LoMa: ablation and per-group HardMatch

ABLATION · HARDMATCH VAL mAA@10PX

Step		mAA@10px
I	ALIKED + LightGlue	36.3
II	DaD + DeDoDe + LG	42.0
III	Matcher → all data	48.9
IV	Descriptor → all data	51.1
V	Longer training (LoMa-B)	51.5
VI	LoMa-L	52.8
VII	**LoMa-G**	**52.9**

HARDMATCH mAA@10PX, BY GROUP

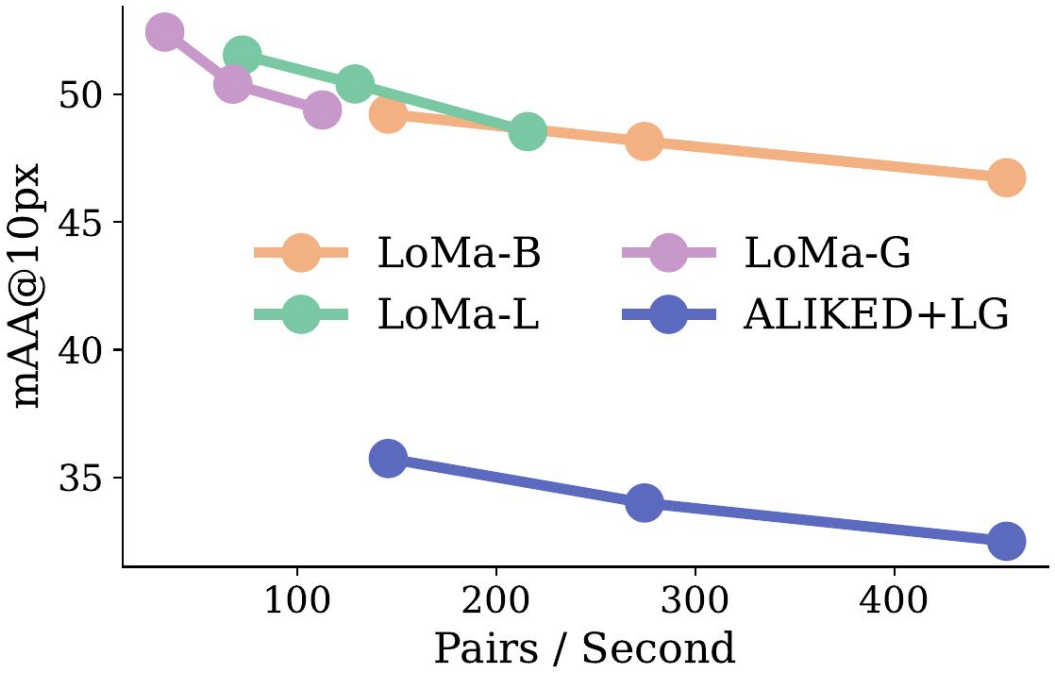
Method	Aerial	Celest.	Doppel.	Viewpt.
MASt3R	13.0	20.2	25.4	18.2
VGGT	12.9	8.7	14.8	15.6
RoMa	27.2	26.1	28.7	20.8
RoMa v2	28.7	28.3	34.1	28.6
ALIKED+LG	14.0	16.4	21.5	10.5
LoMa-G (ours)	**36.9**	**35.7**	**36.3**	**30.9**

Scaling the training data (steps III to IV) is by far the largest single contribution, worth about +9 mAA. Doppelgängers, celestial pairs and extreme viewpoint change remain the hardest groups for every method.

APPENDIX

LoMa: speed and accuracy trade-off

HardMatch mAA against throughput on an A100, sweeping the early-exit layer L ∈ {3, 5, 9}.



Layer-wise supervision means you can stop the matcher early and trade accuracy for speed.

LoMa-B at LightGlue's runtime still produces far better matches, close to 300 pairs/s for a single pair at L=3.

APPENDIX

RoMa v2: dense matching and extreme benchmarks

Dense matching at 640×640. WxBS, SatAst and RUBIK for robustness.

	TA-WB				MegaDepth				ScanNet++ v2			
Method	**EPE ↓**	**1px**	**3px**	**5px**	**EPE ↓**	**1px**	**3px**	**5px**	**EPE ↓**	**1px**	**3px**	**5px**
RoMa	60.61	35.1	52.6	56.2	2.34	74.8	93.7	96.4	27.52	20.2	42.8	53.6
UFM	15.85	31.3	65.5	75.1	3.15	55.3	88.0	93.7	6.93	31.4	67.7	80.0
RoMa v2	**13.82**	**67.7**	**81.8**	**85.8**	**1.47**	**79.6**	**94.7**	**96.7**	**4.00**	**45.5**	**77.3**	**86.6**

Method	**WxBS mAA@10px**	**SatAst AUC@10px**	**RUBIK success %**
RoMa	60.8	23.5	47.3
UFM	42.3	1.8	53.1
RoMa v2	**55.4**	**37.0**	**57.3**

The one regression: RoMa is still more robust on WxBS. That is exactly the gap RoMa-Ω closes.

APPENDIX

RoMa-Ω: matching from VGGT-Ω's raw predictions

Relative-pose AUC on ScanNet-1500, with no matching-specific training.

Method	AUC@5°	AUC@10°	AUC@20°
VGGT-Ω raw predictions			
Depth warp	**34.2**	**57.6**	**74.8**
Mutual NN in 3D	22.7	43.0	59.5
Direct (from predicted cameras)	34.2	57.6	74.9
Trained matchers			
VGGT	33.9	55.2	73.4
RoMa v2	33.6	56.2	73.8

Competitive out of the box, but only 29.0 mAA on WxBS and 20.2 on HardMatch, so it collapses under illumination and modality shift. That is what motivates training a matcher on top.

APPENDIX

RoMa-Ω: benchmarks, localization and cost

EXTREME MATCHING

Method	WxBS mAA@10px	HardMatch mAA@10px	RUBIK AUC@20°
VGGT-Ω	29.0	20.2	84.3
RoMa	72.6	48.1	75.4
RoMa v2	64.8	46.5	85.6
RoMa-Ω (ours)	**72.9**	**50.0**	**88.0**

MAP-FREE LOCALIZATION · AND WHAT IT COSTS

Method	VCRE<90px AUC	VCRE<45px AUC	Memory (GB)	Params (total)
RoMa	84.4	68.5	3.9	414M
RoMa v2	93.3	79.4	4.0	425M
RoMa-Ω (ours)	**94.5**	**81.7**	**29.9**	**1033M**

The real limitation is cost: ~40% slower than RoMa v2 and 30 GB of VRAM against 4.

APPENDIX

RoMa-Ω: architecture ablation

Coarse matcher after 100K steps, PCK.

Variant	Hypersim 1px	3px	5px	ETH3D 1px	3px	5px
I RoMa v2 (baseline)	21.0	61.8	74.2	40.6	85.8	94.2
II VGGT-Ω backbone + DPT head	4.9	28.3	48.0	8.7	47.5	70.7
III + transformer decoder	21.5	64.6	78.8	34.1	82.3	93.0
IV + only 2 deepest levels (RoMa-Ω)	**27.8**	**68.5**	**80.9**	**43.4**	**87.7**	**95.1**
V + full width in DPT (2048)	23.2	65.1	79.1	37.6	85.3	94.4

The transformer decoder is essential. A DPT head alone on VGGT-Ω is far worse than the baseline.

APPENDIX

RoMa-Ω: qualitative dense warps

HardMatch pairs. Top: the input pair. Bottom: image B warped into image A by RoMa-Ω.



APPENDIX

MuM: full results

Frozen ViT-L encoders. Pose by AUC@30° ↑. Point clouds by median accuracy and completeness ↓.

MULTI-VIEW CAMERA POSE · AUC@30° ↑

Method	CO3Dv2	RE10K	MegaDepth
Frozen encoder			
CroCo v2	58.2	27.7	60.7
DINOv3	66.9	36.7	59.3
MuM	**71.5**	**50.8**	**73.0**
Distillation finetuning			
Random init.	10.6	28.0	41.8
MuM	**62.6**	**50.9**	**78.6**

POINT CLOUD ESTIMATION ↓

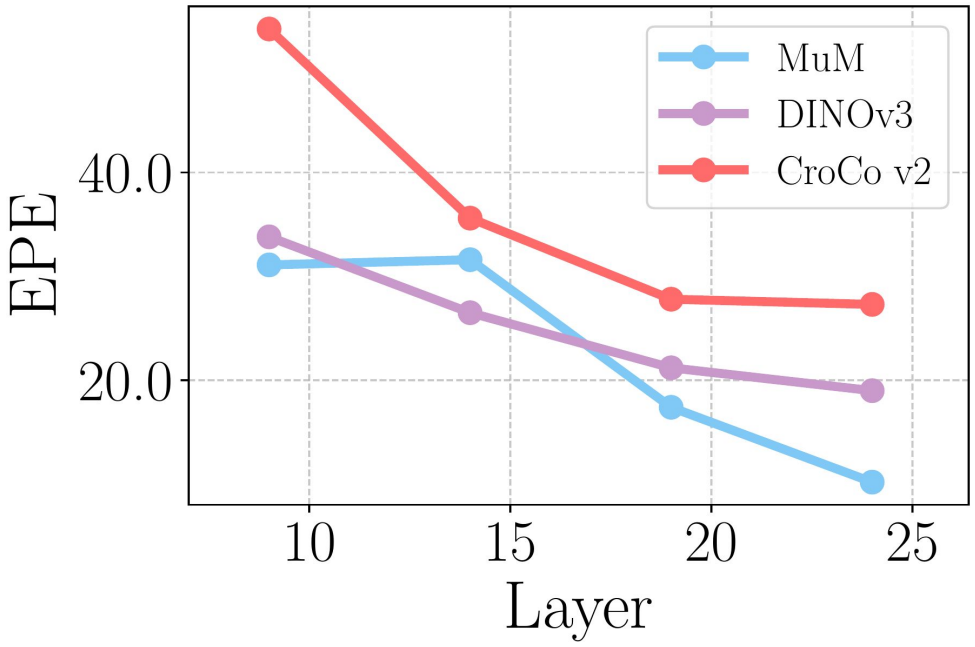
Method	DTU Acc.	Comp.	ETH3D Acc.	Comp.
Frozen encoder				
CroCo v2	8.5	12.3	0.9	1.0
DINOv3	6.4	3.7	0.9	1.0
MuM	**3.7**	**1.6**	**0.8**	**0.8**
Distillation finetuning				
Random init.	8.4	11.7	1.0	1.4
MuM	**6.4**	**2.6**	**0.6**	**0.5**

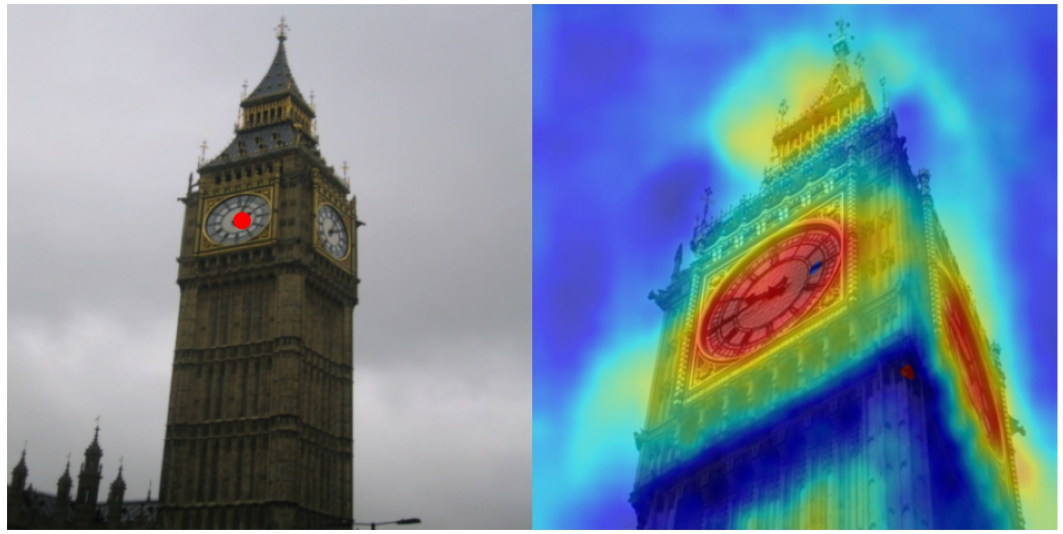
MuM wins every multi-view task against both DINOv3 and CroCo v2, on substantially less compute and data. It loses to DINOv3 on single-image semantic tasks such as classification and segmentation.

APPENDIX

MuM: feature quality across layers

Left: layer-wise feature behavior. Right: attention-based matching.





APPENDIX

Orochi: feed-forward reconstruction

Relative pose by AUC over 10 random frames. Point clouds by median accuracy (mm) and normal consistency. 4×H200 for 3 days per run.

	RE10K		ScanNet++		MegaDepth		ETH3D		DTU	
Method	**AUC@3°**	**@30°**	**AUC@3°**	**@30°**	**AUC@3°**	**@30°**	**Acc. ↓**	**NC ↑**	**Acc. ↓**	**NC ↑**
Train only the heads, frozen backbone										
DINOv3	0.0	18.9	0.0	9.4	1.5	55.5	1.18	0.59	10.76	0.54
Muskie	1.1	36.3	0.0	19.0	0.1	46.9	1.12	0.60	11.43	0.55
MuM	0.0	29.8	0.0	18.7	0.1	48.5	1.12	0.60	12.60	0.54
Orochi	**2.4**	**51.9**	**0.1**	**48.7**	**3.7**	**67.5**	**0.82**	**0.65**	**10.60**	**0.59**
Train a transformer on the frozen backbone										
DINOv3	1.0	29.1	0.0	22.0	1.7	65.6	0.95	0.66	10.63	0.58
Muskie	1.8	41.5	0.0	28.1	1.7	58.9	0.88	0.67	11.95	0.58
MuM	1.4	40.1	0.0	30.2	1.9	60.8	1.00	0.60	12.26	0.56
Orochi	**5.2**	**62.2**	**1.3**	**52.1**	**5.2**	**69.5**	**0.75**	**0.68**	**10.52**	**0.61**
Finetune the backbone										
Random init.	0.0	17.4	0.0	11.3	0.0	35.9	1.24	0.60	12.34	0.56
DINOv3 init.	1.3	31.2	0.0	30.9	7.4	73.8	0.98	0.68	10.87	0.59
Orochi	**8.3**	**68.2**	**4.5**	**67.0**	**6.9**	**74.6**	**0.81**	**0.80**	**6.73**	**0.63**

APPENDIX

Orochi: multi-view correspondence

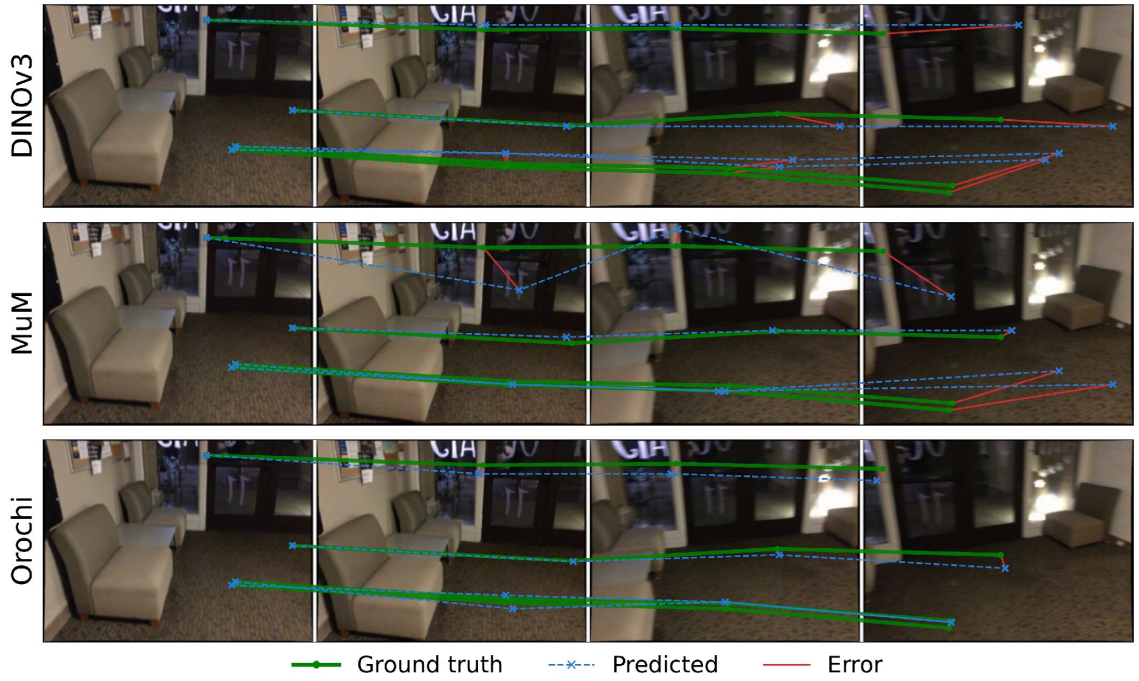
Zero-shot patch tracking across 8 views, PCK. Baselines are reported at their best layer, while Orochi is strong at every layer.

Method	ScanNet 10px	25px	50px	NAVI 10px	25px	50px
Feature nearest-neighbor matching						
VGGT-Ω	32.8	52.9	69.0	25.9	51.6	70.9
Pi3	38.6	61.3	75.6	30.3	60.2	78.8
DINOv3	33.9	56.8	72.2	28.7	59.6	78.7
Muskie	42.4	70.1	83.1	32.7	64.7	80.5
MuM	40.5	66.9	79.9	30.2	60.2	74.9
Orochi	**46.2**	**79.5**	**89.6**	**33.7**	**68.7**	**84.2**
Attention matching						
VGGT-Ω	32.9	54.2	68.9	23.9	46.8	63.7
Pi3	40.7	68.7	83.8	30.3	62.2	82.0
Muskie	34.7	62.9	81.5	22.3	46.2	67.9
MuM	37.2	64.7	77.2	28.5	56.6	68.7
Orochi	**43.9**	**83.7**	**94.9**	**34.7**	**74.5**	**86.5**

APPENDIX

Orochi: qualitative correspondence tracks

Predicted tracks in blue, ground truth in green, error in red. Orochi produces geometrically consistent tracks, while the baselines do not.



APPENDIX

Orochi: feature PCA

Principal components of the dense features, colored as RGB.

