

Appendices

A Implement details

We use PyTorch for all implementations. For the visual prompts, the prompt length is 32 for each transformer layer. For constructing triplets, we consider all available images. Considering a scene with n images, the total number of triplets is n^3 . For computational efficiency, if the total number of triplets exceeds 165, we randomly sample 165 triplets as the test-time training set. For test-time training, we adopt the Adam [67] optimizer. Given the varying number of available views in each dataset, we select different learning rates accordingly. We set the learning rate to 0.00001, 0.00008, 0.00004, and 0.00001 for 7Scenes [65], NRGBD [66], DTU [57], and ETH3D [58], respectively. We only fine-tune Test3R for 1 epoch at the specific test scene.

B Consumption

This section presents the computational overhead of test-time training, including both test-time training time and GPU memory usage, to emphasize the **nearly cost-free** nature of our approach.

Test Time Training Consumption. Given our previously described triplet construction strategy in appendix A, the maximum training duration remains constant. Test3R introduces only 14 seconds of extra time overhead, which we consider negligible during the entire process. In fact, the majority of Test3R’s time consumption comes from the pair-wise prediction and global alignment introduced by DUST3R [12]. This proportion increases as the number of images grows. For example, for a set of 50 images on the 7Scenes dataset, pair-wise prediction takes nearly 250s, and global alignment consumes nearly 70 seconds.

Memory Allocation. We reports the parameter footprint, and memory allocation of Test3R on the scene *Office-seq-09* from the 7Scenes [65]. The result is shown in Table 5. Compared to the vanilla DUST3R, only prompts are introduced in each transformer layer, resulting in negligible overhead in terms of both parameter footprint and memory consumption for Test3R. By fine-tuning these additional parameters, our model can effectively enhance the final reconstruction quality.

Method	Parameter		Memory	
	Promptsts	Total	Promptsts	Total
DUST3R	-	571.17M	-	2178.85M
Test3R	0.79M	571.96M	3M	2181.85M

Table 5: **Comparison of model parameter count and memory consumption.**

C Discussion and Analysis

C.1 Cross-pair Consistency

Our self-supervised training objective is designed to maximize cross-pair consistency of pointmap. In this section, we demonstrate the effectiveness of this objective.

Specifically, we visualize the pointmaps of the same reference view but paired with different source views. The pointmaps are visualized in 2D space by projecting them onto the corresponding depthmaps, which provides a clearer representation while avoiding interference caused by viewpoint variations. The relationship between the pointmap and the depthmap is defined by the following equation:

$$X_{i,j} = K^{-1}[iD_{i,j}, jD_{i,j}, D_{i,j}] \quad (9)$$

where $(i, j) \in \{1 \dots W\} \times \{1 \dots H\}$ is the pixel coordinates and $K \in \mathbb{R}^{3 \times 3}$ is the camera intrinsics. $X_{i,j}$ and $D_{i,j}$ are the corresponding pointmap and depthmap.

We visualize the depthmap on the *scan1* from the DTU [57] dataset, as shown in Figure 6. Compared to vanilla DUST3R, Test3R demonstrates superior consistency across different pairs. The depthmaps predicted by DUST3R exhibit significant inconsistencies in regions with limited overlap. After optimizing by cross-pairs consistency objective, Test3R generates consistent and reliable depth predictions in these regions. Moreover, even in the (I^{ref}, I^{ref}) pair case, Test3R can still predict relatively consistent depth maps.

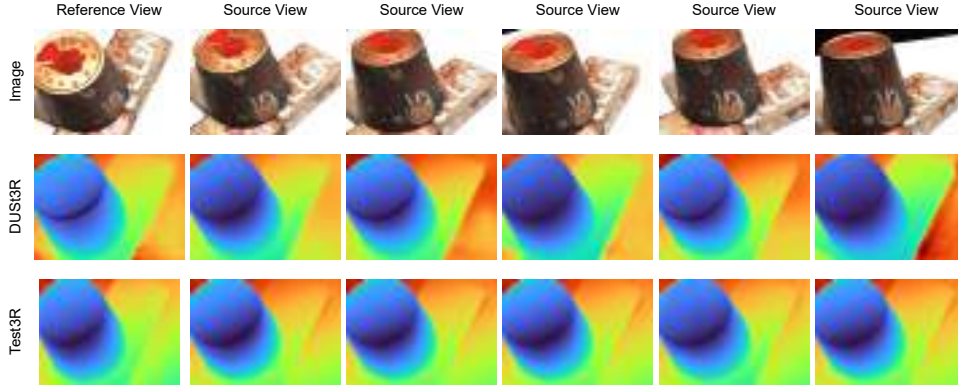


Figure 6: **Comparison on cross-pair consistency.** The depth map of the same reference view but paired with different source views. Test3R demonstrates superior cross-pair consistency compared to vanilla DUST3R.

C.2 Compared to single forward-based model

We compare DUST3R and Test3R with the current single forward-based models, Fast3R [37] and VGGT [17]. These models can process multiple images in a single forward pass, which may enhance the model’s robustness and accuracy. Therefore, we report the result on the NRGBD [66] dataset, as shown in Table 6. It demonstrates that these models still struggle to generalize to unseen scenes. Fast3R shows significantly inferior reconstruction quality compared to DUST3R and Test3R on the NRGBD dataset. Meanwhile, although VGGT achieves relatively strong performance on this dataset, it requires substantial computational resources for training and still underperforms Test3R on several metrics. These results validate the effectiveness and robustness of our model across diverse scenes. By maximizing cross-pair consistency, our model can adapt to previously unseen scenes, thereby enabling more accurate reconstruction of challenging scenes with minimal test-time training overhead and parameter footprint.

Method	Acc↓		Comp↓		NC↑	
	Mean	Med.	Mean	Med.	Mean	Med.
Fast3R [37]	0.377	0.215	0.254	0.152	0.695	0.804
VGGT [17]	0.076	0.043	0.084	0.045	0.901	0.980
DUST3R [12]	0.144	0.019	0.154	0.018	0.871	0.982
Ours	0.083	0.021	0.079	0.019	0.870	0.983

Table 6: **Comparison with Single forward-based model.**

C.3 Implementation on Multi-view methods.

In this section, we demonstrate that our method can also be applied to multi-view models. We further evaluated the results of implementing our method on VGGT [17] across all 18 scenes from the 7Scenes dataset. For detailed implementation, we specifically insert the prompts into the image tokens of each input image, in a manner similar to VGGT’s handling of the register token. During test-time training, it still takes a triplet as input to predict two pointmaps under the same reference

Method	Acc ↓	Comp ↓	NC ↑
VGGT	0.084	0.088	0.785
VGGT+Ours	0.051	0.066	0.760

Table 7: **Performance comparison on the multi-view model.**

view but paired with different source views as introduced in Section 4.2. To validate the effectiveness of the model, we follow the default experimental settings and construct the triplets as described in appendix A. The final results are presented in the Table 7. We outperform vanilla VGGT [17] on both the Acc and Comp metrics, with only the NC metric showing comparatively lower performance. This demonstrates that our method is also applicable to multi-view input approaches such as VGGT.

C.4 PEFT evaluation.

Method	Acc ↓	Comp ↓	NC ↑
DUSt3R	0.146	0.181	0.736
LoRA	0.134	0.139	0.730
Fine-tune	0.137	0.139	0.722
Ours	0.105	0.136	0.746

Table 8: **Comparison of different optimization methods.**

Our method employs visual prompts [15] as optimization carriers for adapting the model. In this section, we further explore alternative optimization paradigms, including full fine-tuning and methods based on Low-Rank Adaptation (LoRA) [68].

We report averaged results across all 18 scenes on the 7Scenes dataset. All experimental settings remain consistent with our default setting. The average scores of all scenes are presented in the Table 8. As shown, leveraging our training objective to maximize cross-pair consistency enables all three fine-tuning methods to achieve improvements in the final reconstruction results. The LoRA-based method, direct fine-tuning, and our approach resulted in decreases in Acc. of 0.012, 0.009, and 0.04, respectively. The superior performance of our method highlights the efficacy of prompt tuning compared to other optimization strategies.

C.5 Camera pose evaluation.

Method	ATE ↓	RPE_{trans} ↓	RPE_{rot} ↓
DUSt3R	0.306	0.176	9.476
Test3R	0.171	0.088	1.864

Table 9: **Evaluation of camera pose estimation.**

Following the experiment setting on MonST3R [16], we report the results across 14 scenes on the Sintel [69] dataset. The average score of all scenes is reported in the Table 9. Compared to DUSt3R, Test3R demonstrates a substantial improvement across all evaluation metrics even without incorporating any dynamic scenes priors.

D Limitations

While Test3R significantly improves the quality of the reconstruction on the DUSt3R, there are still some limitations. Firstly, the final reconstruction quality still heavily depends on the input images. It still struggles with in-the-wild data, which is often characterized by occlusions, dynamic objects, and varying illumination. Secondly, Test3R lacks efficient utilization of inference results. It only considers the pointmaps from the reference views, without leveraging the pointmaps from the source views. Many current baselines incorporate a camera head into the prediction stage. Therefore, using

camera poses to align different viewpoints is a promising direction for future research. Thirdly, we focus on scenarios with sparse viewpoints in our study, where Test3R can consider each view. However, when the number of viewpoints increases, considering every viewpoint is computationally expensive. Therefore, how to effectively sample these views when forming triplets remains an open question.

E More reconstruction result

We provide more reconstruction results, as shown in Figure 7. We observe that Test3R achieves more detailed and consistent reconstructions than DUS3R, as specifically illustrated within the red boxes. The objects, like fences and stone pillars, remain consistent under different viewpoints, demonstrating improved cross-view consistency. Furthermore, Test3R produces fewer outliers in ambiguous or low-texture regions, such as the distant trees and sky, highlighting its robustness.

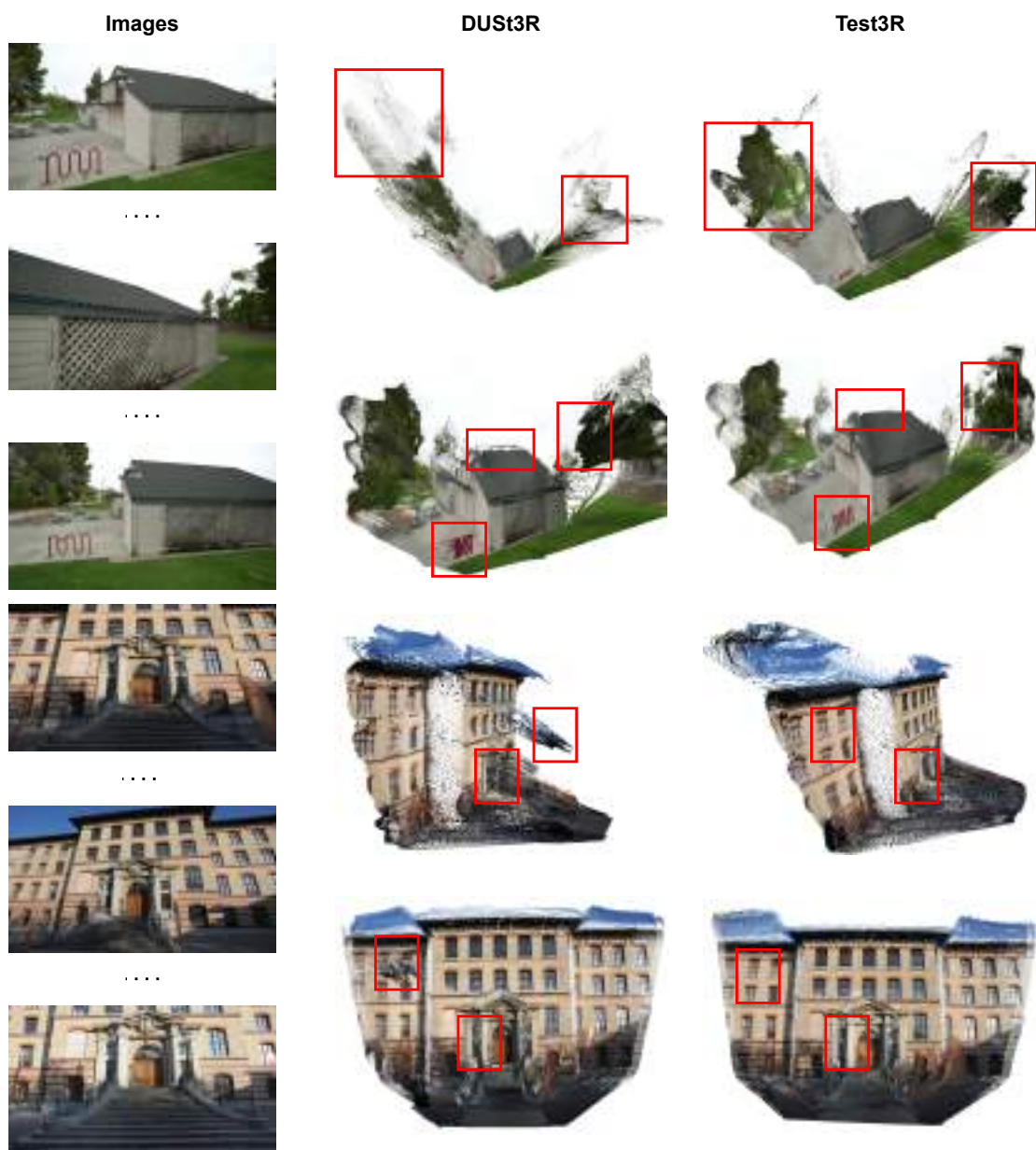


Figure 7: Qualitative comparisons of DUST3R and our method.