ZipMap: Linear-Time Stateful 3D Reconstruction via Test-Time Training

Supplementary Material

Outline

In this Supplementary Material, we provide the following:

- **Appendix A: Evaluation Details.** Comprehensive details on baseline evaluation, runtime evaluation, long-sequence evaluation protocols.
- **Appendix B: Training Details.** Full descriptions of the training datasets, the complete training loss function, and additional implementation details for finetuning the model for scene state query and streaming reconstruction.
- **Appendix C: More Results for the Implicit Scene State.** Visualizations demonstrating the ability to query the learned implicit scene state (Figure 7).
- **Appendix D: More Evaluation Results.** Additional quantitative and qualitative results, including monocular depth estimation benchmarks (Table 8), general qualitative comparisons (Figure 6), effects of removing the reference view (Figure 8, Tables 10, 11, 12), and more long-sequence evaluation results (Figure 9, Figure 10).
- **Appendix E: Limitations.** A discussion of the current limitations of our proposed method.

A. Evaluation Details

A.1. Baseline Evaluation Details

To produce the results in Section 4 of the main paper, we evaluated CUT3R [71], TTT3R [13], VGGT [68], π^3 [76], and our method directly. For all other baselines, we use the results reported in the π^3 paper. We resize input images according to patch size: for CUT3R and TTT3R (patch size 16), we set the image width to 512, while for VGGT, π^3, and our method (patch size 14), we set the image width to 518.

A.2. Runtime Evaluation Details

To produce the runtime analysis shown in Figure 1 of the main paper, we evaluate all methods on a single H100 SXM5 GPU using PyTorch 2.7.1 and CUDA 12.8. All implementations use the pytorch `scaled_dot_product_attention` function for softmax attention, which is a cuDNN implementation of FlashAttention-2 [15]. Input resolutions follow the same aspect ratio as the ScanNet-v2 [14] dataset used to evaluate the pose estimation error in Figure 1: 392×518 for VGGT, π^3, and our method (patch size 14), and 384×512 for CUT3R and TTT3R (patch size 16). The original VGGT implementation ran out of GPU memory on long input sequences (e.g., 750 frames) because it caches features from all layers, even though the DPT heads only use four of them. To enable long-sequence evaluation, we optimized VGGT's implementation to store only the features required by the DPT heads, which eliminates these out-of-memory issues without affecting accuracy or runtime. We evaluate sequences up to 750 frames, as this is already close to the memory limit (80GB) of both our method and the baseline [68]. The reported runtime is averaged over 10 iterations, after 2 warm-up iterations. See Table 7 for detailed runtimes.

As we see in Table 7, when the input views are very sparse (e.g., 5 frames), all methods are able to complete reconstruction quickly. Our method is slightly slower than the quadratic methods (VGGT and π^3), likely because (i) our method implements the test-time training block using standard PyTorch code, whereas the quadratic baselines rely on highly optimized fused FlashAttention kernels, and (ii) our method applies Newton–Schulz orthonormalization during the forward pass (Equation 4 in the main paper), which incurs a constant additional cost. As the number of input frames increases, our method exhibits a clear speed advantage. At 750 frames, our method finishes reconstruction in under 10 seconds (75 FPS) , which is more than $20\times$ faster than VGGT and $15\times$ faster than π^3.

When querying the implicit scene state, we only do the apply operation to the TTT blocks without performing any update step. As a result, querying is faster than reconstruction, reaching about 100 FPS in our experiments.

A.3. Long-Sequence Evaluation Details

We follow the evaluation protocol in π^3 [76] and take the first N frames of each test sequence of ScanNet-v2 [14] dataset for evaluating camera pose estimation and video depth estimation. We evaluated up to $N = 750$ frames on ScanNet-v2. We also follow the evaluation protocol in π^3 for evaluating 3D point estimation on 7-Scenes [58] dataset, by either taking the first N frames or uniformly subsampling N frames. Due to the slow ICP alignment process when computing the chamfer distance, we only evaluated up to $N = 300$ frames under time constraint. For the evaluation of camera pose estimation on DL3DV [37] (55 scenes in test split), we additionally exclude 5% of scenes with the largest errors when calculating the metrics for each method to mitigate the impact of outliers. We only evaluated up to $N = 300$ frames since most of DL3DV scenes have no more than 400 frames.

B. Training Details

B.1. Full Training Datasets

We train our model on a diverse collection of 29 publicly available datasets. We use 23 static scene datasets, includ-

Table 7. **Runtime evaluation.** Inference time (in seconds) as a function of the number of input images N. VGGT and π^3 scale quadratically with N, resulting in slow speeds when N is large. CUT3R, TTT3R, and our method scale linearly with N, with ours being the fastest for dense input frames.

Model	**Complexity**	**Params**	**Time to Reconstruct N Frames (seconds)**									
			$N = 5$	10	25	50	100	200	300	400	500	750
VGGT [68]	$\mathcal{O}(N^2)$	1.26B	0.102	0.194	0.569	1.524	4.689	16.040	34.022	58.842	90.389	200.364
π^3 [76]	$\mathcal{O}(N^2)$	959M	0.087	0.157	0.450	1.186	3.604	12.190	25.765	44.464	68.255	151.159
CUT3R [71]	$\mathcal{O}(N)$	793M	0.206	0.413	1.018	2.056	4.088	8.222	12.430	16.618	21.025	31.246
TTT3R [13]	$\mathcal{O}(N)$	793M	0.206	0.411	1.033	2.036	4.128	8.267	12.435	16.511	20.767	31.197
Ours	$\mathcal{O}(N)$	1.40B	0.125	0.183	0.383	0.712	1.362	2.681	4.017	5.348	6.671	9.999

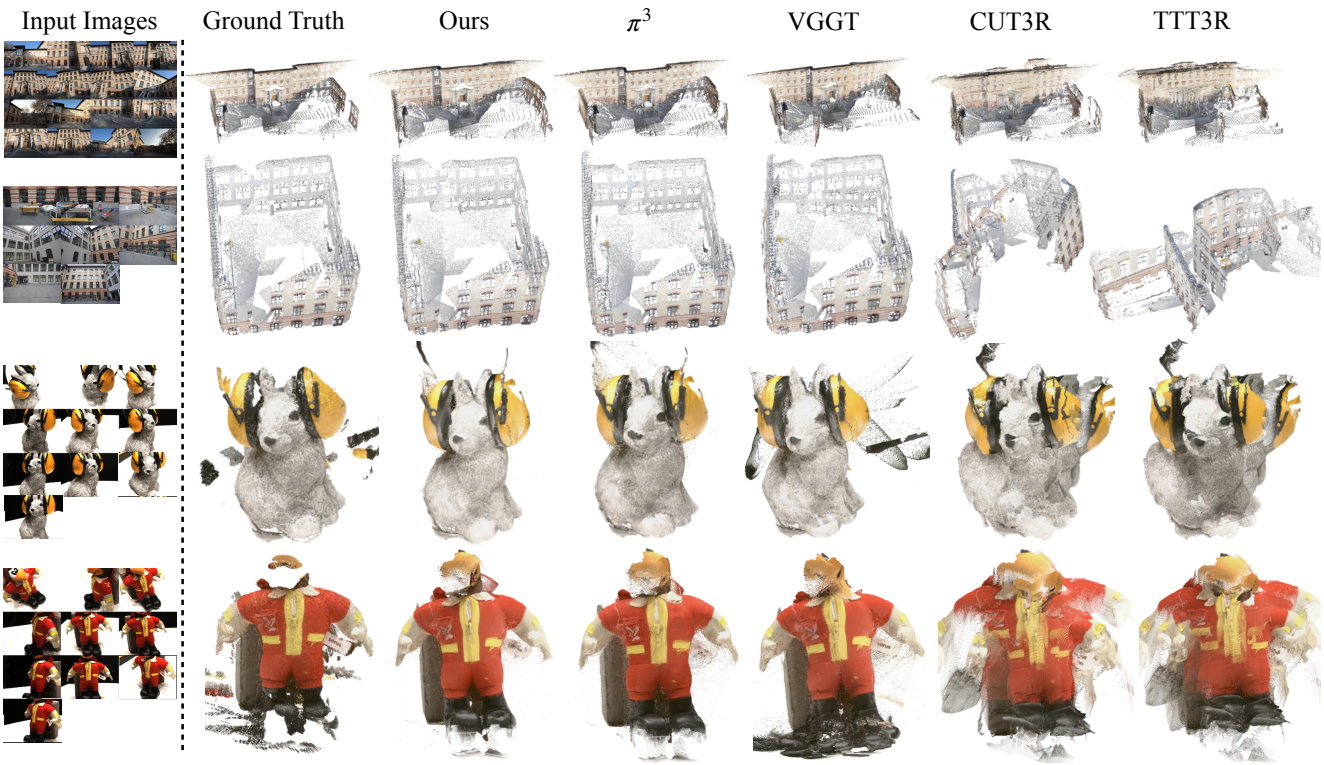

Figure 6. **Qualitative comparison.** Point cloud reconstructions of scenes from the ETH3D and DTU datasets.

ing Aria Synthetic Environments [43], ARKitScenes [5], BlendedMVS [81], Co3dv2 [48], DL3DV [37], GTA-SfM [69], Hypersim [49], MapFree [2], Matrixcity [35], Matterport3D [12], MegaDepth [36], MidAir [17], MVS-Synth [25], OmniObject3D [77], ScanNet [14], ScanNet++ [83], ScenenetRGBD [38], TartanAir [75], TartanGround [44], Unreal4k [87], Virtual KITTI [11], Waymo [63], WildRGBD. We also use 6 dynamic scene datasets, including BEDLAM [8], Dynamic Replica [31], Kubric [21], OmniWorld [90], PointOdyssey [88], and Spring [39].

B.2. Full Training Loss

In addition to the training loss described in the main paper, we also use a normal loss $\mathcal{L}_{\text{point-normal}}$ to supervise local point map prediction, and a gradient loss $\mathcal{L}_{\text{depth-grad}}$ to regularize predicted depths to be locally smooth:

$$\mathcal{L}_{\text{point-normal}} = \underset{i,j}{\text{mean}}\big(\arccos\big(\mathbf{n}_{i,j} \cdot \mathbf{n}^*_{i,j}\big)\big)\,, \tag{14}$$

$$\mathcal{L}_{\text{depth-grad}} = \underset{i}{\text{mean}}(\|\Sigma_i \circ (\nabla(\hat{s}D_i) - \nabla D^*_i)\|_1)\,, \tag{15}$$

where the normal $\mathbf{n}_{i,j}$ of pixel j from view i is obtained by computing the cross product of its adjacent edges on the predicted local point map, and $\mathbf{n}^*_{i,j}$ is computed from the ground truth point map with the same procedure.

B.3. More Implementation Details

Training Implementation. Our model is trained with Fully Sharded Data Parallel (FSDP), and we apply `torch.compile` to the test-time training block to accelerate training. Following VGGT [68], we randomly apply color jitter, Gaussian blur, and grayscale to the input frames as data augmentation. For training stability, we normalize the ground-truth cameras, depths, and local points using the global point cloud scale. During training, input images are resized to a width of 518 pixels with a random aspect ratio sampled from $[0.33, 1.0]$. For each scene, we randomly sample 2–48 frames and cap the number of images per GPU at 48.

Finetune the Model for Scene State Query. We fine-tune the trained model to enable scene state queries. We use the first input frame as the reference view and express the target query camera pose in the coordinate system of this reference view. Our camera prediction is scale-invariant, but we need to fix the scale of the target camera to improve training stability. Therefore, the scale of the target camera is different from the predicted cameras. Specifically, we determine the target camera translation scale from the maximum distance of all input camera centers to the origin (the camera center of the reference view).

We fine-tune the model for 100K iterations. During fine-tuning, we keep all other training losses unchanged and additionally include the query losses described in the main paper. Since the RGB loss requires photometrically consistent inputs, we disable color-based data augmentation (color jitter, Gaussian blur, and grayscale) on the input frames. In addition, we exclude dynamic datasets and static datasets with inconsistent image collections, such as MegaDepth [36]. We observed that the LPIPS loss introduces substantial extra GPU memory overhead. Therefore, during fine-tuning we reduce the maximum number of images per GPU from 48 to 44. For each scene, we randomly sample 4–44 frames and use half of them as input frames and the other half as target frames. Consequently, each training example uses 2–22 input views, and the number of target views matches the number of input views.

Finetune the Model for Streaming Reconstruction. To enable streaming reconstruction, we replace the transformer-based camera head with a lightweight two-layer MLP, and finetune the Stage-3 checkpoint (trained without an explicit reference view) on 32 H100 GPUs. We train the model on all datasets used before. We first train it for 60k steps with a learning rate of $1e{-5}$, using 36 images per GPU (12 images per scene), and then continue for another 30k steps at the same learning rate with longer context (24 images per scene), using 48 images per GPU. *We observe a notable gain when increasing the training context from 12 to 24 views.* Due to time constraints we stop at 24 views; however, since our streaming baselines are trained with longer contexts (up to 64 views), we reasonably expect further scaling the context length toward 64 views to yield an even larger advantage.

C. More Results for the Implicit Scene State

In Figure 5 of the main paper, we demonstrate our model's ability to infer scene structure in unseen regions.

In Figure 7, we further show that the reconstructed implicit scene state can be directly queried at novel view camera poses to obtain RGB and depth predictions. These predictions can then be back-projected into 3D to form colored point cloud. Notably, the point cloud attained solely from state queries closely resembles the geometry and appearance of the point cloud reconstructed from the input images, indicating that the learned scene state faithfully captures both the geometry and appearance of the underlying scene.

D. More Evaluation Results

Table 8. **Monocular Depth Estimation.** We evalute the frame-independent monocular depth estimation on the Sintel [9], Bonn [41], KITTI [20] and NYU-v2 [59] datasets.

	Sintel		Bonn		KITTI		NYU-v2	
Method	AbsRel↓	$\delta < 1.25$↑	AbsRel↓	$\delta < 1.25$↑	AbsRel↓	$\delta < 1.25$↑	AbsRel↓	$\delta < 1.25$↑
MASt3R [34]	0.413	0.569	0.123	0.833	0.077	0.948	0.110	0.865
MonST3R [84]	0.402	0.525	0.069	0.954	0.098	0.895	0.094	0.887
Fast3R [79]	0.544	0.509	0.169	0.796	0.120	0.861	0.093	0.898
CUT3R [71]	0.418	0.520	0.058	0.967	0.097	0.914	0.081	0.914
FLARE [85]	0.606	0.402	0.130	0.836	0.312	0.513	0.089	0.898
MoGe v1 [72]	0.273	0.695	0.050	0.976	0.054	0.977	0.055	0.952
MoGe v2 [73]	0.277	0.687	0.063	0.973	0.049	0.979	0.060	0.940
VGGT [68]	0.329	0.600	0.051	0.974	0.089	0.939	0.055	0.953
π^3 [76]	0.276	0.622	0.052	0.971	0.059	0.972	0.054	0.956
Ours	0.268	0.666	0.056	0.973	0.063	0.960	0.052	0.959

Table 9. **Video Depth Estimation: Sintel [9], Bonn [41] and KITTI [20].** We have reported results under scale-only alignment in the main paper. Here we further report results using **joint scale-and-shift** alignment.

		Sintel		Bonn		KITTI	
Method	**Params**	AbsRel↓	$\delta < 1.25$↑	AbsRel↓	$\delta < 1.25$↑	AbsRel↓	$\delta < 1.25$↑
$\mathcal{O}(N^2)$ Inference Speed							
Fast3R [79]	648M	0.518	0.486	0.196	0.768	0.139	0.808
FLARE [85]	1.40B	0.791	0.358	0.142	0.797	0.357	0.579
VGGT [68]	1.26B	0.226	0.683	0.049	0.974	0.059	0.961
π^3 [76]	959M	0.206	0.735	0.045	0.976	0.036	0.986
$\mathcal{O}(N)$ Inference Speed							
CUT3R [71]	793M	0.534	0.551	0.067	0.961	0.124	0.850
TTT3R [13]	793M	0.508	0.566	0.054	0.973	0.120	0.870
Ours	1.40B	0.198	0.731	0.052	0.973	0.050	0.972

D.1. Monocular Depth Estimation

We present quantitative results for monocular depth estimation in Table 8, evaluated on four standard benchmarks. Overall, our method consistently outperforms VGGT [68] and π^3 [76], and performs comparably to the state-of-the-art monocular depth estimator MoGe [72, 73] despite our model never having been trained with purely monocular input.

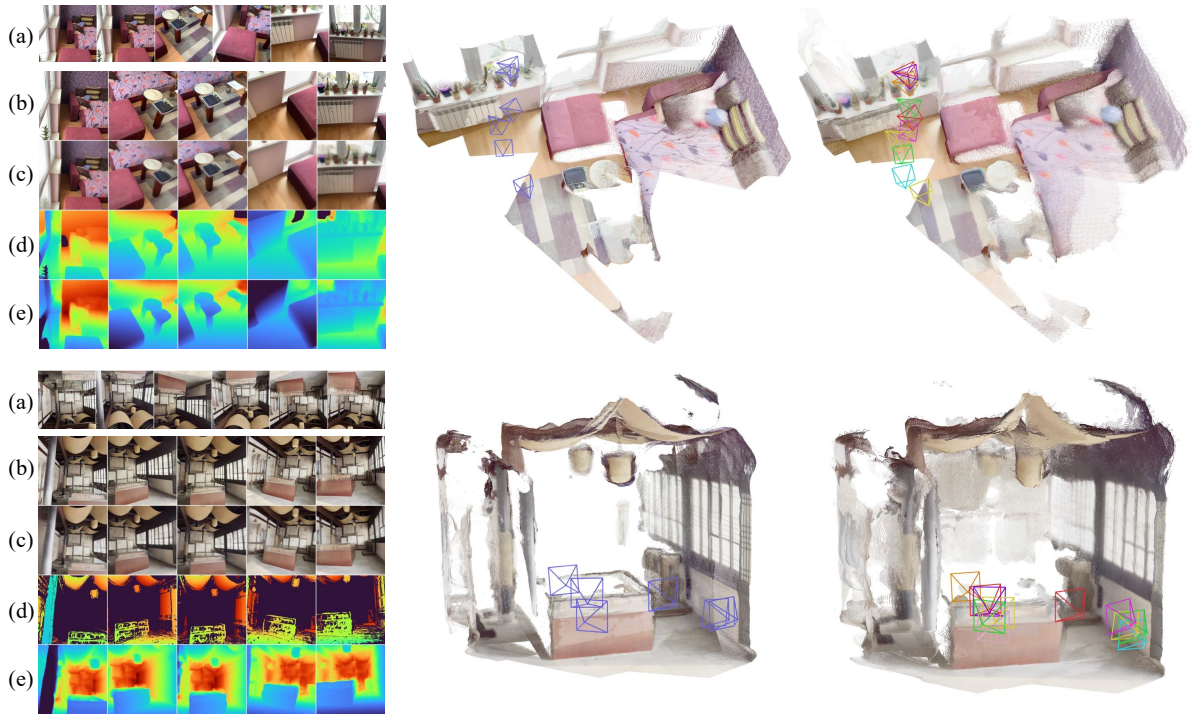

Figure 7. **Querying the Scene State.** The left panels show: input images (a), GT RGB at query poses (b), our RGB predictions (c), GT depth (d), and predicted depth (e). The middle panels visualize the 3D point clouds reconstructed from the input images. The right panels show point clouds attained **solely by querying the scene state**. The close visual match between these two point clouds indicates that the learned scene state faithfully captures the geometry and appearance of the input scene.

D.2. Video Depth Estimation

We have reported video depth estimation results under scale-only alignment in the main paper. In Table 9, we further report results using **joint scale-and-shift** alignment.

D.3. Qualitative Comparison

In Figure 6, we show a qualitative comparison on the DTU [26] and ETH3D [55] datasets. Quantitative results for these datasets are shown in Table 3 of the main paper.

D.4. Effects of Removing the Reference View

As described in the main paper, our model is trained in three stages. In the final stage, we remove the explicit reference-view selection: instead of treating the first frame as a reference camera and expressing all poses in its coordinate frame, we fine-tune the model using the affine-invariant camera loss proposed by π^3 [76]. This loss computes relative pose errors between pairs of views, making the supervision independent of any particular reference frame (see π^3 [76] for further details). To evaluate the effect of removing the reference view, we can compare the checkpoint from stage 2 ("Ours w/ ref") with the checkpoint from stage 3 ("Ours w/o ref"). As shown in Tables 10, 11, and 12, we find that in our case, neither variant shows a clear or consistent advantage over the other on the standard benchmarks used in Section 4 of the main paper. That said, we observe that removing the

Table 10. **Ablation: Removing the Reference View (Camera pose estimation).**

Method	Sintel			TUM-dynamics			ScanNet (seen)		
	ATE↓	RPE trans↓	RPE rot↓	ATE↓	RPE trans↓	RPE rot↓	ATE↓	RPE trans↓	RPE rot↓
Ours w/ ref	**0.125**	**0.058**	**0.420**	**0.012**	**0.009**	**0.309**	**0.034**	**0.015**	0.398
Ours w/o ref	0.132	0.066	0.438	**0.012**	0.010	0.310	**0.034**	**0.015**	**0.385**

Table 11. **Ablation: Removing the Reference View (Video depth estimation).** We use "Joint Scale & Shift" alignment here.

Method	Sintel		Bonn		KITTI	
	AbsRel↓	$\delta < 1.25$↑	AbsRel↓	$\delta < 1.25$↑	AbsRel↓	$\delta < 1.25$↑
Ours w/ ref	0.205	**0.731**	0.053	**0.973**	**0.048**	**0.972**
Ours w/o ref	**0.198**	**0.731**	**0.052**	**0.973**	0.050	**0.972**

Table 12. **Ablation: Removing the Reference View (Point Map Estimation).**

Method	DTU						ETH3D					
	Acc. ↓		Comp. ↓		N.C. ↑		Acc. ↓		Comp. ↓		N.C. ↑	
	Mean	Med.	Mean	Med.	Mean	Med.	Mean	Med.	Mean	Med.	Mean	Med.
Ours w/ ref	1.584	0.901	**1.558**	0.667	**0.687**	**0.779**	**0.202**	**0.138**	0.413	0.278	0.852	0.953
Ours w/o ref	**1.228**	**0.671**	1.649	**0.663**	0.675	0.764	0.254	0.171	**0.249**	**0.159**	**0.865**	**0.965**

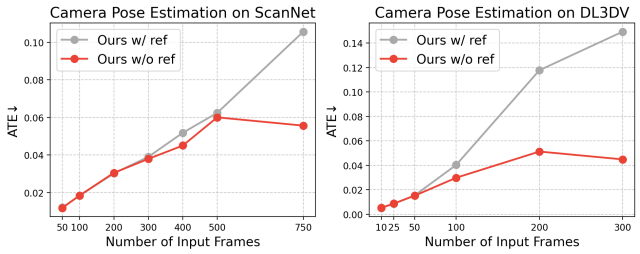

Figure 8. **Long-sequence camera estimation.** We evaluate camera ATE on the ScanNet-v2 and DL3DV datasets by taking the first N frames of each test sequence and gradually increasing N. We see that, when the input sequence length becomes long, removing the reference view and fine-tuning with the affine-invariant camera loss from π^3 [76] ("Ours w/o ref") improves the camera pose estimation accuracy compared to the reference-based variant ("Ours w/ ref").

reference view improves accuracy for long input sequences (as shown in Figure 8), hence its inclusion in our complete model.

Table 13. **Streaming Video Depth Estimation.** We report results under **scale-only** alignment on Sintel [9], Bonn [41] and KITTI [20].

Method	Sintel		Bonn		KITTI	
	AbsRel↓	$\delta < 1.25$↑	AbsRel↓	$\delta < 1.25$↑	AbsRel↓	$\delta < 1.25$↑
CUT3R [71]	0.432	0.510	0.072	0.951	0.152	0.805
TTT3R [13]	0.426	0.522	**0.061**	**0.970**	0.149	0.812
Ours-streaming	**0.273**	**0.638**	0.067	0.965	**0.100**	**0.903**

Table 14. **Streaming Camera Pose Estimation: Sintel [9] and Co3Dv2 [48].** For Sintel, we report ATE and RPE translation/rotation errors. For Co3Dv2, we report pose AUC under angular error thresholds of 5/15/30 degrees.

Method	Sintel			Co3Dv2		
	ATE↓	RPE trans↓	RPE rot↓	AUC@5↑	AUC@15↑	AUC@30↑
CUT3R [71]	0.2160	0.0710	**0.6220**	24.88	56.28	71.72
TTT3R [13]	0.2040	0.0850	0.6900	22.61	53.49	69.46
Ours-Streaming	**0.1593**	**0.0655**	0.7508	**45.38**	**72.58**	**83.12**

Table 15. **Streaming Point Map Reconstruction Comparison.** We report the results on DTU [26], ETH3D [55], and NRGBD-dense.

Method	DTU			ETH3D			NRGBD-dense		
	Acc. ↓	Comp. ↓	N.C. ↑	Acc. ↓	Comp. ↓	N.C. ↑	Acc. ↓	Comp. ↓	N.C. ↑
CUT3R [71]	5.045	6.437	0.666	**0.593**	**0.747**	**0.754**	0.065	0.036	0.812
TTT3R [13]	5.337	6.593	0.666	0.763	0.881	0.739	0.074	0.037	0.803
Ours-streaming	**4.091**	**3.495**	**0.693**	0.614	0.941	0.750	**0.038**	**0.028**	**0.836**

D.5. Streaming Reconstruction Comparison

With a simple fine-tuning procedure, we deploy our model in a streaming setting by updating the TTT-based scene state one view at a time. As shown in Tables 13, 15, and 14, our streaming variant generally outperforms CUT3R and TTT3R across point-map reconstruction, video depth, and camera pose estimation. Notably, due to time constraints we only finetune the model with 24-view context length and our streaming baselines are trained with longer contexts (up to 64 views), we reasonably expect that further scaling the finetuning context length will yield an even larger advantage.

D.6. More Long-Sequence Evaluation

We show more long sequence evaluation results on video depth estimation and 3D point estimation in Figure 9 and Figure 10, respectively.

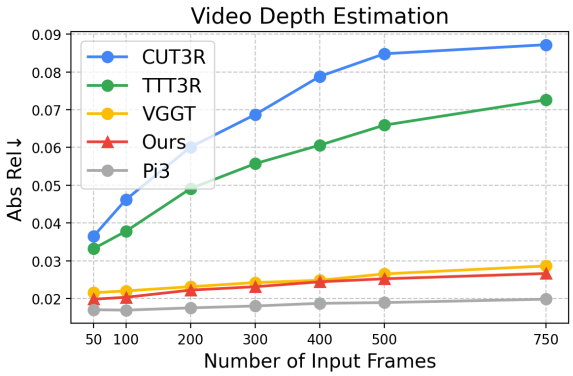

Figure 9. **Long-sequence video depth estimation.** We evaluate on the ScanNet-v2 dataset by taking the first N frames of each test sequence and gradually increasing N.

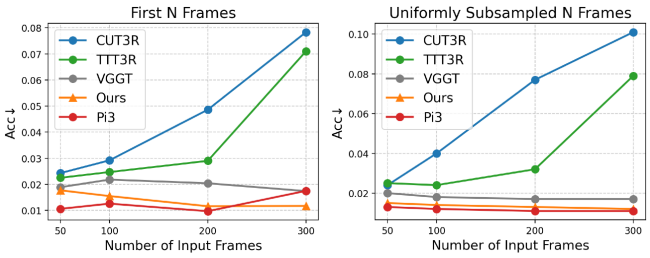

Figure 10. **Long-sequence 3D point estimation.** We evaluate on the 7-Scenes [58] dataset under two cases. **Left**: increasing *scene scale* by using the first N frames of each sequence; **Right**: increasing *view density* by uniformly subsampling N frames.

E. Limitations

Though our model achieves high reconstruction accuracy and fast inference speeds, it has several limitations. First, we observe noticeable performance degradation when evaluating very long sequences where the scene scale extends far beyond the training distribution. This limitation appears to be shared by all existing feed-forward methods. Promising directions to address this issue include: (i) training the model

Table 16. **NVS Evaluation on Mip-NeRF360 [4].** Baseline results are taken from Tab. 1 of the AnySplat paper. ZipMap is competitive with several baselines. However, as noted in our Limitations, our query results are primarily designed to generate colored point clouds rather than high-fidelity novel view synthesis. Accordingly, NVS is not our main focus, and we do not claim state-of-the-art performance compared with explicit 3DGS-based methods.

Method	6 Views			16 Views		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
NoPoSplat [82]	15.92	0.416	0.541	15.47	0.361	0.606
Flare [85]	15.35	0.407	0.573	13.21	0.348	0.695
AnySplat [28]	18.32	0.524	0.329	21.85	0.670	0.250
Ours	15.60	0.414	0.486	17.65	0.459	0.374

on longer sequences using efficient large-context training strategies such as context parallelism (CP); and (ii) combining our methods with global alignment techniques, such as VGGT-Long [16]. Thankfully, because our method offers significantly faster runtimes on long sequences compared to prior approaches, it may be well-suited to training on longer sequences at higher throughput rates than prior models. Second, although our experiments demonstrate that our model can query RGB information from the implicit scene representation with novel view pose conditions, the resulting novel views often suffer from blurry artifacts in high-frequency regions. We suspect this is partly caused by a mismatch between the GT camera pose used for conditioning and the geometry implicitly encoded in the state learned from unposed images, which may make training RGB supervision noisy. In our experiments, we primarily focus on visualizing the queried point cloud from the implicit representation. We do not claim that our current method enables SoTA high-quality unposed novel view synthesis, as shown in Tab 16. Improving the RGB rendering quality of our model to support high-fidelity unposed novel view synthesis remains an interesting future work.