

MUS_t3R: Multi-view Network for Stereo 3D Reconstruction

Supplementary Material

This supplementary material first includes in Appendix A all ablation studies that justify the design choices of the MUS_t3R architecture. We then provide the promised qualitative examples of MUS_t3R on real scenes from various datasets in Appendix B and the snippet of code for the online reconstruction in Appendix C. Finally, we detail in Appendix D the complete quantitative evaluations of our method on the full sets of sequences from TUM-RGBD¹ and ETH3D² datasets [3, 4]. This supplementary is accompanied with a video explaining our contributions and showing real time examples of online 3D reconstruction from MUS_t3R in challenging scenarios, including large loop-closure and dynamic scenes. Note that we plan to release the code to reproduce all these results.

A. Ablation studies

The aim of this section is to provide empirical analysis of our design choices, starting with our simplification of the DUS_t3R architecture in Appendix A.1, and our memory management and loss choices in Appendix A.2. Then in Appendix A.3 we study the scalability of the network to more views than seen during training.

A.1. Simplifying DUS_t3R

As explained in the main paper, the architecture of DUS_t3R cannot easily scale to more views for its asymmetric design would require to train N different decoders for N views, which is not trivial and computationally impractical. For this reason, we propose to leverage a *Siamese* decoder, *i.e.* one decoder for all views, in a symmetric manner. The symmetric decoder is the core of our model as it enables to seamlessly scale to more views without the need to re-train the model. We evaluate here how this choice impacts the performance compared to the original DUS_t3R architecture. In more details, leveraging a symmetric decoder for N views predictions requires to disable the Rotary positional encoding (RoPE) in the Cross-Attention (CA) and to add a learnable encoding to distinguish between the reference view, that defines the origin of the coordinate system, and all the other views. We ablate these in the following.

Setup and metric. To diminish compute and increase speed, we perform ablations in the pairwise setting ($N=2$) on a small subset of datasets namely the union of subsets of Habitat and Megadepth, for a total of $2M$ pairs of 224×224 resolution images. We evaluate performance on the validation sets of these two datasets. We believe this setup to be

¹<https://cvg.cit.tum.de/data/datasets/rgbd-dataset>

²https://www.eth3d.net/slam_datasets

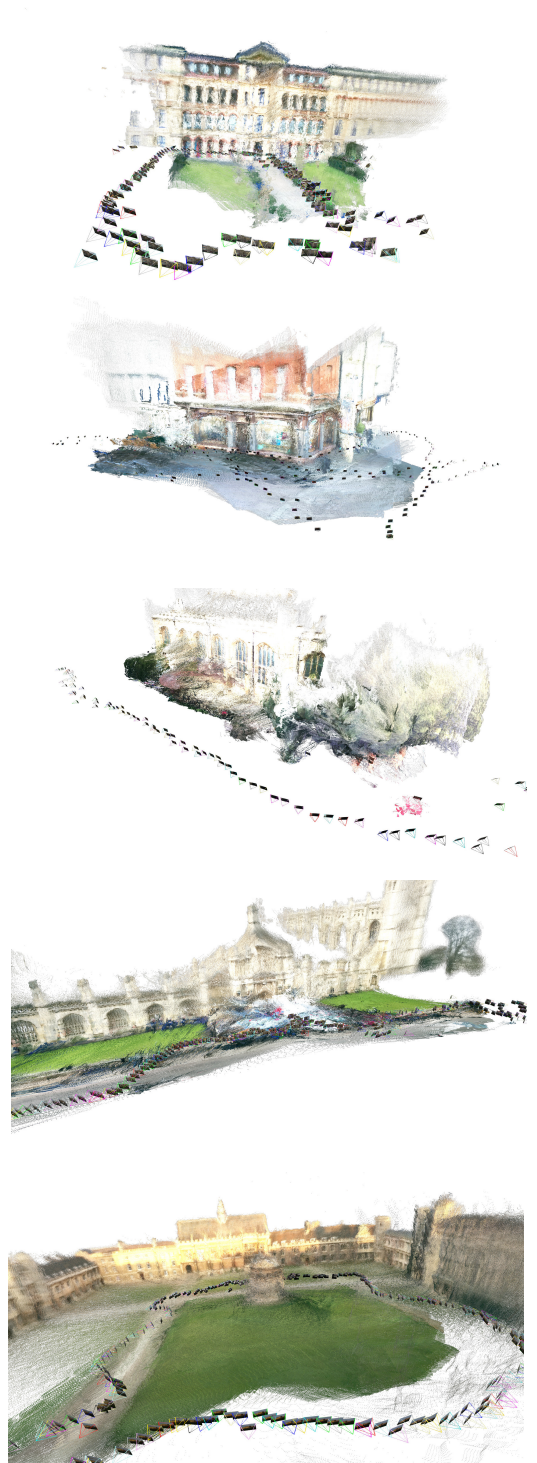


Figure 5. Qualitative example of MUS_t3R reconstructions of Cambridge Landmarks [2].

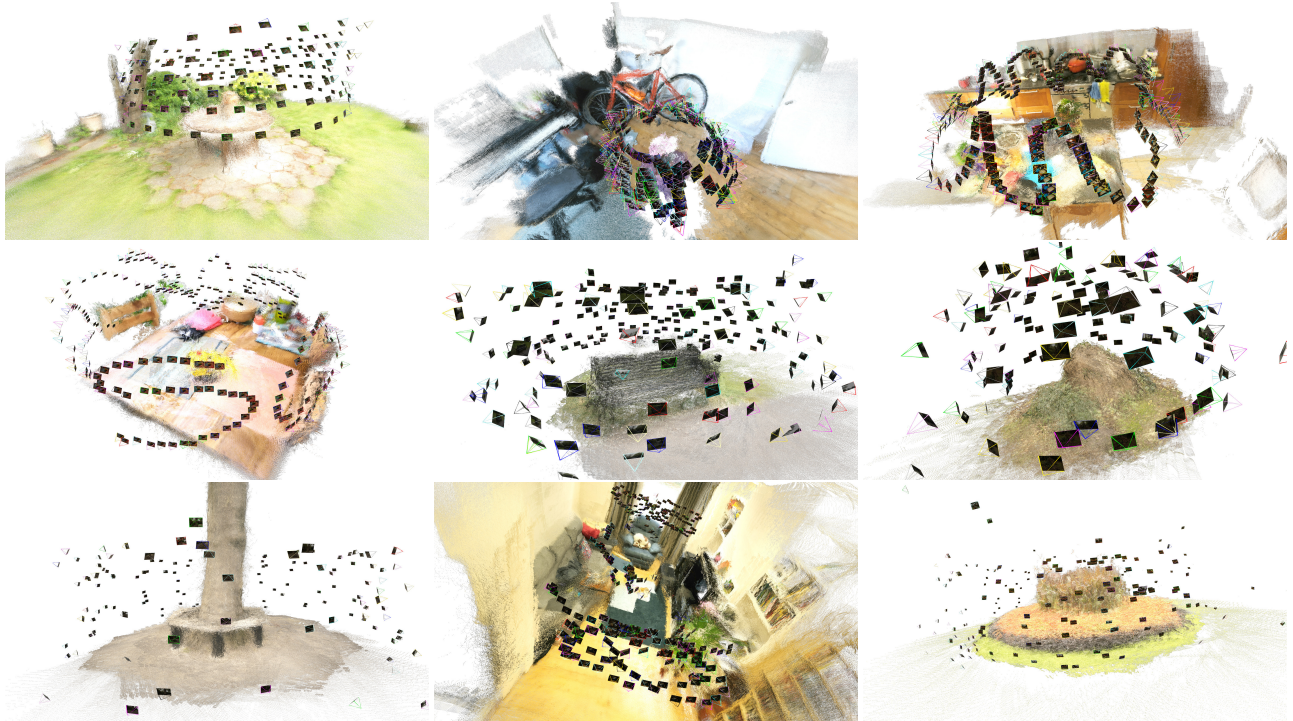


Figure 6. Qualitative example of MUST3R reconstructions of MIP-360 [1].

representative enough for us to reasonably extend the conclusions to the full training set. We report in Tab. 9 the average regression error that directly measures the quality of predicted pointmaps. We observe this metric generally represents well the performance on downstream tasks. In the table, **Baseline** refers to the asymmetric DUST3R architecture. In **No RoPE in CA**, we remove RoPE in the cross attentions. In **Symmetric**, we switch to the symmetric decoder setup where we remove the second decoder and HEAD^{3D} and we use instead the same shared decoder and head for both images. In **Symmetric(embed)**, we add the learned embedding B , while it is absent in **Symmetric(none)**.

Results. We report the results of these ablations in Tab. 9. First, we show that RoPE in the CA block does not matter for performance. We can thus remove it without loss in prediction quality. Then, when using a symmetric decoder without learnable B encoding collapses in **red**. It is rather natural as the network needs to distinguish between the views, justifying the need for the learnable embedding B that achieves a performance similar to that of DUST3R. Overall, it is clear that a symmetric architecture does not impact the performance despite halving the amount of trainable parameters in the decoder.

A.2. MUST3R architecture and loss

We verified in Appendix A.1 that a symmetric architecture does not impact the performance in the binocular case.

	Regression Loss ↓
Baseline	0.193
No RoPE in CA	0.191
Symmetric(none)	0.560
Symmetric(embed)	0.192

Table 9. Analysis of the effects of our proposed simplifications on DUST3R in the pairwise setting.

How to handle N views predictions still remains a challenge however, as directly processing all the views would involve an intractable number of tokens for both training and testing. For this reason, we favor a memory-based approach where only a subset of image tokens is used in memory. Images are thus processed sequentially and we keep the computational complexity low. The following contains the ablations that lead our choices in the design of MUST3R training in the multi-view setting. In more details, we demonstrate the importance of the global 3D feedback from the terminal layer to earlier layers. We also analyze the effect of computing the loss in log space, that becomes crucial for larger scenes.

Setup and metric. For the N views case, we choose to train all variants on 10-tuples and validate on 100-tuples from



Figure 7. Qualitative example of MUST3R reconstructions of ScanNet++ [8] 0d2ee665b and a24f64f7fb (left column) with varying focal lengths. RealEstate10K [11] 000c3ab189999a83 and 00ca5123d8ff6f83 (center column). TUM RGB-D [4] freiburg1 desk and xyz (right column).

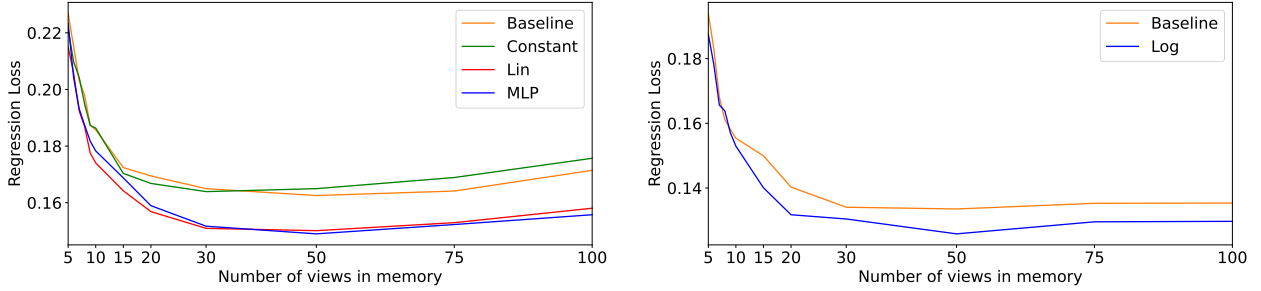


Figure 8. **Median absolute regression error** for ScanNet scenes for 100 views rendered with a varying number of views in memory. (left) Different injection INJ^{3D} architecture choices. (right) Comparison between a log and the default MAST3R loss for metric pointmap regression.

the ScanNet++ [8] dataset, split into a training set of 2M 10-tuples and a validation set of 300 100-tuples, totaling for the latter 30K images from 50 different scenes. We report the metric 3D regression loss with pointmaps expressed in the coordinate frame of the first view, meaning no alignment between the prediction and the *ground-truth* is required. All results are *renderings* $N=100$ images with a varying number of images stored in memory ($5 \leq n \leq N$). We plot in Fig. 8 the median scene regression error over the validation subset.

3D feedback. We ablate in Fig. 8 (left) different variants of Global 3D Feedback from section 3.3 of the main paper. In particular, we demonstrate that the **Baseline** architecture without feedback is far inferior to the **MLP** injection from the tokens of the last memory block presented in the main paper (Sec. 3.3 and Fig. 4). As a reminder, this approach injects the terminal memory state $L-1$ to all others with a LayerNorm followed by a 2 layer MLP, with a fourfold increase in the hidden dimensions. Interestingly, **Lin** injection with a simple linear layer instead of an MLP performs

almost as well except when scaling the memory size. Since we expect training on all datasets to be a harder task, we favor a slightly more elaborate injection mechanism. We also verify that a simple learnable parameter **Constant** is not sufficient and actually degrades the performance, demonstrating that our feedback layer is effectively fetching information from the terminal memory layer and not simply adding a constant bias to the memory tokens. Importantly, a natural choice would be to inject global information from the final layer L of the network, *i.e.* the one right before the prediction head. We find that this method leads to a significantly degraded performance. We hypothesize that using L might be too constrained by HEAD^{3D} , and therefore it is better to add feedback from layer $L-1$, the terminal memory layer.

Computing the loss in log space. To analyze the choice of log space loss in section 4 of the main paper, we compare two MUST3R models trained following the recipe of Sec. 4 of the main paper.

The **Log** model is trained with the log space loss (Eq.

Methods			7-Scenes						NRGBD						DTU						FPS	Mem
			Acc ↓		Comp ↓		NC ↑		Acc ↓		Comp ↓		NC ↑		Acc ↓		Comp ↓		NC ↑			
<i>n</i>	<i>s</i>		Mean	Med.	Mean	Med.	Mean	Med.	Mean	Med.	Mean	Med.	Mean	Med.	Mean	Med.	Mean	Med.	Mean	Med.	Mean	Med.
F-Recon [7]			0.124	0.076	0.055	0.023	0.619	0.688	0.285	0.206	0.151	0.063	0.655	0.758	-	-	-	-	-	-	(≤ 1)	
DUST3R-224 [6]			0.029	0.012	0.028	0.009	0.668	0.768	0.054	0.025	0.032	0.010	0.802	0.953	2.296	1.297	2.158	1.002	0.747	0.848	0.74 (0.78)	38.1G
Spann3R [5]			0.034	0.015	0.024	0.009	0.664	0.763	0.069	0.032	0.029	0.011	0.778	0.937	4.785	2.268	2.743	1.295	0.721	0.823	27.38 (65.49)	5.0G
MUST3R-224 10 1			0.037	0.017	0.026	0.010	0.654	0.741	0.064	0.028	0.032	0.011	0.778	0.922	3.256	1.863	2.193	0.995	0.715	0.815	86.31	2.5G
MUST3R-224 20 1			0.030	0.013	0.026	0.010	0.662	0.753	0.061	0.026	0.029	0.011	0.786	0.928	3.256	1.863	2.193	0.995	0.715	0.815	59.51	2.9G
MUST3R-224 all 1			0.028	0.012	0.027	0.010	0.665	0.758	0.062	0.025	0.031	0.012	0.788	0.930	3.256	1.863	2.193	0.995	0.715	0.815	40.41	4.1G
MUST3R-224 all 5			0.035	0.015	0.027	0.011	0.662	0.752	0.059	0.025	0.029	0.011	0.792	0.934	3.261	1.855	2.157	1.016	0.715	0.815	68.83	4.1G
MUST3R-224 all 10			0.037	0.016	0.028	0.011	0.661	0.751	0.057	0.025	0.028	0.011	0.792	0.934	3.269	1.805	2.147	0.978	0.715	0.816	72.59	4.8G
MUST3R-512 10 1			0.028	0.009	0.025	0.007	0.617	0.682	0.052	0.023	0.019	0.008	0.764	0.907	3.261	1.681	1.965	0.765	0.661	0.741	25.01	4.3G
MUST3R-512 20 1			0.025	0.009	0.026	0.008	0.617	0.683	0.048	0.022	0.019	0.008	0.769	0.911	3.261	1.681	1.965	0.765	0.661	0.741	17.37	5.3G
MUST3R-512 all 1			0.026	0.009	0.027	0.009	0.617	0.682	0.048	0.022	0.020	0.008	0.768	0.911	3.261	1.681	1.965	0.765	0.661	0.741	12.10	8.1G
MUST3R-512 all 5			0.036	0.014	0.025	0.008	0.616	0.680	0.048	0.021	0.019	0.008	0.770	0.912	3.353	1.741	1.982	0.772	0.663	0.742	13.52	9.9G
MUST3R-512 all 10			0.042	0.016	0.024	0.008	0.615	0.679	0.047	0.021	0.019	0.007	0.769	0.910	3.493	1.888	2.046	0.809	0.664	0.745	12.79	10.5G

Table 10. **Comparison with Spann3R.** We evaluate MUST3R for different maximum size of memory (n) and different number of images at once when updating the memory (s). For $s=1$, we initialize with two images to match the training configuration. FPS numbers in parenthesis are from [5]. For DUST3R, our FPS and GPU memory numbers were obtained with the 224 linear model and a complete graph.

8 in the main paper) while **Baseline** is trained with the default metric regression loss from MAST3R. Again, we show the results for varying number of images used in the memory in Fig. 8 (Right). We can notice a clear gap between the two losses when going above 10 views in memory, the model being trained with the loss in log space providing a significantly better scaling to more views. For instance with $n = 50$ views in memory the baseline achieves 13.4cm accuracy *vs.* 12.6 for the log space loss. An interpretation is that the log space loss allows the network to regress larger values which is crucial for more input views as they will most often observe larger parts of the scenes.

A.3. More views at inference

Beyond architectural choices, an important result of the previous section is the scalability of MUST3R to more views than ever seen during training, as explained in section 3.2 of the main paper. In particular, we can use our architecture to robustly predict from $n \gg 10$ views in memory despite being trained only with $n \leq 10$ as seen in Fig. 8. Interestingly, it can also process $s \gg 1$ views at the same time while having been trained to predict $s=1$ views at the same time.

In this section, we study the impact of the number of views in the memory (n) and the number of views used at once (s) when updating the memory in MUST3R. These comparisons can be seen on the one hand in Tab. 10 where we complement Tab. 6 of the main paper with experiments on MUST3R varying n and s during inference (see columns 2 and 3).

Number of views in memory. The plots in Fig. 8 suggest that the model can successfully leverage up to $n = 50$ views in the memory, despite being only trained with at most 10

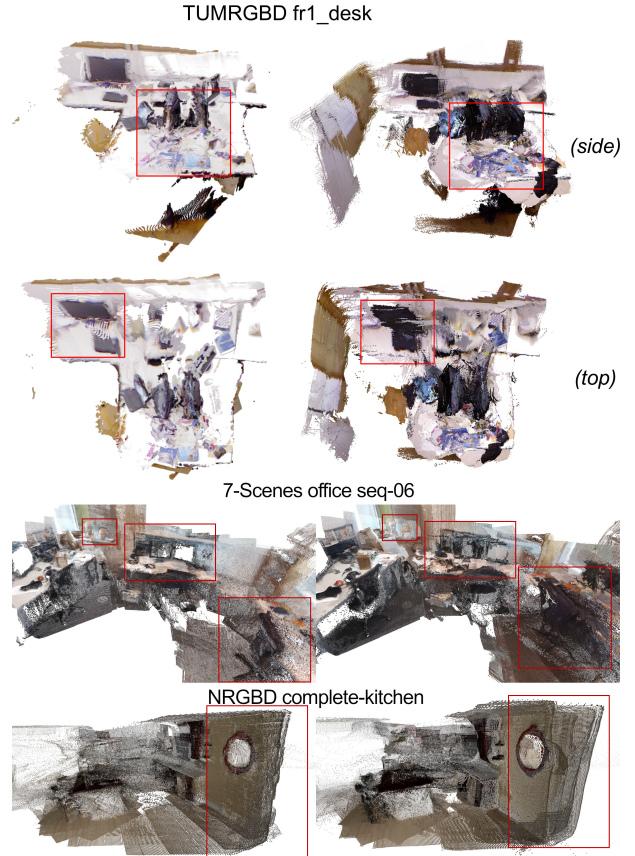


Figure 9. Qualitative comparison between MUST3R (left) and Spann3R (right) on sequences from the TUM RGBD, 7-Scenes and NRGBD datasets.

	f1										f2										f3										Avg																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																					
	floor	360hoph	360kid	coke	desk.p	dishes	flow.bqt	flow.bqr	met.sph	met.sph2	pion_360	pion_slm	pion_slm2	pion_slm3	rry	cab	icab	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far		ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near	ns_at_far	ns_at_near

Table 11. **Detailed results on TUM RGBD [4]** (34 sequences not included in Tab. 1, and summarized in Tab. 2 of the main text). One Dense (D) versus dense unconstrained (U) methods on TUM-RGBD SLAM benchmark. (*) GIORIE-SLAM [10] was re-run without Loop Closure and global Bundle Adjustment.

	casht	casht2	cei1	dc3	dech1	eiffl1	eifflc1	eille2	eiglc3	mq1	mq4	mqfa1	mqfa2	mqhe	mol	plr2	plt3	plt4	plt5	pltsc1	ref1	rep	sfmbe	sfmgr	sfmhl	sfmrl1	sf2	sf3	sfsh	vili1	vili2	Avg
RMSE ATE Tracking Error [m] ↓																																
Spann3R[5]	0.05	0.08	2.3	0.96	1.25	1.41	0.68	0.12	0.76	0.73	1.09	0.5	0.03	0.25	0.65	0.38	0.18	0.05	0.25	0.95	0.39	0.91	0.5	5.93	7.72	1.36	0.1	0.14	0.1	0.84	0.18	0.99
MUST3R-224-C	0.06	0.05	2.01	0.89	0.81	1.32	0.73	0.09	0.93	0.79	0.6	0.36	0.07	0.19	0.18	0.07	0.28	0.03	0.24	0.08	0.39	0.76	0.14	4.32	6.12	0.72	0.36	0.35	0.1	0.63	0.26	0.77
MUST3R-224	0.04	0.03	1.98	1.02	0.85	1.29	0.59	0.08	0.64	0.32	0.33	0.42	0.04	0.16	0.26	0.07	0.12	0.06	0.2	0.1	0.39	0.31	0.11	5.96	5.0	0.72	0.05	0.08	0.09	0.28	0.14	0.70
Vertical FoV Error (in degrees) ↓																																
Spann3R [5]	21.26	17.24	27.83	26.53	26.16	28.28	25.55	26.28	25.78	28.63	30.33	67.47	26.86	27.41	25.92	28.64	26.01	27.0	25.96	26.5	25.51	25.61	16.62	26.09	26.94	26.61	25.51	25.37	25.62	67.27	26.95	28.50
MUST3R-224-C	3.68	1.04	5.89	2.01	1.35	13.67	2.56	1.82	1.93	5.18	6.28	9.25	5.51	7.79	1.75	10.39	14.07	11.47	12.37	1.12	3.56	0.03	2.04	4.84	4.28	1.42	1.65	2.8	2.43	11.52	13.92	5.40
MUST3R-224	0.63	1.01	1.4	1.0	1.39	10.97	1.17	1.16	1.07	3.27	2.12	12.15	6.84	3.76	0.81	7.46	4.28	8.04	6.56	1.0	3.1	1.6	0.33	0.72	2.63	1.25	0.94	0.11	2.66	4.21	4.58	3.17
Scale Error (ratio of GT scale) ↓																																
Spann3R [5]	1.03	0.79	1.17	0.4	1.51	2.33	4.87	5.44	4.02	1.8	2.9	0.64	1.67	0.86	4.22	0.25	1.14	0.16	0.05	3.31	0.63	0.3	3.42	8.06	13.27	7.2	3.62	4.66	1.85	3.05	2.11	2.79
MUST3R-224-C	0.49	0.32	0.28	0.59	0.32	0.67	0.25	0.19	0.27	0.48	0.26	0.63	0.53	0.59	0.13	0.65	0.01	0.62	0.54	0.29	0.57	0.4	0.19	0.49	0.04	0.02	0.13	0.06	0.3	0.21	0.65	0.36
MUST3R-224	0.32	0.4	0.49	0.77	0.41	0.61	0.05	0.12	0.04	0.67	0.41	0.01	0.74	0.72	0.05	0.48	0.43	0.82	0.62	0.11	0.67	0.05	0.19	0.09	0.37	0.01	0.11	0.06	0.21	0.34	0.61	0.35

Table 12. **Detailed results on ETH3D SLAM [3]** (32 sequences not included in Table 5 of the main paper). RMSE ATE, Vertical FoV and Scale errors for three dense unconstrained (U) methods.

views. Beyond 50 views, the results do not improve further. The results in Tab. 10 reinforce this observation. MUST3R-224 $n = 10, s = 1$ is worse than $n = all, s = 1$. We note that in the Spann3R evaluation protocol, there are only 10 images used for DTU sequences, from 10 to 42 images for NRGBD and from 25 to 50 for 7-Scenes explaining the limited difference between $n = 20, s = 1$ and $n = all, s = 1$.

Number of views at once. We also study how MUST3R behaves when using multiple views to update the memory at the same time. Note that in most experiments we only use one image at a time, except for initialization where we start with $s=2$ images. We run experiments with $s = 5$ and $s = 10$ views to update the memory. Results on DTU, NRGBD and 7-Scenes are shown in Tab. 10 for both MUST3R-224 and MUST3R-512. We observe that predicting multiple images at the same time does not significantly degrade the performance. Interestingly, it yields a decent increase in FPS for MUST3R-224: (72.59 for $s = 10$ vs. 40.41 for $s = 1$), at the cost of a slightly increased memory usage (4.8G for $s = 10$ vs. 4.1G for $s = 1$). For MUST3R-512, the FPS difference is not significant and does not really justify the increased memory cost.

B. Qualitative examples

As promised in Sections 1 and 5.1 of the main paper, we provide here additional qualitative results on various datasets. Image poses are predicted via Procrustes analysis, and for visualization purposes, we only show highly confident points. We wish to demonstrate the ease of use and plug-and-play nature of MUST3R in varied scenarios, and its robustness indoors Fig. 7, outdoors Fig. 5 or object-centric Fig. 6. We emphasize that our approach does not assume a single set of intrinsics for an image collection and seamlessly works with heterogeneous sensors, as shown in the ScanNet++ scenes in Fig. 7 (left).

Video Material. We provide a supplementary video containing examples of uncalibrated online reconstructions on the TUM-RGBD dataset. It is worth noting that the visualizations are the raw predictions by the network: global pointmaps and camera poses obtained via Procrustes. We denote in green the chosen memory frames, and only show the pointmaps of those, plus the current prediction with associated camera frustum in blue. The only post-processing is a simple filtering of the pointmaps with a threshold on the confidence (bottom right in the video) for better visibility. This threshold explains why dynamic objects and humans do not appear in the visualization: the network consistently predicts low confidence values for these and are thus not visible in the interface.

```

perc=85
thresh=0.05
def online_update(frame,munst3r,memory,scene3d):
    # forward view with current memory
    Xil, Xii, new_tokens=munst3r(frame,memory)
    # depth and focal from local prediction
    depth = Xii[...,-1] # [H,W]
    focal = get_focal(depth)
    # pose via procrustes between local and global
    pose = procrustes_align(Xii,Xil)
    # viewing direction for each pixel
    rays = normalize(Xil-pose.t)
    # viewing-direction aware 3D discovery rate
    dists = scene3d.query(Xil,rays)
    if percentile(dists/depth,perc) > thresh:
        # Append frame tokens to memory
        memory.append(new_tokens)
        # Add predicted points and directions
        scene3d.add(Xil, rays)
    return Xil,focal,pose,depth

```

Figure 10. Python code for Uncalibrated Visual Odometry.

Qualitative comparisons. We also provide in Fig. 9 qualitative comparisons between MUST3R and Spann3R.

C. Memory Management

We provide in Fig. 10 the pseudo-code for the online algorithm described in Sec. 3.4 of the main paper. We believe MUST3R to be an important step towards a simplified VO and even RGB-SLAM pipeline. In fact, the network is internally managing all the traditionally heavily engineered steps such as keypoint selection and matching, pose estimation and 3D triangulation. Our results show that a simplistic memory frame selection mechanism is enough to outperform traditional VO methods.

D. Full Visual Odometry results

In this section we provide the detailed tables of evaluating online MUST3R model on TUM RGB-D [4] and ETH3D [3].

D.1. Full TUM RGB-D dataset

Tables 1, 3, 4 in the main paper reported RMSE APE, vertical FoV and scale errors for 11 sequences of TUM-RGBD dataset [4]; these sequences were selected for being most frequently evaluated by the recent state-of-the-art methods. Here, Tab. 11 completes our evaluation and presents detailed results of MUST3R-224 on all for 34 remaining TUM-RGBD sequences (see the main paper and Table 2 for the results aggregated over 5 categories of sequences). The table also includes comparison to GIORIE-SLAM [10] run in VO mode. The full dataset contains extremely hard sequences and, to the best of our knowledge, it has never been fully tested previously by other VO works. Despite its simplicity, MUST3R performs overall on par with GIORIE-VO

while being substantially faster. Furthermore, MUST3R outperforms Spann3R on all sequences.

D.2. ETH3D dataset

Finally, Tab. 12 presents detailed results of MUST3R-224 on 32 ETH3D sequences which complete the evaluation of the 8 sequences included in the main paper (Tab. 5). The table reports RMSE APE, vertical FoV and scale errors for all 32 sequences and compares MUST3R and MUST3R-C to Spann3R, both in 224 resolution. As the table clearly shows, both versions of MUST3R again outperform Spann3R on the vast majority of sequences, although we note that there is still room for improvements on some sequences, usually when the scene size becomes too large. We hope these systematic results, along with code, will help foster research in this direction.

References

- [1] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. In *CVPR*, 2022. 2
- [2] Alex Kendall, Matthew Grimes, and Roberto Cipolla. PoseNet: a Convolutional Network for Real-Time 6-DOF Camera Relocalization. In *ICCV*, 2015. 1
- [3] Thomas Schöps, Torsten Sattler, and Marc Pollefeys. BAD SLAM: Bundle Adjusted Direct RGB-D SLAM. In *CVPR*, 2019. 1, 5, 6
- [4] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *IROS*, 2012. 1, 3, 5, 6
- [5] Hengyi Wang and Lourdes Agapito. 3D Reconstruction with Spatial Memory. arXiv:2408.16061, 2024. 4, 5
- [6] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jérôme Revaud. DUST3R: Geometric 3D Vision Made Easy. In *CVPR*, 2024. 4
- [7] Guangkai Xu, Wei Yin, Hao Chen, Chunhua Shen, Kai Cheng, and Feng Zhao. FrozenRecon: Pose-free 3D Scene Reconstruction with Frozen Depth Models. In *ICCV*, 2023. 4
- [8] Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. ScanNet++: A High-Fidelity Dataset of 3D Indoor Scenes. In *ICCV*, 2023. 3
- [9] Ganlin Zhang, Erik Sandström, Youmin Zhang, Manthan Patel, Luc Van Gool, and Martin R. Oswald. GIORIE-SLAM: Globally Optimized RGB-only Implicit Encoding Point Cloud SLAM. arXiv:2403.19549, 2024. 5
- [10] Ganlin Zhang, Erik Sandström, Youmin Zhang, Manthan Patel, Luc Van Gool, and Martin R. Oswald. GIORIE-SLAM: Globally Optimized RGB-only Implicit Encoding Point Cloud SLAM. arXiv:2403.19549, 2024. 5, 6
- [11] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavel. Stereo Magnification: Learning View Synthesis using Multiplane Images. *ACM Transactions on Graphics*, 37(4):1–12, 2018. 3