GGPT: Geometry Grounded Point Transformer – Supplementary Material –

A. Results with RoMa v2 and Comparison with Additional Baselines

In the main paper, we ensemble the dense correspondences predicted by RoMa [6] and UFM [40] (Sec. E.1) within the SfM pipeline. This configuration is used to produce the results reported in the main paper and the ablation studies in Sec. B. Following the initial submission, we observed that replacing RoMa and UFM with the more recent dense matcher, *i.e*. RoMa v2 [7], further improves performance across all evaluation datasets. The updated results are reported in Tab. 2 and Tab. 1. In addition, we include two recent feed-forward models for comparison: *i.e*. Pi3X [36][1] and DepthAnything3 [19]. As shown in Tab. 2 and Tab. 1, our method GGPT consistently outperforms these feed-forward baselines, regardless of whether it employs RoMa + UFM or RoMa v2, with particularly notable gains on cross-domain and out-of-domain datasets.

Table 1. **Multi-view 3D reconstruction on out-of-domain datasets.** AUC@1/5 cm (% ↑) for human body reconstruction (4D-DRESS [35]) and AUC@1/5 mm (% ↑) for surgical scene reconstruction (MV-dVRK [1]). **Here we report additional results with latest dense matcher RoMa v2 [7], which further boosts the performance.**

	4D-DRESS	MV-dVRK
VGGT [32]	10/45	8/33
[32] + Ours (RoMa + UFM)	**66/77**	**45/61**
[32] + Ours (RoMa v2)	**75/87**	**62/77**
Pi3 [36]	8/50	18/51
[36] + Ours (RoMa + UFM)	**63/80**	**40/67**
[36] + Ours (RoMa v2)	**76/88**	**67/82**
Pi3X [36]	3/12	4/22
[36] + Ours (RoMa + UFM)	**49/66**	**42/57**
[36] + Ours (RoMa v2)	**69/83**	**60/76**
MapAnything [14]	2/12	3/13
[14] + Ours (RoMa + UFM)	**42/52**	**35/47**
[14] + Ours (RoMa v2)	**54/75**	**33/52**
DepthAnything3 [19]	2/5	19/52
[19] + Ours (RoMa + UFM)	**29/47**	**47/68**
[19] + Ours (RoMa v2)	**49/70**	**67/80**
MAtCha [8]	48/68	37/62
[8] + Ours (RoMa + UFM)	**62/75**	**50/67**
[8] + Ours (RoMa v2)	**67/80**	**64/80**
MASt3R-SfM [5]	54/71	39/62
[5] + Ours (RoMa + UFM)	**64/75**	**49/66**
[5] + Ours (RoMa v2)	**70/82**	**63/79**

B. Ablation Studies

B.1. Structure-from-Motion

We conduct experiments to validate the contribution of our three SfM design components. (1) *Dense matchers*: Using dense matchers rather than sparse matchers. (2) *Sparse BA*: Selecting a small subset of matches for Bundle Adjustment (BA) instead of a large set. (3) *DLT triangulation*: Using Direct Linear Triangulation (DLT) rather than conventional RANSAC-based non-linear triangulation.

The ablation is conducted on the ETH3D test set [25] with varying baseline coverages with 4, 8, and 16 input views. In Tab. 3, we report the running time of each step (*Time*), and the accuracy of the camera poses (*Cam.*) and sparse points estimation (*Pts.*). We adhere to standard Structure-from-Motion (SfM) evaluation protocols [20, 22, 31]. For camera poses, we compare the relative poses to the ground truth, compute the maximum of rotational and translational angular errors, and report the Area Under the recall Curve (AUC) up to 1°, 5°, and 10°. For sparse points, we compute the chamfer distance between the triangulated points and the ground truth, report the accuracy and completeness of the reconstruction as the ratio of points within a given distance of each other.

Dense matchers. We compare dense matching regressors [6, 40] with the SOTA sparse keypoint-based MASt3R [18], which uses a reciprocal nearest-neighbor (NN) searching algorithm to extract a sparse grid of correspondences between each pair. Tab. 3 shows that the sparse MASt3R matcher is noticeably slower and produces less reliable correspondences than dense matching regressors, which leads to suboptimal camera and points accuracy. due to the sparsity of the matchings, it can only produce a extremely sparse set of points, yielding a low completeness, *e.g*. 4 % vs 20 % on the 8-view setup. For SOTA dense matchers, we find that RoMa [6] offers better accuracy while UFM [40] offers better completeness and efficiency. Ensembling the two dense matchers can achieve the balance between points accuracy and completeness with minimal computational overhead, as shown in the *RoMa + UFM (Ours, Dense)* row of Tab. 3. In Sec. B.3, we will pro-

[1] Pi3X is an enhanced version of Pi3 with engineering updates. The model weights are publicly available at https://github.com/yyfz/Pi3.

Table 2. **Multi-view 3D Reconstruction on Standard Test Sets.** We report AUC@5/10 cm (% ↑). Our method (our SfM and GGPT) can improve dense predictions of various models. Despite GGPT is trained solely on ScanNet++ and VGGT's dense prediction, as we highlight the top-left cells as **within-domain**, it generalises well to **cross-domain datasets** and other methods' predictions (rows 3–10). For the results reported in the main paper, we ensemble the outputs of RoMa [6] and UFM [40] for dense matchings. **Here we report additional results with latest dense matcher RoMa v2 [7], which further boosts the performance.**

	ScanNet++ [39]			**ETH3D** [25]			**T&T** [16]		
	4 imgs	8 imgs	16 imgs	4 imgs	8 imgs	16 imgs	4 imgs	8 imgs	16 imgs
VGGT [32]	23/37	19/32	16/29	27/41	23/36	19/32	26/40	25/39	24/38
[32] + Ours (RoMa + UFM)	**38/53**	**45/60**	**50/66**	**41/55**	**47/61**	**49/63**	**34/47**	**42/57**	**43/57**
[32] + Ours (RoMa v2)	**47/62**	**56/70**	**62/76**	**54/66**	**57/69**	**62/75**	**39/52**	**50/63**	**49/62**
Pi3 [36]	**54/69**	**56**/71	58/**74**	31/47	25/41	23/38	25/39	26/42	25/40
[36] + Ours (RoMa + UFM)	**54**/68	**56/72**	**59/74**	**36/53**	**36/53**	**37/54**	**27/43**	**32/50**	**33/50**
[36] + Ours (RoMa v2)	**58/72**	**61/75**	**65/78**	**44/59**	**45/60**	**46/63**	**27/42**	**35/53**	**36/53**
Pi3X [36]	**51**/66	**54**/70	**55/72**	34/49	29/45	26/41	25/40	24/40	23/39
[36] + Ours (RoMa + UFM)	**51/67**	**54/71**	**55/72**	**38/54**	**42/57**	**49/64**	**31/44**	**34/51**	**41/55**
[36] + Ours (RoMa v2)	**56/70**	**63/76**	**68/80**	**60/71**	**63/74**	**69/79**	**39/51**	**50/63**	**50/63**
MapAnything [14]	40/57	38/57	38/58	10/20	7/15	5/12	10/21	9/20	8/17
[14] + Ours (RoMa + UFM)	**44/61**	**48/64**	**52/68**	**32/43**	**33/45**	**34/47**	**29/43**	**40/55**	**42/56**
[14] + Ours (RoMa v2)	**56/70**	**62/75**	**67/79**	**45/57**	**48/61**	**51/64**	**35/49**	**46/61**	**48/61**
DepthAnything3 [19]	10/20	7/16	5/13	28/42	25/39	23/37	10/21	8/19	8/18
[19] + Ours (RoMa + UFM)	**36/50**	**45/60**	**53/68**	**35/49**	**41/57**	**45/60**	**30/43**	**31/48**	**38/53**
[19] + Ours (RoMa v2)	**47/59**	**57/70**	**65/77**	**53/64**	**58/70**	**61/72**	**38/51**	**46/61**	**48/62**
MAtCha [8]	12/19	15/26	18/32	40/52	41/53	42/56	34/47	36/50	33/47
[8] + Ours (RoMa + UFM)	**27/37**	**40/52**	**48/63**	**42/55**	**47/60**	**50/65**	**35/48**	**43/57**	**43/57**
[8] + Ours (RoMa v2)	**38/48**	**47/60**	**60/72**	**53/65**	**56/67**	**62/74**	**39/52**	**50/63**	**49/61**
MASt3R-SfM [5]	12/22	14/35	16/31	37/50	39/51	40/54	**34**/46	37/50	36/49
[5] + Ours (RoMa + UFM)	**30/41**	**39/52**	**48/64**	**41/55**	**46/59**	**48/63**	33/**47**	**41/56**	**42/56**
[5] + Ours (RoMa v2)	**40/50**	**51/64**	**60/73**	**52/64**	**55/66**	**62/73**	**39/52**	**49/62**	**48/61**

Table 3. **Ablation of SfM Components on ETH3D.** We evaluate design variants across three stages of our SfM—matching, bundle adjustment, and triangulation—and highlight the configuration with the best efficiency or accuracy. For each variant, we report: (1) **Time** denotes the running time (s ↓) of each stage. (2) **Cam.** denotes the AUC@1/5° (% ↑) for camera-pose estimation. (3) **Pts.** (% ↑) denotes the Accuracy/Completeness@1 cm (% ↑) of reconstructed points.

	4 Views			8 Views			16 Views		
	Time (s)	Cam. (%)	Pts. (%)	Time (s)	Cam. (%)	Pts. (%)	Time (s)	Cam. (%)	Pts. (%)
(a) **Dense Matcher**									
MASt3R [18] (Sparse)	8.4	75/86	19/ 2	21	75/85	21/ 4	57	76/86	26/ 8
RoMa [6] (Dense)	2.3	**90/93**	**47**/20	8.8	**87**/91	**46**/25	36	**89/93**	**46**/29
UFM [40] (Dense)	**1.0**	64/84	26/19	**1.7**	81/90	30/25	**8.5**	77/87	33/30
RoMa [6] + UFM [40] (Ours, Dense)	3.3	86/**93**	35/**22**	11	**87/92**	37/**30**	44	85/91	40/**35**
(b) **Sparse Bundle Adjustment (BA)**									
$n_{\text{BA}} = 512$	**0.2**	85/**93**	**36/23**	**0.5**	85/90	**38/31**	**1.0**	84/90	40/35
$n_{\text{BA}} = 2048$	1.0	86/**93**	35/22	1.9	87/**92**	37/30	15	**85/91**	40/35
$n_{\text{BA}} = 4096$	1.9	**87/93**	**36/23**	17	**88/92**	**38/31**	29	**85/91**	**41/36**
(c) **Direct Linear Triangulation (DLT)**									
Ransac-based non-linear triangulation [26]	14	**86/93**	34/**23**	64	**87/92**	36/**31**	272	**85/91**	39/**35**
Direct linear triangulation	**0.03**	**86/93**	**35**/22	**0.08**	**87/92**	**37**/30	**0.3**	**85/91**	**40/35**

vide further experiments showing that this ensemble strategy also yields the best final dense reconstruction.

Sparse BA. We study the effects of n_{BA}, *i.e.* the number of matches per image used for bundle adjustment (BA). Results show that reducing n_{BA} from 4096 to 512 can considerably reduce the running time, particularly for ≥ 8 input views, but achieves quite similar camera and points accuracy. Our default configuration uses $n_{\text{BA}} = 2048$, though this ablation shows that $n_{\text{BA}} = 512$ actually suffices for the ETH3D dataset.

DLT triangulation. For the final point triangulation stage, we emphasise the efficiency of direct linear triangulation (DLT) [9]. We compare it with the standard COLMAP pipeline [26], which conducts LO-RANSAC [3] for outlier rejection and a non-linear iterative optimization for refinement before and after linear triangulation respectively. For this method, we employ the implementation function [2] provided by pycolmap [24], using the same matches and camera poses as in our DLT implementation. The comparison shows that DLT is orders of magnitude faster than the RANSAC-based non-linear pipeline, as DLT can be executed directly as CUDA tensor operations (See Sec.E.1). Remarkably, even without LO-RANSAC and without a subsequent non-linear refinement, DLT attains comparable and in many cases superior accuracy and completeness. This robustness stems from our prefitering of correspondences using cycle consistency and matching confidence.

Table 4. **Ablation Studies on GGPT.** We report AUC@5/10 cm on within-domain **ScanNet++** and cross-domain **ETH3D** 8-view splits and AUC@1/5 cm on out-of-domain **4D-DRESS**.

	SNt++	ETH3D	4DDS.
VGGT [32]	19/32	23/36	10/45
(a) Refinement Architecture			
2D Transformers			
$\mathbf{X}_s \rightarrow$ VGGT [12, 32]	**55/70**	25/40	12/52
$\mathbf{X}_s \rightarrow$ MapAnything [14]	39/57	11/21	5/34
3D Networks			
Minkowski [2]	39/53	47/60	63/73
PTv3 [37] (Ours)	45/60	**47/61**	**66/77**
(b) Input Encodings			
w/o $\mathbf{X}_s$	21/35	20/34	13/42
w/o $\mathbf{X}_{d\rightarrow s}$	24/41	24/41	14/47
w/o $\mathbf{z}_s$	42/57	44/58	56/72
Full model	**45/60**	**47/61**	**66/77**
(c) Patch Size			
r=0.05	30/44	37/50	30/56
r=0.1	39/53	43/57	60/72
r=0.2	**45/60**	**47/61**	**66/77**
r=0.6	40/55	38/53	64/74
r=1.0	41/56	40/56	63/75

[2] pycolmap documentation

B.2. Geometry-Grounded Point Transformer

In this section, we perform ablations on our Geometry-Grounded Point Transformer (Tab. 4).

Refinement architecture. We first verify the advantage of performing geometry-conditioned refinement in 3D space with a 3D point transformer rather than a 2D image transformer. We use two 2D baselines built on pre-trained VGGT [32] and MapAnything [14]. For VGGT, we follow POW3R [12] to convert it into a geometry-conditioned model: $\mathbf{X}_s$ is patchified, encoded with an MLP, and injected into the first multi-view attention layer. MapAnything already supports encoding additional geometry signals. We fine-tune both models on the same ScanNet++ training set. Despite these 2D variants achieve the strongest performance on the **within-domain ScanNet++** test set, their accuracy degrades substantially on the **cross-domain ETH3D** and **out-of-domain 4D-DRESS**. In contrast, our 3D point transformer effectively generalises across diverse domains. Finally, for different 3D backbones, we find that Point Transformer v3 (PTv3) [37] outperforms Minkowski Engine [2].

Input encodings. We evaluate the impact of our input encoding strategies using the partial point cloud $\mathbf{X}_s$ from our SfM pipeline and the dense prediction $\mathbf{X}_d$. (1) Without using $\mathbf{X}_s$ as guidance, the network still learns to unconditionally denoise $\mathbf{X}_d$ on ScanNet++, but fails to generalise to other datasets. (2) Introducing $\mathbf{x}_{d\rightarrow s}$ (Eq. 8 of the main paper) to make the network aware of the correspondence between the guidance and initial prediction proves crucial. (3) Encoding $\mathbf{X}_s$ as $\mathbf{z}_s$ and providing it jointly with $\mathbf{z}_d$ stabilises training and improves overall performance. Tab. 4 demonstrates that all introduced encodings are essential for achieving high-quality results.

Patch-based processing. We study the impact of GGPT's patch size by varying the ratio r between the radii of the input patch and the scene. For larger patch sizes $r \geq 0.6$, GPU memory becomes a bottleneck, and we must reduce the number of input points from 400k to 100k. In this case, we randomly subsample 100k points from each cropped region as the input chunk. We find that a slightly smaller patch size $r = 0.2$ shows optimal performance across different datasets, as shown in Tab. 4. This behaviour may stem from two factors: (1) smaller patches concentrate the model's capacity on capturing fine-grained geometric structure, and (2) partitioning the scene into numerous small patches effectively serves as a form of data augmentation, increasing sample diversity and enhancing generalisation.

B.3. Performance Across Varying SfM Input Densities

As discussed in Section 3.1, adjusting the number of matches and the filtering threshold allows us to control the

Table 5. **Performance of GGPT across Varying SfM Input Densities.** We vary the SfM configuration, including the matcher type, DLT threshold ϵ_{DLT}, and reprojection threshold ϵ_{reproj}, to obtain SfM point clouds of different coverage (%). We report AUC@5/10 cm on within-domain **ScanNet++** and cross-domain **ETH3D** 8-view splits and AUC@1/5 cm (% ↑) on out-of-domain **4D-DRESS**. Our GGPT is robust to SfM points of various densities, and can improve the complete reconstruction even when the SfM point is extremely sparse, *e.g.* 8.5% on ScanNet++.

Dense Matcher	ϵ_{DLT}	ϵ_{reproj}	Point Cloud	SNt++		ETH3D		4DDS	
				Coverage (%)	AUC (%)	Coverage (%)	AUC (%)	Coverage (%)	AUC (%)
–	–	–	VGGT [32]($\mathbf{X}_d$)	100	19/32	100	23/36	100	10/45
RoMa	0.6	2	Our SfM ($\mathbf{X}_s$)	8.5	6 / 7	29	20/23	60	55/56
RoMa	0.6	2	+ GGPT ($\hat{\mathbf{X}}_d$)	100	26/41	100	39/53	100	59/67
RoMa	0.1	4	Our SfM ($\mathbf{X}_s$)	21	12/15	54	35/41	73	62/64
RoMa	0.1	4	+ GGPT ($\hat{\mathbf{X}}_d$)	100	31/45	100	46/59	100	**66/77**
RoMa+UFM	0.6	2	Our SfM ($\mathbf{X}_s$)	60	33/41	63	35/44	70	52/58
RoMa+UFM	0.6	2	+ GGPT ($\hat{\mathbf{X}}_d$)	100	43/58	100	43/57	100	56/67
RoMa+UFM	0.1	4	Our SfM ($\mathbf{X}_s$)	66	36/45	76	43/54	84	61/69
RoMa+UFM	0.1	4	+ GGPT ($\hat{\mathbf{X}}_d$)	100	**45/60**	100	**47/61**	100	63/73

density and accuracy of the SfM estimate $\mathbf{X}_s$, which serves as geometric guidance for GGPT. In this section, we examine how GGPT's final dense predictions vary with different input densities of $\mathbf{X}_s$. The results show that GGPT remains robust and delivers strong performance across a broad spectrum of SfM input densities.

Experimental setup. We control the density and accuracy of $\mathbf{X}_s$ using three factors. (1) Dense Matchers: As shown in Tab. 3, using only RoMa [6] for matching leads to more accurate yet sparser points than using it together with UFM [40]. (2) DLT Confidence Threshold (ϵ_{DLT}): the confidence threshold to pre-filter matches for DLT. A higher value yields more accurate yet sparser $\mathbf{X}_s$. (3) ϵ_{reproj}: the 2D reprojection error threshold to post-filter points after DLT. Increasing this threshold also results in more accurate but sparser $\mathbf{X}_s$. These factors were combined to generate four sets of $\mathbf{X}_s$ with various degrees of density as shown in Tab. 5. These $\mathbf{X}_s$ offer geometry guidance to different ratios of the total $N \times H \times W \times 3$ pixels. For comparison with the dense predictions, we also compute the AUC of each $\mathbf{X}_s$, which corresponds to the average recall at various error thresholds. We then feed each $\mathbf{X}_s$ together with the VGGT prediction $\mathbf{X}_d$ into GGPT and evaluate the final AUC of GGPT's complete prediction $\hat{\mathbf{X}}_d$.

Quantitative results. Tab. 5 presents the pixel coverage and AUC of the VGGT dense prediction $\mathbf{X}_d$, the various SfM partial-point estimates $\mathbf{X}_s$, and the GGPT reconstruction $\hat{\mathbf{X}}_d$ obtained when combining VGGT and SfM inputs. First, when using only RoMa matches with a strict filtering scheme, SfM yields a very small yet accurate subset of points. For example, with ϵ_{DLT}=0.6 and ϵ_{reproj}=2, SfM recovers points for merely 8.5% of pixels, which contributes to only an AUC@1cm of 6%. Despite such sparse guidance, GGPT still substantially enhances the dense reconstruction, raising VGGT's AUC@5 cm from 19% to 26%. Relaxing the filtering thresholds (ϵ_{DLT}=0.1, ϵ_{reproj}=4) increases SfM coverage (21% on ScanNet++), albeit with additional noise. Even so, GGPT continues to improve the dense prediction, achieving an AUC@5 cm of 31%. The same behaviour is observed across all datasets, demonstrating that GGPT is robust to both sparsity and noise in the SfM input. Fig. 1 provides qualitative examples showing how GGPT effectively exploits highly sparse SfM guidance to refine VGGT outputs.

Next, we consider combining RoMa and UFM matches (implementation details in Sec. E.1). This increases SfM point density but also introduces additional noise, consistent with our observation that UFM attains higher recall but lower accuracy than RoMa (Tab. 3). On 4D-DRESS, incorporating UFM increases SfM coverage yet slightly lowers AUC@1 cm, which in turn reduces GGPT performance compared with using RoMa alone—although GGPT still clearly surpasses both the raw SfM points and VGGT predictions. On ScanNet++ and ETH3D, however, GGPT benefits from the richer geometric cues and achieves the best dense reconstructions. Consequently, our default inference setting uses only RoMa on 4D-DRESS and the RoMa s+ UFM ensemble on other datasets.

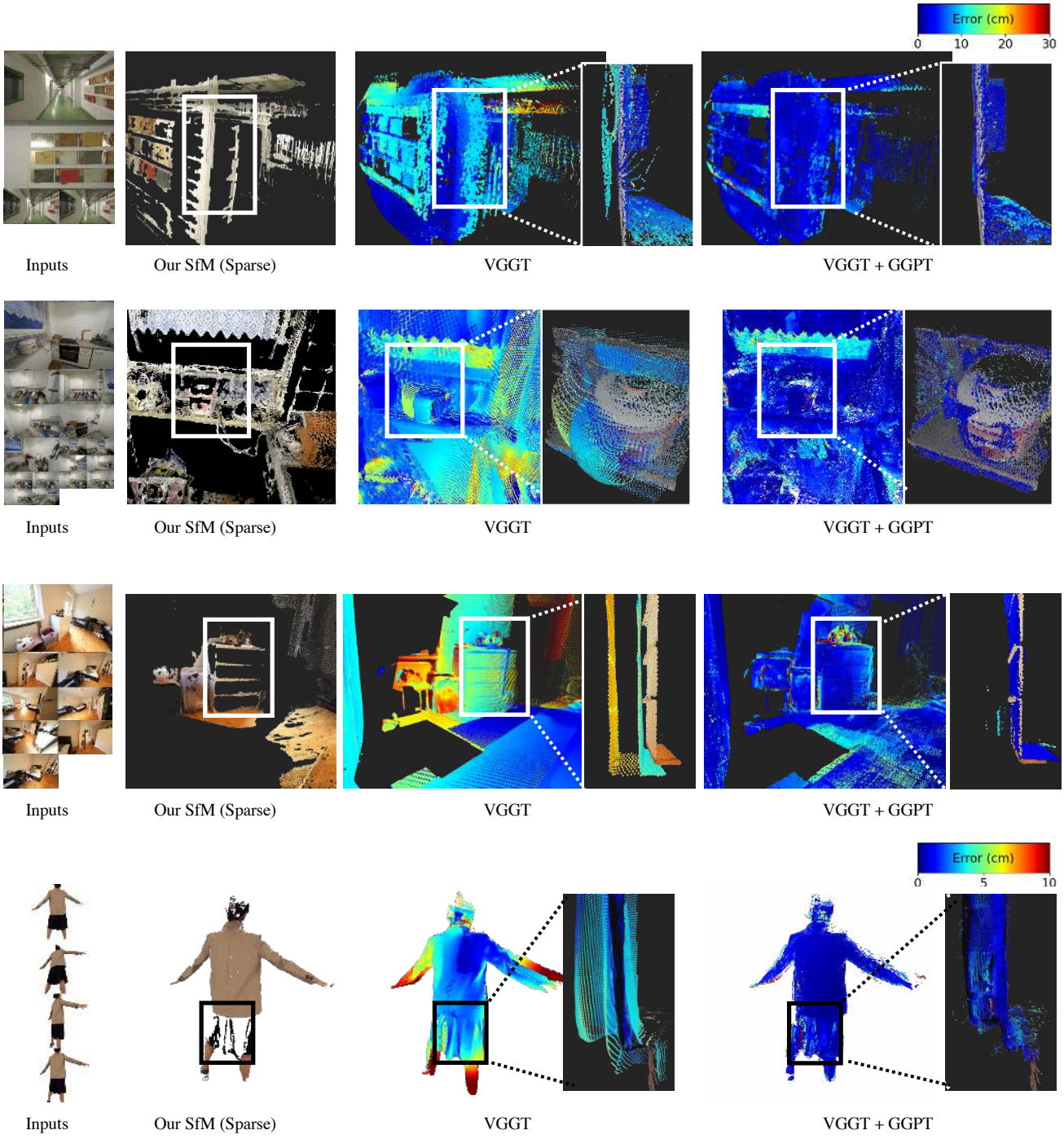

Figure 1. **GGPT's Robustness to Sparse Guidance.** Test examples from ETH3D [25] (1st row), ScanNet++ [39] (2nd and 3rd rows), and 4D-DRESS [35] (4th row). Points with predicted confidence below the 10% quantile are filtered. SfM points are produced with matchings from RoMa [6]. We visualise the SfM points and compare error maps before and after GGPT refinement. In the zoomed-in regions, the ground truth (coloured by input RGBs) is overlaid with the predictions (coloured by error) to highlight how our method corrects misaligned input points. SfM guidance provides little support in textureless regions, such as white walls or black cloth, but GGPT effectively propagates sparse guidance to unguided areas, improving the overall accuracy of the point map. Quantitative evaluation can be found in Tab. 5.

Table 6. **Comparison of GPU Memory Consumption and Inference Speed.** (a) We show the per-stage GPU memory and running time of our pipeline. (b) We compare our method with feed-forward method [32] and hybrid methods [5, 8] across varying view counts.

	Peak GPU Memory (GB) ↓			Inference Time (s) ↓			AUC@5/10 cm (%) ↑		
	4 Views	8 Views	16 Views	4 Views	8 Views	16 Views	4 Views	8 Views	16 Views
(a) **Pipeline Breakdown**									
VGGT Initialisation	6.9	7.0	8.0	0.1	0.2	0.3	–	–	–
Our SfM									
Dense Matcher - RoMa [6]	1.4	1.9	3.1	1.9	9.0	36	–	–	–
Dense Matcher - RoMa v2 [7]	4.9	5.3	6.1	1.5	3.1	10	–	–	–
Dense Matcher - UFM [40]	2.2	2.7	4.1	0.5	1.7	8.8	–	–	–
Bundle Adjustment	0.3	0.8	2.1	1.0	1.9	15	–	–	–
DLT Triangulation	0.7	3.1	5.9	<0.1	0.1	0.3	–	–	–
GGPT Refinement	1.6	3.1	4.8	2.6	4.1	6.1	–	–	–
(b) **Comparison with Baselines**									
MAST3R-SfM [5]	**3.3**	**3.3**	**3.3**	21	52	115	37/50	39/51	40/54
MAtCha [8]	3.3	3.6	9.4	48	102	242	40/52	41/53	42/56
VGGT [32]	6.9	7.0	8.0	**0.1**	**0.2**	**0.3**	27/41	23/36	19/32
VGGT + Ours (RoMa + UFM)	6.9	7.0	8.0	7	18	70	41/55	47/61	49/63
VGGT + Ours (RoMa v2)	6.9	7.0	8.0	5	9	34	**56/66**	**58/68**	**63/73**

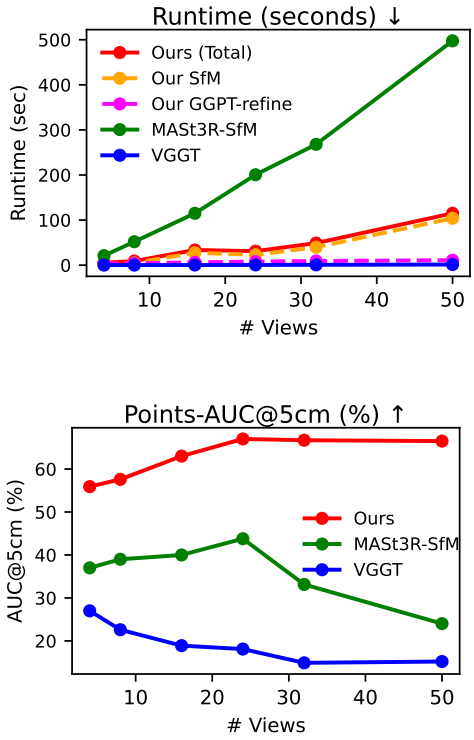

Figure 2. **Performance vs. number of views on ETH3D.** Using the latest dense matcher RoMa v2 [7], our method consistently surpasses the baselines with a favourable runtime profile across a wide range of input views, scaling effectively up to 50.

C. Efficiency Analysis

We provide an analysis of GPU memory consumption and inference time of our method on the ETH3D test set in Tab. 6.

C.1. Pipeline Breakdown

Our dense point map prediction pipeline comprises three stages: feed-forward initialisation, SfM reconstruction, and GGPT refinement. We use VGGT [32] as the representative feed-forward model for analysis, as other feed-forward approaches [14, 36] exhibit comparable inference costs.

GPU memory consumption. VGGT demands the most GPU memory, driven by full self-attention across many multi-view patches and a deep hierarchy of attention layers. The dense matcher [6, 40], bundle adjustment, DLT triangulation, and GGPT refinement incur smaller memory footprints. The stages of DLT triangulation and GGPT refinement also support iterative batch processing: input images or matches can be divided into independent batches to reduce memory usage when handling larger numbers of views.

Inference time. The dense matcher RoMa [6] is the dominant source of running time in our pipeline, as it is applied to all image pairs. This cost can be reduced by restricting matching to overlapping views. Bundle adjustment is lightweight, requiring less than two seconds for up to eight views, and for sixteen or more views its runtime can be further reduced by lowering n_{BA} to 512, as shown in Tab. 3. DLT triangulation adds only negligible overhead. GGPT

refinement over all patches contributes additional seconds. After the initial submission, we find that adopting more recent RoMa v2 [7] as the dense matcher further improves efficiency and accuracy.

C.2. Comparison with baselines

Compared with feed-forward models such as VGGT [32], our method maintains a similar memory footprint while introducing a modest increase in inference time, yielding substantially higher geometric accuracy. Although this makes the approach slower than pure feed-forward alternatives, we believe the additional runtime represents a worthwhile trade-off in applications where precise geometry is critical.

Additionally, with RoMa v2 as the dense matcher, our method can scale to more views while maintaining a favourable speed-accuracy trade-off. As shown in Fig. 2, when scaling to 50 input views, our approach is five times faster than MASt3R-SfM and substantially outperforms the baselines across varying numbers of views.

D. Comparison with more SfM methods

D.1. Comparison with Dense-SfM [17]

Similar to our SfM, Dense-SfM [17] adopts RoMa dense matcher to extract pairwise correspondences. In contrast with our SfM, Dense-SfM integrates dense matchings into an incremental COLMAP pipeline, requires a Gaussian Splatting [15] process to extend multi-view tracks, and performs iterative track refinement and global BA. We compare our SfM with Dense-SfM to demonstrate the effectiveness and efficiency of our method.

Since the Dense-SfM code is not publicly available, we follow the evaluation protocol in their paper and evaluate our SfM on the same benchmarks, *i.e.* the ETH3D [25] sequences with complete input views and the IMC [13] test set. We use RoMa for exhaustive matching, consistent with their setting, and compare camera pose accuracy with the results reported in Tab. 2 of their paper. Since both methods use the same matching procedure, we exclude matching time from the runtime comparison for clarity. Dense-SfM further reports around 5 minutes for 3D Gaussian training on ETH3D (Sec. 8.1 of their supplementary), in addition to iterative track refinement and BA, whereas our SfM requires only a single sparse BA.

Table 7. **Comparison with Dense-SfM.**

Method	**ETH3D** ($\sim$ 36 views)		**IMC** ($\sim$ 9 views)	
	AUC@1/3°	Runtime (s)	AUC@3/5°	Runtime (s)
Dense-SfM	61/78	>300	48/61	unavailable
Our SfM	**81/85**	**38**	**56/66**	**1.5**

Tab. 7 shows that our method achieves higher accuracy while being substantially faster, thanks to our feed-forward initialisation and effective track selection strategy.

E. Additional Implementation Details

E.1. Structure From Motion

Image Resolutions. All images are resized by setting a target width while preserving the original aspect ratio. For the feed-forward models [8, 32, 36], we use a width of 518 pixels. For the RoMa dense matcher [6], we provide full-HD inputs (width = 1920), which we find improves matching accuracy. For the UFM matcher [40], we use the UFM-Refine model[3], which internally resizes inputs to width = 560. All output matches are then resized to width = 518. Bundle adjustment, DLT triangulation, GGPT refinement, and final evaluation are performed at this resolution.

Dense matchers ensemble. We combine the match predictions from RoMa [6] and UFM [40] to produce the global correspondences and confidence tensors $(\mathbf{T}, \mathbf{C})$ in Eq. (2) of the main paper. Concretely, we denote the tensors predicted by RoMa and UFM by $(\mathbf{T}_1, \mathbf{C}_1)$ and $(\mathbf{T}_2, \mathbf{C}_2)$ respectively. We compute their cycle consistency tensor (Eq. (3) of the main paper) and denote them as ε_1 and ε_2.

$$\varepsilon_1[i,k,\mathbf{u}] = \left\|\mathbf{T}_1[k,i,\mathbf{T}_1[i,k,\mathbf{u}]] - \mathbf{u}\right\|_2 \quad (1)$$

$$\varepsilon_2[i,k,\mathbf{u}] = \left\|\mathbf{T}_2[k,i,\mathbf{T}_2[i,k,\mathbf{u}]] - \mathbf{u}\right\|_2 \quad (2)$$

Each query pixel $\mathbf{u}$ in source view i has two predicted correspondences in target view k from the two matchers. We choose the correspondence with a smaller cycle consistency error. Formally,

$$\mathbf{T} = [\varepsilon_1 < \varepsilon_2] \odot \mathbf{T}_1 + [\varepsilon_1 > \varepsilon_2] \odot \mathbf{T}_2 \quad (3)$$

$$\mathbf{C} = [\varepsilon_1 < \varepsilon_2] \odot \mathbf{C}_1 + [\varepsilon_1 > \varepsilon_2] \odot \mathbf{C}_2 \quad (4)$$

Ensembling two matchers improves the completeness of SfM points, as shown in Sec. B.1 and Tab. 3, and it subsequently improves the final dense prediction, as shown in Sec. B.3 and Tab. 5.

Direct linear triangulation. We provide the mathematical formulation of the direct linear triangulation (DLT) used in our SfM pipeline. The objective of DLT is to estimate the 3D homogeneous points $\tilde{\mathbf{X}} \in \mathbb{R}^{M\times 4}$ that minimise the 2D reprojection error, given multi-view tracks in 2D homogeneous coordinates $\mathbf{T} \in \mathbb{R}^{N\times M\times 2}$ and its visibility mask $\mathbf{M} \in \mathbb{R}^{N\times M}$ [4], as well as the BA-estimated intrinsics $\mathbf{K} \in \mathbb{R}^{N\times 3\times 3}$ and extrinsics $\mathbf{E} \in \mathbb{R}^{N\times 3\times 4}$. Here N denotes the number of views, and M denotes the number of multi-view tracks, which is also the number of points to estimate.

[3] https://huggingface.co/infinity1096/UFM-Refine

[4] Here, $(\mathbf{T}, \mathbf{M})$ are derived from $(\mathbf{T}_{\text{DLT}} \in \mathbb{R}^{N\times H\times M\times 2}, \mathbf{M}_{\text{DLT}} \in \mathbb{R}^{N\times H\times M})$ as introduced in Sec. 3.1 of the main paper.

For each 3D point $\tilde{\mathbf{X}}_{[i]} \in \mathbb{R}^4$, its 2D reprojections in N views can be represented in 2D homogeneous coordinates as $\mathbf{P}\tilde{\mathbf{X}}_{[i]} \in \mathbb{R}^{N\times 3}$, where $\mathbf{P} = \mathbf{KE}$ is the multi-view projection matrix. As the 2D projections should be close to the observation $\mathbf{T}_{[:,i]} \in \mathbb{R}^{N\times 2}$ in visible views, the objective of DLT is to estimate

$$\tilde{\mathbf{X}}_{[i]} = \underset{\tilde{\mathbf{X}}_{[i]}}{\arg\min} \left\| \mathbf{M}_{[:,i]} \odot \left(\mathrm{deh}(\mathbf{P}\tilde{\mathbf{X}}_{[i]}) - \mathbf{T}_{[:,i]} \right) \right\| \quad (5)$$

where $\mathrm{deh}(\cdot)$ is the de-homogenising operation.

$$\mathrm{deh}(\mathbf{Y}) = \mathbf{Y}_{[:,[0,1]]} / \mathbf{Y}_{[:,2]} \quad (6)$$

This can be approximated by a least-squares problem

$$\tilde{\mathbf{X}}_{[i]} \sim \underset{\substack{\mathbf{x}\in\mathbb{R}^4 \\ \|\mathbf{x}\|=1}}{\arg\min} \left\| \mathbf{A}_i \mathbf{x} \right\|. \quad (7)$$

$$\mathbf{A}_i = \begin{bmatrix} \vdots \\ \mathbf{M}_{[n,i]}(\mathbf{T}_{[n,i,0]}\,\mathbf{P}_{[n,2,:]} - \mathbf{P}_{[n,0,:]}) \\ \mathbf{M}_{[n,i]}(\mathbf{T}_{[n,i,1]}\,\mathbf{P}_{[n,2,:]} - \mathbf{P}_{[n,1,:]}) \\ \vdots \end{bmatrix}_{n=1,\ldots,N}. \quad (8)$$

Here, $\sim$ denotes equality up to scale. The solution has a closed-form via singular value decomposition. Moreover, all $\tilde{\mathbf{X}}_{[i]}$ and $\mathbf{A}_i$ can be processed in parallel to solve for $\mathbf{X}$ efficiently using CUDA tensor operations in PyTorch.

After DLT triangulation, for each estimated point, we compute its maximal 2D reprojection error and minimal triangulation angle across all visible views, both of which can be efficiently evaluated via tensor operations. Points with maximal 2D reprojection error exceeding 4 pixels or minimal triangulation angle below 3° are discarded. The remaining points constitute our final geometry guidance, denoted as $\mathbf{X}_s \in \mathbb{R}^{M'\times 3}$.

To establish correspondences between the geometry estimate $\mathbf{X}_s \in \mathbb{R}^{M'\times 3}$ and the initial dense prediction $\mathbf{X}_d \in \mathbb{R}^{N\times H\times W\times 3}$, we project $\mathbf{X}_s$ back to their visible views, round the 2D coordinates, and assign the 3D coordinates to the corresponding integer pixels. For pixels receiving multiple projections, we compute the average of the 3D estimates.

E.2. Geometry-Grounded Point Transformer

Architecture. The point transformer encoder begins with an MLP embedding layer, followed by five down-pooling and four up-pooling stages, producing features of dimensionality $V = 96$. The down-pooling stages include $(2, 2, 2, 6, 2)$ attention blocks with hidden dimensions $(64, 96, 128, 256, 512)$, with each stage followed by a down-sampling grid-pooling layer. The up-pooling stages contain $(2, 2, 2, 2)$ attention blocks with hidden dimensions $(256, 128, 96, 96)$, each stage preceded by an up-sampling grid-pooling layer. The point cloud is voxelised at a grid resolution of 384, with strides of $(2, 2, 2, 2)$ for the grid-pooling layers. For details on the attention blocks and grid pooling, see [37]. Each point feature output by the point transformer backbone is concatenated with its input embedding $\mathbf{z}$ and passed through a three-layer MLP with two intermediate ReLU activations and 256 hidden dimensions, producing four outputs: three for 3D coordinate residuals and one for confidence. The complete GGPT model has a total of 53 million parameters.

Training dataset. We curate the training dataset by sampling 20k multi-view sequences from the 856 training scenes of ScanNet++ [39]. Each multi-view sequence consists of 4-16 input views, which are randomly selected following the view-sampling algorithm in [14]. For each sequence, we use VGGT [32] to predict dense predictions $\mathbf{X}_d$ and provide camera-pose initialisation for our SfM. We use the configurations described in Sec. E.1 to run our SfM pipeline except we only use RoMa [6] matcher for dense correspondence extraction. The training dataset generation used eight NVIDIA GH200 GPUs for one day.

Training hyperparameters. We set the weight of the identity consistency loss as $\lambda_{\text{id}} = 1$, and the weight of the confidence term as $\alpha = 0.2$. We use Adam optimiser with a learning rate of $1e-4$ and a batch size of 24. The model is trained for 100k iterations, which is performed on eight NVIDIA GH200 GPUS for one day.

Patch-based processing. We patchify the input point cloud during both training and inference. We calculate the scene radius R as $3 \times \mathrm{std}(\mathbf{X}_s)$ and sample each input chunk with a half width of $r = 0.2R$. To avoid out-of-memory issue, we input a maximal number of 400k points for each chunk. If the sampled chunk contains more points than this, which happens sporadically, we reduce the width by $\times 0.9$. During training, we randomly sample a chunk from each scene by sampling an anchor point from $\mathbf{X}_s$ and cropping a box centred at it. Each batch thus contains 24 chunks from 24 possibly different scenes. During inference, we cover the whole point cloud by iteratively sampling an anchor point from $\mathbf{X}_s$ and taking its surrounding points. At each iteration, we only sample anchors which are not covered previously, but still include other points in the box even if they have been processed. This ensures some overlapping between chunks while saving redundancy. sampling terminates once all points in $\mathbf{X}_s$ are covered. Empirically, this yields around 50 chunks for the whole scene on average. For unprocessed points in $\mathbf{X}_d$, which usually make up fewer than 5% of the scene and are regions without any valid geometry constraints, we retain their initial dense predictions in the final output.

E.3. Point Evaluation

Ground-truth reference. We use the ground-truth metric point map $\mathbf{X}_{\text{gt}} \in \mathbb{R}^{N\times H\times W\times 3}$ at a resolution of $W = 518$ as the reference for evaluation. For ETH3D [25] and ScanNet++ [39], the provided depth maps are first undistorted to linear camera models and then unprojected to obtain the ground-truth point maps. For T&T [16], we employ PyTorch3D to render the provided laser point clouds into 2D depth maps, remove occluded pixels, and subsequently unproject them back to 3D. For Blender-rendered datasets, including 4D-DRESS [35] and MV-dVRK [1], we unproject the ground-truth depth maps directly output by Blender. Owing to the incompleteness of laser scans in real-world datasets or the invalid background regions in synthetic datasets, each $\mathbf{X}_{\text{gt}}$ is associated with a valid mask $\mathbf{M}_{\text{gt}} \in [0,1]^{N\times H\times W}$. Accuracy is evaluated exclusively within the valid regions defined by $\mathbf{M}_{\text{gt}}$.

Procrustes alignment. Given the ground-truth metric point map $\mathbf{X}_{\text{gt}} \in \mathbb{R}^{N\times H\times W\times 3}$ and predicted scale-invariant point map $\mathbf{X}_{\text{pred}} \in \mathbb{R}^{N\times H\times W\times 3}$, we first estimate the optimal similarity transformation $(s, \mathbf{R} \in SO(3), \mathbf{t} \in \mathbb{R}^3)$ that aligns $\mathbf{X}_{\text{pred}}$ to $\mathbf{X}_{\text{gt}}$. Following prior work [18, 32, 33], we use the Umeyama [28] algorithm to compute the alignment using $\mathbf{M}_{\text{gt}} \odot \mathbf{X}_{\text{gt}}$ and $\mathbf{M}_{\text{gt}} \odot \mathbf{X}_{\text{pred}}$. To improve robustness against outliers in the prediction, we employ the RANSAC-based robust Umeyama implemented by pycolmap [24][5] with a minimum inlier ratio of 0.8 and a maximum error of 3 cm. The aligned predicted points are denoted as $\mathbf{X}'_{\text{pred}} = s\mathbf{R}\mathbf{X}_{\text{pred}} + \mathbf{t}$.

Metrics. For each pixel with valid ground truth, we compute the 3D Euclidean error as

$$\varepsilon[n, \mathbf{u}] = \|\mathbf{X}_{\text{gt}}[n, \mathbf{u}] - \mathbf{X}'_{\text{pred}}[n, \mathbf{u}]\|_2, \tag{9}$$

where n indexes frames and $\mathbf{u}$ indexes pixels.

Given an error threshold τ, the recall at threshold τ (Recall@τ) is defined as the fraction of pixels with error below τ:

$$\text{Recall@}\tau = \frac{1}{|M|} \sum_{(n,\mathbf{u})\in M} \mathbf{1}\{\varepsilon[n, \mathbf{u}] < \tau\}, \tag{10}$$

where $M = \{(n, \mathbf{u}) \mid \mathbf{M}(n, \mathbf{u}) = 1\}$ is the set of pixels with valid ground truth, and $|M|$ is its cardinality.

Finally, the area under the curve up to threshold τ (AUC@τ) is computed by averaging recall over all integer thresholds $1, 2, \ldots, \tau$:

$$\text{AUC@}\tau = \frac{1}{\tau} \sum_{k=1}^{\tau} \text{Recall@}k. \tag{11}$$

[5] pycolmap documentation

F. Depth Evaluation

In addition to point-based metrics, we can also evaluate depth prediction. Specifically, after the Procrustes alignment in Sec. E.3, we use the estimated similarity transformation to transform the camera extrinsics into the same world coordinate system as $\mathbf{X}'_{\text{pred}}$. We then project $\mathbf{X}'_{\text{pred}}$ to each view to obtain per-view depth predictions, which are compared against ground truths. Note that no further scaling is applied to the predicted depths, as the similarity transformation already accounts for scale.

Following standard depth evaluation protocol [42], we report the absolute relative error (Rel) and inlier ratio at 1.01 and 1.03 (τ) before and after GGPT refinement. Tab. 8 shows that GGPT consistently improves the depths.

Table 8. **Depth metrics.** We report relative absolute error (Rel %) and inlier ratio at 1.01/1.03 (τ %) on **cross-domain** and **out-of-domain** test sets.

	ETH3D		T&T		4DDress		MV-dVRK	
	Rel↓	τ ↑	Rel↓	τ ↑	Rel↓	τ ↑	Rel↓	τ ↑
VGGT	4.3	27/57	2.9	40/79	75.4	0/0	31.8	1/3
+ Ours	**2.3**	**63/85**	**2.3**	**58/85**	**7.9**	**72/81**	**11.4**	**40/60**
Pi3	3.7	27/60	2.9	38/75	8.6	7/20	9.4	10/29
+ Ours	**2.2**	**59/86**	**2.1**	**58/87**	**4.8**	**27/66**	**5.9**	**34/63**

G. Additional Qualitative Results

We visualise additional results in Fig. 3 (ETH3D), Fig. 4 (ScanNet++), Fig. 5 (T&T), Fig. 6 (MV-dVRK), and Fig. 7 (4DDress). We apply our method to existing dense reconstruction methods [5, 8, 14, 32, 36] and compare the error map of the point maps before and after our refinement. Note that we only visualise the error map for pixels with valid ground truth reference. Additionally, following prior work [14, 32, 36], we filter out points with predicted confidence below the 10% quantile.

Visual results demonstrate that our method can improve various baselines consistently across different datasets. On **ScanNet++** [39], the within-domain test set, Pi3 [36] achieves near-saturated performance because it is trained on a large corpus of similar indoor scenes, leaving little room for improvement. However, our method can still noticeably improve its performance on other datasets, particularly out-of-domain **4DDress** and **MV-dVRK** datasets.

With these visualisations, we also acknowledge that our method shows limitations in refining local details, exhibiting red error patterns in some detailed areas, failing to correct geometry in regions without geometry guidance, and sometimes exhibiting patchy artefacts. For these limitations, we further discuss possible solutions and future directions in the next section.

H. Limitations and Future Directions

End-to-end 3D reconstruction. Our pipeline currently consists of three sequential stages to obtain a multi-view consistent reconstruction. Because the SfM geometry guidance and the GGPT dense refinement operate in separate steps, any error in the SfM stage propagates directly into the refinement stage. As a result, GGPT struggles in regions where the geometry guidance is inaccurate or incomplete. A natural direction for future work is to develop an end-to-end approach that jointly estimates geometry based on multi-view constraints and predicts the dense reconstruction. One possibility is to integrate 2D image cues into the point embeddings and train the 3D point transformer to refine points directly through 2D photometric consistency.

Improving point transformer architecture. As described in Sec. 3.2, our current GGPT adopts patch-based processing strategy and confidence-weighted fusion. This saves memory consumption and allows us to process large-scale scenes. It is also used by point cloud denoising networks [23, 29]. Additionally, we find in our ablation (Tab. 4) that a small patch size enables the network to improve both fine-level accuracy and generalisation. However, this strategy also has two downsides. (1) It may result in some patch-wise artefacts and non-smooth transitions between different patches. (2) It sacrifices long-range spatial dependency, since points in two non-overlapping patches cannot attend to each other. As a consequence, our prediction still exhibits error in regions which do not have geometry guidance and need to leverage guidance from distant areas. A future direction will be to develop a multi-scale hierarchical point transformer to improve the long-range dependency and global smoothness while maintaining the accuracy in local detail refinement.

Scaling up training datasets. Our current GGPT is trained solely on indoor scenes from ScanNet++ [39], yet already generalises to cross-domain [16, 25] and out-of-domain [1, 35] datasets. Thanks to our efficient SfM pipeline and scalable point transformer architecture, training GGPT on larger datasets is practical. Future work could incorporate more diverse datasets, such as Tartan [34] and BMVS [38], to further enrich the model with general 3D geometry priors, which is expected to improve sharp, fine-grained geometric details in the dense reconstruction.

Extending to large numbers of input views and dense captures. Our work targets 3D reconstruction from a limited set of unposed input views, characterised by discrete spatial distribution and random overlap. Accordingly, we conduct experiments using at most 16 input views. Scaling up to captures with hundreds or thousands of images—typical in large outdoor or indoor scenes—would require further optimisation to keep computation manageable. One avenue is to replace VGGT with recent memory-efficient feed-forward variants [4, 21, 27] for initialisation. Another is to avoid exhaustive pairwise matching by adopting the strategy of MAST3R-SfM [5], running matchers only on overlapping image pairs, with overlap predicted by a feed-forward model. Additionally, our focus on discrete input views means that we do not treat video-based settings, where frames provide continuous coverage with dense overlap. In such cases, camera poses are typically obtained with classical SfM pipelines [24, 26], and dense reconstruction, given known camera parameters, is usually handled by Multi-view Stereo methods [10, 11, 30, 41]. The reliance on camera parameters and the small triangulation angles involved (often below 10°) places MVS methods in a different regime from ours. Nonetheless, it would be interesting to explore as a future direction whether our idea of geometry-guided 3D point transformer could also benefit dense-overlap scenarios.

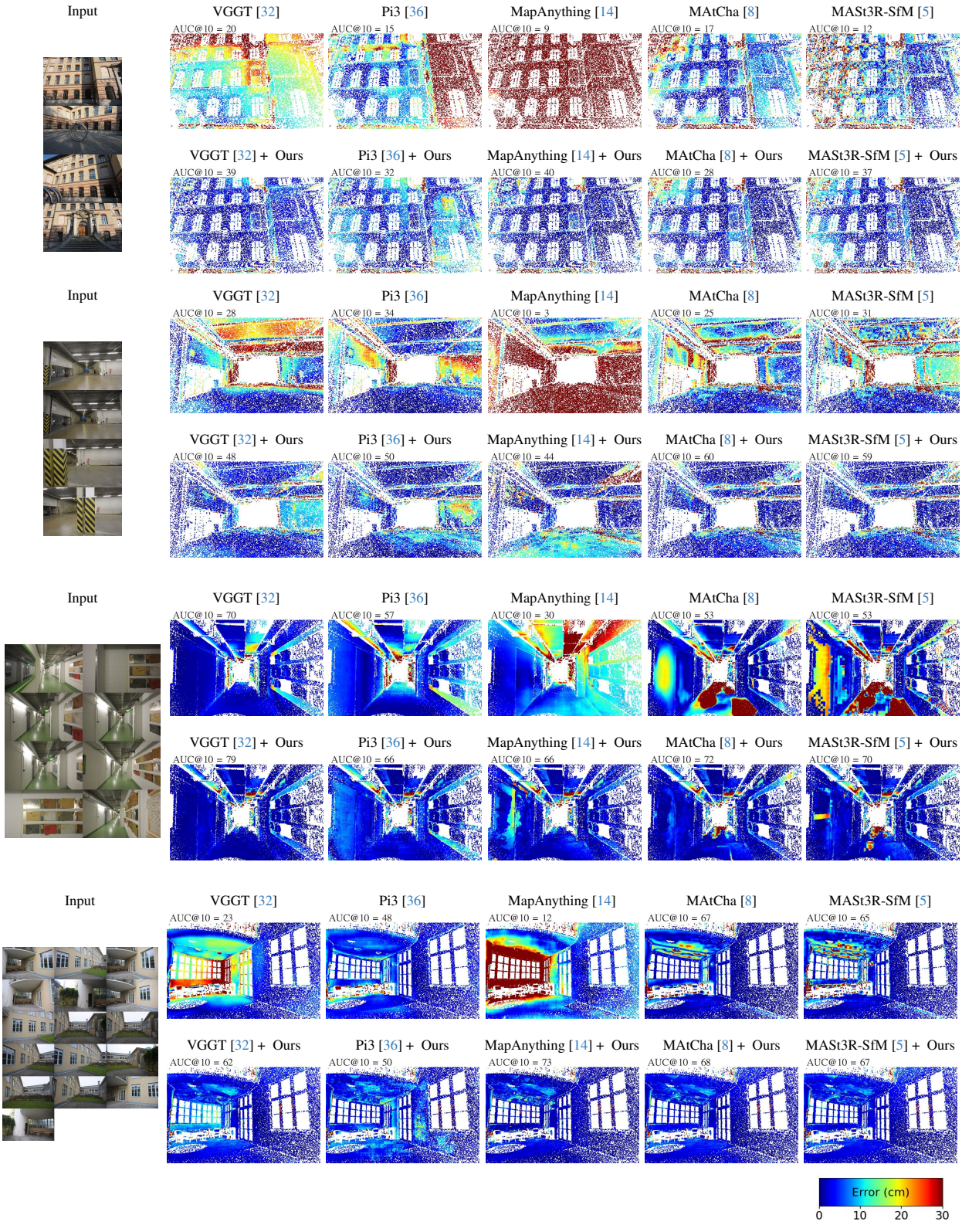

Figure 3. **Visual Results on ETH3D [25].** We present error maps for all baselines alongside our refined predictions and report the AUC@10 cm (%) for each scene. Points with confidence below the 10 % quantile are discarded. Because the laser ground truth is incomplete, some pixels in the error maps have no valid values.

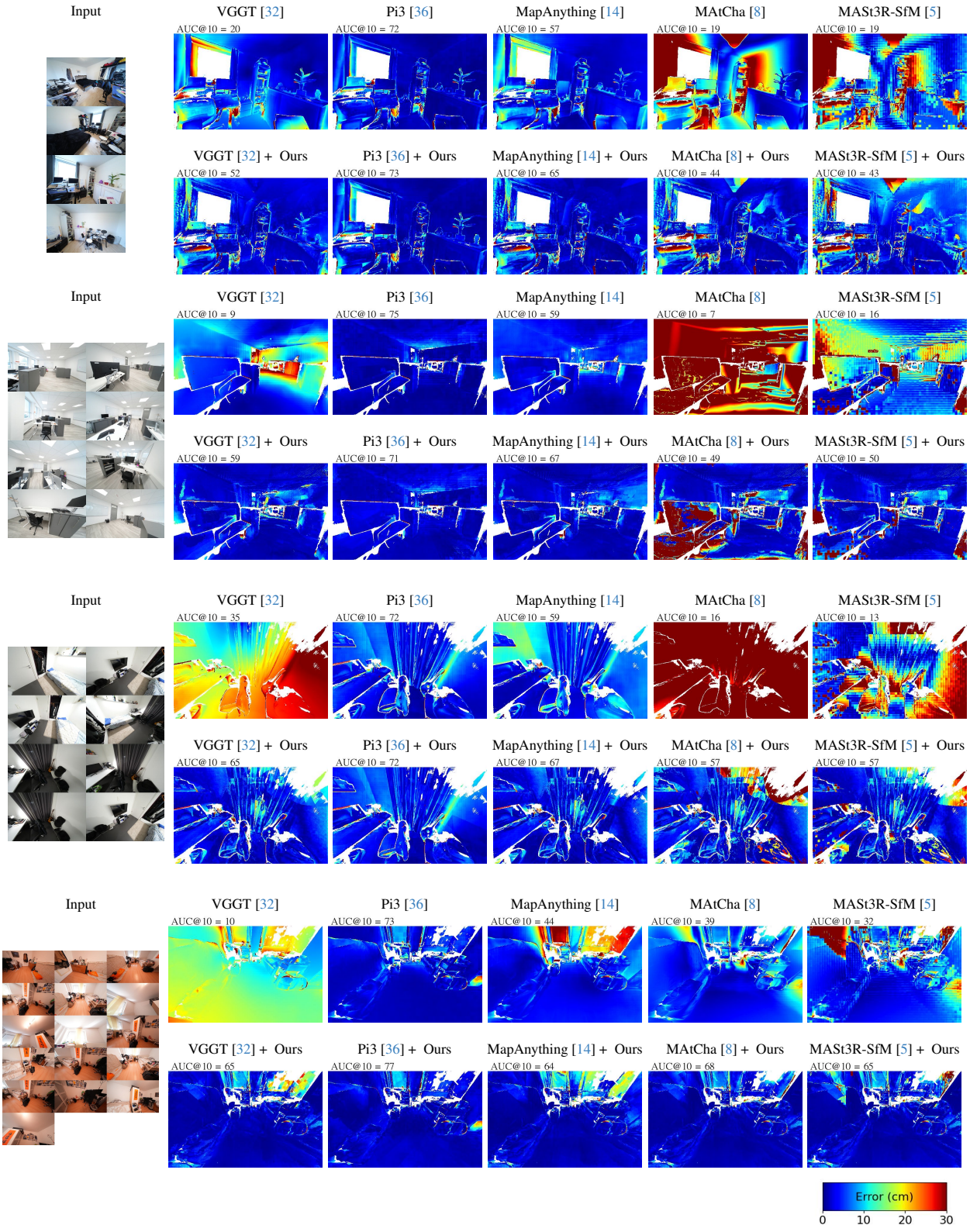

Figure 4. **Visual Results on ScanNet++ [39].** We compare the error maps between baselines and our refinement and report the AUC@10 cm (%) for each scene. We filter points with predicted confidence below the 10% quantile. On the within-domain ScanNet++, Pi3 [36] achieves saturated performance as it is trained on a large scale of similar indoor scenes, leaving limited room for further improvement. Nevertheless, our method still enhances its performance on other datasets, as demonstrated in the accompanying figures.

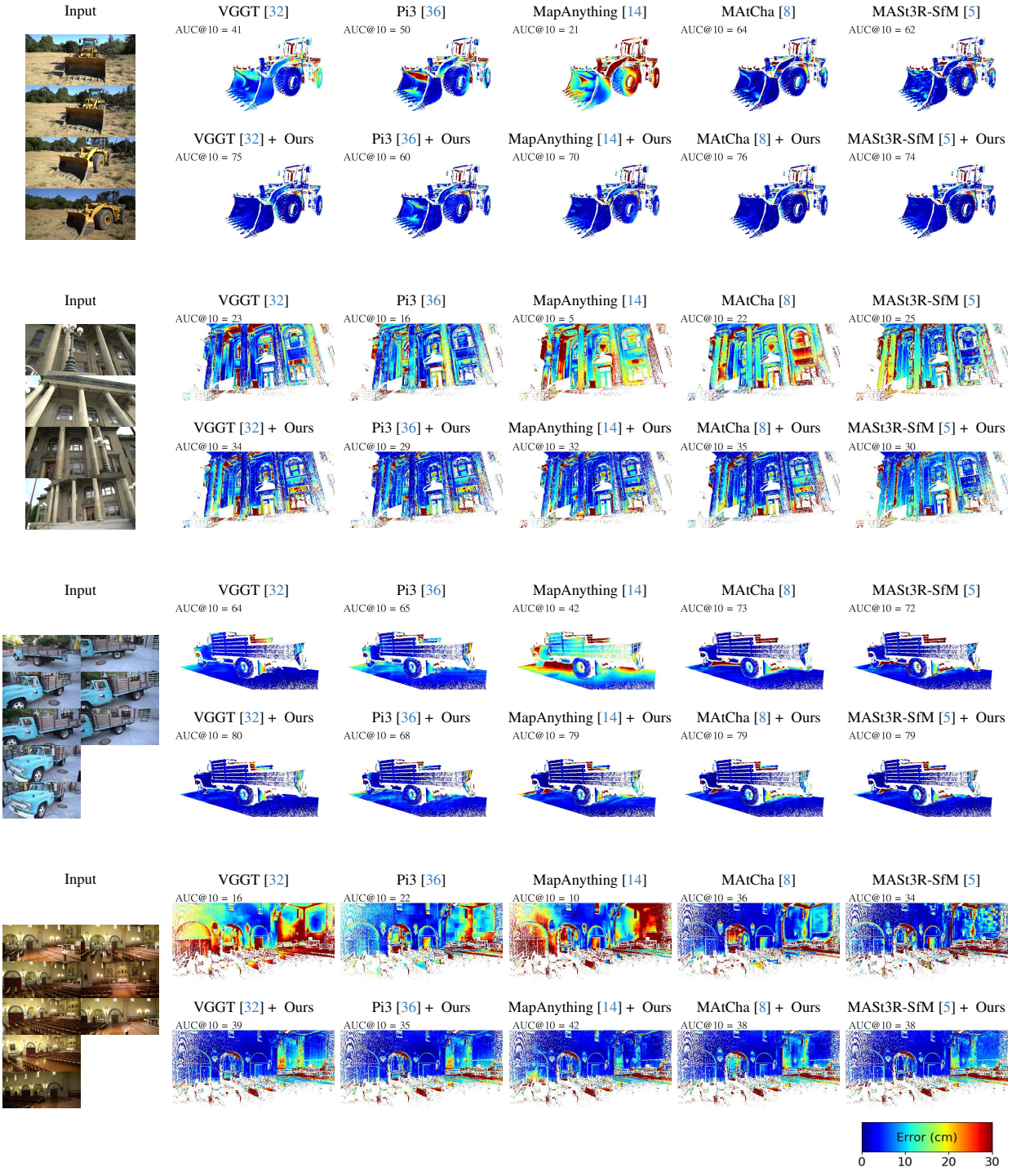

Figure 5. **More Visual Results on T&T [25].** We compare the error maps of the baselines with those of our refinement and report the AUC@10 cm (%) for each scene. Points with predicted confidence below the 10% quantile are removed. As the laser ground truth does not cover all pixels, some regions in the error maps remain unlabelled.

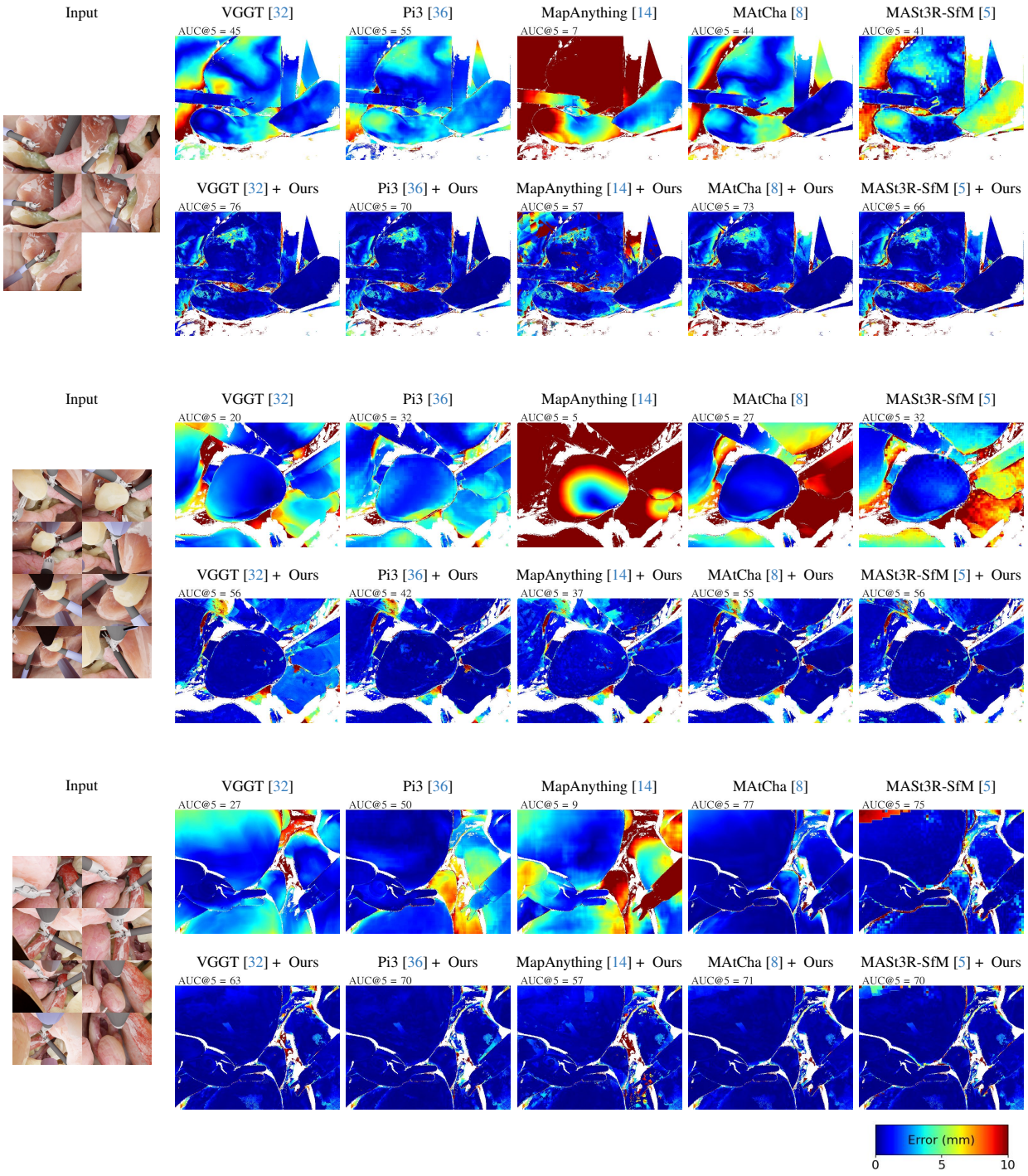

Figure 6. **More Visual Results on MV-dVRK [1].** We compare the error maps of the baselines with those of our refinement and report the AUC@5 mm (%) for each scene. Points with predicted confidence below the 10% quantile are discarded.

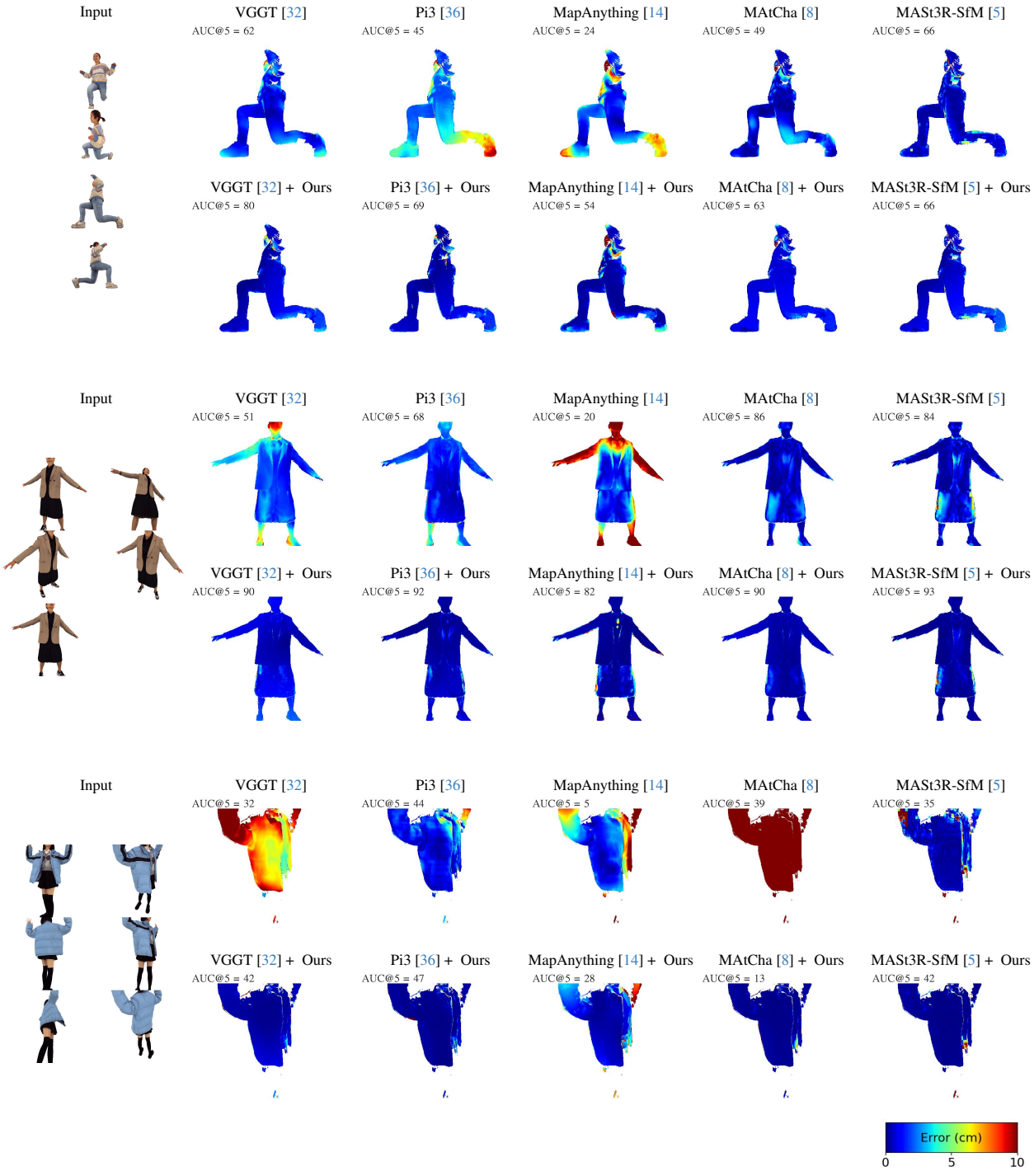

Figure 7. **More Visual Results on 4D-DRESS [35].** We compare the error maps of the baselines with those of our refinement and report the AUC@5 cm (%) for each scene. Points with predicted confidence below the 10% quantile are discarded.

References

[1] Guido Caccianiga, Sergey Prokudin, Bernard Javot, Yutong Chen, Omer Burak Aladag, Rachael L'Orsa, Yarden Sharon, Jens Rolinger, Ivan Capobianco, Siyu Tang, Anton Deguet, and Katherine J. Kuchenbecker. MV-dVRK: Integrated methods for multi-viewpoint surgical telerobotics. *In submission*, 2026. Available at: https://mv-dvrk.is.mpg.de/. 1, 9, 10, 14

[2] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *CVPR*, 2019. 3

[3] Ondřej Chum, Jiří Matas, and Josef Kittler. Locally optimized ransac. In *Joint pattern recognition symposium*, pages 236–243. Springer, 2003. 3

[4] Kai Deng, Zexin Ti, Jiawei Xu, Jian Yang, and Jin Xie. Vggt-long: Chunk it, loop it, align it – pushing vggt's limits on kilometer-scale long rgb sequences, 2025. 10

[5] Bardienus Pieter Duisterhof, Lojze Zust, Philippe Weinzaepfel, Vincent Leroy, Yohann Cabon, and Jerome Revaud. MASt3r-sfm: a fully-integrated solution for unconstrained structure-from-motion. In *3DV*, 2025. 1, 2, 6, 9, 10, 11, 12, 13, 14, 15

[6] Johan Edstedt, Qiyu Sun, Georg Bökman, Mårten Wadenbäck, and Michael Felsberg. RoMa: Robust Dense Feature Matching. In *CVPR*, 2024. 1, 2, 4, 5, 6, 7, 8

[7] Johan Edstedt, David Nordström, Yushan Zhang, Georg Bökman, Jonathan Astermark, Viktor Larsson, Anders Heyden, Fredrik Kahl, Mårten Wadenbäck, and Michael Felsberg. RoMa v2: Harder Better Faster Denser Feature Matching. *arXiv preprint arXiv:2511.15706*, 2025. 1, 2, 6, 7

[8] Antoine Guédon, Tomoki Ichikawa, Kohei Yamashita, and Ko Nishino. Matcha gaussians: Atlas of charts for high-quality geometry and photorealism from sparse views. *CVPR*, 2025. 1, 2, 6, 7, 9, 11, 12, 13, 14, 15

[9] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 3

[10] Po-Han Huang, Kevin Matzen, Johannes Kopf, Narendra Ahuja, and Jia-Bin Huang. Deepmvs: Learning multi-view stereopsis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 10

[11] Sergio Izquierdo, Mohamed Sayed, Michael Firman, Guillermo Garcia-Hernando, Daniyar Turmukhambetov, Javier Civera, Oisin Mac Aodha, Gabriel J. Brostow, and Jamie Watson. MVSAnywhere: Zero shot multi-view stereo. In *CVPR*, 2025. 10

[12] Wonbong Jang, Philippe Weinzaepfel, Vincent Leroy, Lourdes Agapito, and Jerome Revaud. Pow3r: Empowering unconstrained 3d reconstruction with camera and scene priors. In *CVPR*, 2025. 3

[13] Yuhe Jin, Dmytro Mishkin, Anastasiia Mishchuk, Jiri Matas, Pascal Fua, Kwang Moo Yi, and Eduard Trulls. Image matching across wide baselines: From paper to practice. *IJCV*, 2021. 7

[14] Nikhil Keetha, Norman Müller, Johannes Schönberger, Lorenzo Porzi, Yuchen Zhang, Tobias Fischer, Arno Knapitsch, Duncan Zauss, Ethan Weber, Nelson Antunes, Jonathon Luiten, Manuel Lopez-Antequera, Samuel Rota Bulò, Christian Richardt, Deva Ramanan, Sebastian Scherer, and Peter Kontschieder. MapAnything: Universal feed-forward metric 3D reconstruction, 2025. 1, 2, 3, 6, 8, 9, 11, 12, 13, 14, 15

[15] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM ToG*, 2023. 7

[16] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM ToG*, 2017. 2, 9, 10

[17] JongMin Lee and Sungjoo Yoo. Dense-sfm: Structure from motion with dense consistent matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6404–6414, 2025. 7

[18] Vincent Leroy, Yohann Cabon, and Jerome Revaud. Grounding image matching in 3d with mast3r, 2024. 1, 2, 9

[19] Haotong Lin, Sili Chen, Jun Hao Liew, Donny Y. Chen, Zhenyu Li, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth anything 3: Recovering the visual space from any views. *arXiv preprint arXiv:2511.10647*, 2025. 1, 2

[20] Philipp Lindenberger, Paul-Edouard Sarlin, Viktor Larsson, and Marc Pollefeys. Pixel-Perfect Structure-from-Motion with Featuremetric Refinement. In *ICCV*, 2021. 1

[21] Yang Liu, Chuanchen Luo, Zimo Tang, Junran Peng, and Zhaoxiang Zhang. Vggt-x: When vggt meets dense novel view synthesis, 2025. 10

[22] Zador Pataki, Paul-Edouard Sarlin, Johannes L. Schönberger, and Marc Pollefeys. MP-SfM: Monocular Surface Priors for Robust Structure-from-Motion. In *CVPR*, 2025. 1

[23] Marie-Julie Rakotosaona, Vittorio La Barbera, Paul Guerrero, Niloy J Mitra, and Maks Ovsjanikov. Pointcleannet: Learning to denoise and remove outliers from dense point clouds. In *Computer Graphics Forum*, 2020. 10

[24] Johannes Lutz Schönberger. Colmap – general-purpose structure-from-motion and multi-view stereo pipeline. https://colmap.github.io/, 2025. Version 3.13.0.dev0. 3, 9, 10

[25] Thomas Schöps, Johannes L. Schönberger, Silvano Galliani, Torsten Sattler, Konrad Schindler, Marc Pollefeys, and Andreas Geiger. A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *CVPR*, 2017. 1, 2, 5, 7, 9, 10, 11, 13

[26] Johannes L. Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016. 2, 3, 10

[27] You Shen, Zhipeng Zhang, Yansong Qu, and Liujuan Cao. Fastvggt: Training-free acceleration of visual geometry transformer. *arXiv preprint arXiv:2509.02560*, 2025. 10

[28] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *TPAMI*, 2002. 9

[29] Mathias Vogel, Keisuke Tateno, Marc Pollefeys, Federico Tombari, Marie-Julie Rakotosaona, and Francis Engelmann. P2p-bridge: Diffusion bridges for 3d point cloud denoising. In *ECCV*, 2024. 10

[30] Fangjinhua Wang, Qingshan Xu, Yew-Soon Ong, and Marc Pollefeys. Lightweight and accurate multi-view stereo with

confidence-aware diffusion model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025. 10

[31] Jianyuan Wang, Nikita Karaev, Christian Rupprecht, and David Novotny. Vggsfm: Visual geometry grounded deep structure from motion. In *CVPR*, 2024. 1

[32] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *CVPR*, 2025. 1, 2, 3, 4, 6, 7, 8, 9, 11, 12, 13, 14, 15

[33] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *CVPR*, 2024. 9

[34] Wenshan Wang, Delong Zhu, Xiangwei Wang, Yaoyu Hu, Yuheng Qiu, Chen Wang, Yafei Hu, Ashish Kapoor, and Sebastian Scherer. Tartanair: A dataset to push the limits of visual slam. 2020. 10

[35] Wenbo Wang, Hsuan-I Ho, Chen Guo, Boxiang Rong, Artur Grigorev, Jie Song, Juan Jose Zarate, and Otmar Hilliges. 4d-dress: A 4d dataset of real-world human clothing with semantic annotations. In *CVPR*, 2024. 1, 5, 9, 10, 15

[36] Yifan Wang, Jianjun Zhou, Haoyi Zhu, Wenzheng Chang, Yang Zhou, Zizun Li, Junyi Chen, Jiangmiao Pang, Chunhua Shen, and Tong He. π^3: Scalable permutation-equivariant visual geometry learning, 2025. 1, 2, 6, 7, 9, 11, 12, 13, 14, 15

[37] Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point transformer v3: Simpler, faster, stronger. In *CVPR*, 2024. 3, 8

[38] Yao Yao, Zixin Luo, Shiwei Li, Jingyang Zhang, Yufan Ren, Lei Zhou, Tian Fang, and Long Quan. Blendedmvs: A large-scale dataset for generalized multi-view stereo networks. In *CVPR*, 2020. 10

[39] Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. Scannet++: A high-fidelity dataset of 3d indoor scenes. In *CVPR*, 2023. 2, 5, 8, 9, 10, 12

[40] Yuchen Zhang, Nikhil Keetha, Chenwei Lyu, Bhuvan Jhamb, Yutian Chen, Yuheng Qiu, Jay Karhade, Shreyas Jha, Yaoyu Hu, Deva Ramanan, Sebastian Scherer, and Wenshan Wang. Ufm: A simple path towards unified dense correspondence with flow, 2025. 1, 2, 4, 6, 7

[41] Zhe Zhang, Rui Peng, Yuxi Hu, and Ronggang Wang. Geomvsnet: Learning multi-view stereo with geometry perception. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21508–21518, 2023. 10

[42] Yiming Zuo, Willow Yang, Zeyu Ma, and Jia Deng. Omnidc: Highly robust depth completion with multiresolution depth integration. In *ICCV*, 2025. 9