SAIL-Recon: Large SfM by Augmenting Scene Regression with Localization

Supplementary Material

In this supplementary material, we provide additional implementation details and experimental setups in Sec.6. We also present further experiments and discussions in Sec.7. More visualization result will be posted in Sec. 8. We also provide a video in the attached files.

6. More Implementation Details

Training Details. As described, we follow the training set of VGGT [86], which we use a cosine learning rate scheduler with a maximum learning rate of 2×10^{-4} and a warm-up of 2K iterations. The input images are resized to a maximum of 518 pixels while preserving the aspect ratios between $[0.33, 1.0]$. Data augmentation includes random color jittering, Gaussian blur, and grayscale conversion. Training is performed with bfloat16 precision, gradient checkpointing, and a mixed anchor–query frame strategy.

Implementation of attn$^{\text{query}}$. We apply an attention mask to realize a cross-attention–like operation between tokens from the query image $\mathcal{I}^q$ and the scene representation $\mathcal{R}_j = \Theta(t_j^{\prime\mathcal{I}})$ from layer j. Specifically, during training, the mask enforces two types of interaction:

- Anchor interaction: tokens from anchor frames are allowed to attend to each other, enabling mutual information exchange across different anchor views
- Query restriction: tokens from query frames cannot attend to tokens from other query frames; they can only attend to tokens within the same frame and to the scene representation $\mathcal{R}_j$.

This design ensures that query tokens extract information primarily from the global scene representation and their own local context, while anchor tokens remain fully connected to maximize cross-view aggregation. The same attention mask is also applied to the attention layer in the pose head.We also develop an alternative version that employs two distinct blocks for the anchor and query, respectively. The query block is initialized from the anchor block and fine-tuned during training. We observe that both versions perform similarly.

Inference Details. As stated in Sec.3.1, we first extract the scene representation $\mathcal{R}$ from the anchor frames and then process query images sequentially. Specifically, we employ a KV-cache[50] to store $\mathcal{R}$ as the keys and values in each global attention layer, which effectively accelerates computation and reduces memory usage. For each subsequent query image, its tokens serve as the queries in the attention mechanism, while the keys/values are formed by concatenating the query tokens with the cached scene tokens. Through the attention operation, the query image tokens are updated by aggregating information from the global scene representation. After passing through all attention layers, we obtain tokens enriched with localization information. These tokens are then fed into the camera head and depth head to predict the corresponding camera parameters, depth and scene coordinate maps, yielding the reconstructed scene from the query viewpoint.

Post Refinement Details We adopt Nerfacto [70] within NeRF Studio, applying its camera pose optimizer for post refinement. For scenes with $\leq 2,000$ images, models undergo 10,000 training iterations with a regularization weight $\lambda = 0.001$ on both camera translation and rotation. The optimizer uses an initial learning rate of 10^{-3}, which decays to 10^{-4} after $1,000$ iterations via cosine annealing. All other parameters follow the defaults of NeRF Studio. For scenes with $\geq 2,000$ images, we perform two separate optimizations with 10000 and 30000 iterations, respectively. The second round uses the poses from the end of the first round as initialization, and the camera optimizer's learning rate begins at 0.0005 and reduces to 0.00001. Other parameters remain constant. Optimizations typically take 2.5 minutes for every 10k iterations, regardless of the number of images.

More experimental setup.

- We denote DROID-SLAM* as the variant that first calibrates intrinsics using GeoCalib [81] on the first image of each sequence and then uses the calibrated parameters in DROID-SLAM.
- All experiments are conducted on an NVIDIA RTX 4090 GPU; To align with V100-based results, runtimes are scaled by a factor of 1.5, reflecting the measured FP16 inference speed gap. For anchor frame selection, we use 50 frames per scene in 7 scenes, with PSNR reported in the combined training and test sets, and the relocalization accuracy evaluated on the test set after ACE0 [9]. For mip-NeRF 360, 50 anchors are selected per sequence. For Tanks and Temples, we select an average of 100 keyframes per sequence to relocalize all remaining images and video frames. For TUM RGB-D, we use 50–100 anchors depending on sequence length: floor, plant, and teddy use 100 frames, and others use 50.
- In cases where the first frame contains limited semantic information, it is replaced with a semantically richer frame as the first anchor.
- For visualization in Fig. 1, we remove points in the lowest 50% confidence on the depth confidence map, corresponding to sky, glass, and other ambiguous surfaces, and apply moderate point cloud downsampling to enhance visual clarity.
- We cite the results in Tabs. 3, 4, 5 from ACE0 [9]. De-

tails on default parameters and configurations for baselines such as COLMAP (default), COLMAP (Sparse + Reloc + BA) and Nope-NeRF are available in the supplementary material of ACE0 [9].

7. Additional Results

7.1. Pose Estimation

TUM RGBD. We report the root mean square error (RMSE) of the absolute trajectory error (ATE), comparing our method with a broader set of state-of-the-art approaches [11, 14, 41, 45, 75, 76, 100] under calibrated settings, as summarized in Tab. 8. Our method achieves accuracy on par with the most advanced SLAM systems while remaining robust across diverse sequences without requiring camera calibration. Compared with geometry-based pipelines, such as ORB-SLAM3, our approach exhibits stronger robustness and achieves comparable or superior accuracy to learning-based baselines. The main weakness appears in the floor sequence, where images contain limited visual cues dominated by textureless floor regions. In this case, reference view selection becomes critical: large viewpoint gaps between the query and reference views significantly degrade localization. We further visualize these effects in the trajectory results (Sec. 8).

7.2. Novel View Synthesis

Tanks & Temples. To further evaluate our scalability for large-scale reconstruction, we apply our method to the Tanks & Temples [32] video sequences. For each sequence, we uniformly sample 100 images as anchors and perform localization on all frames.

For clarity, Table **??** reports the results on both the image set and the full video sequences, with the latter shown on the right. As a reference, we include *Sparse COLMAP + Reloc + BA* (CMP (SRB)), which initializes from a sparse COLMAP reconstruction using 150–500 images, registers the remaining frames and performs global bundle adjustment. Our approach consistently outperforms RealityCapture, DROID-SLAM [76], and ACE0 [9] across all splits, with only ACE0 initialized from sparse COLMAP poses (CMP + ACE0) achieving comparable performance. In particular, on the most challenging 'advanced' split, our method achieves the highest PSNR of all methods. Despite each sequence containing more than 10k images on average, our feedforward approach maintains competitive efficiency - only slightly slower than SLAM-based pipelines - while delivering strong reconstruction quality.

7 Scenes. We quote the table from ACE0 [9] and report our results in Tab. 10. For each 7-Scenes sequence, we uniformly sample 50 frames from the train/test splits and estimate poses for all images in the scene. We compare against COLMAP since bundle-adjusted COLMAP poses provide a more accurate reference. Our method attains PSNR comparable to the COLMAP reference and exceeds ACE0. In terms of runtime, even under "fast" settings COLMAP still requires around 13 h per scene; DROID-SLAM returns results quickly but performs poorly on 7-Scenes; ACE0 takes 1 h. In contrast, our approach finishes in 25 min without any pose initialization, achieves higher PSNR than ACE0, and matches the PSNR of ACE0 when initialized from KinectFusion (KF-Init., 7 min).

To further compare against learning-based approaches under constrained memory budgets, we follows [9] to downsample each sequence to 200 frames. Sequential-dependent scene regression models (e.g., Cut3R [87], SLAM3R [42]) require dense temporal input, and VGGT [86] still exceeds memory limits on 200 images. We therefore compare to BARF [37] and NoPe-NeRF [5]: Both BARF [37] and Nope-NeRF [5] fail to recover the scene after a long fitting time. Using our localization of all frames, we consistently obtain the highest PSNR in this 200-frame setting. while remaining faster than these baselines.

8. Visualization

As shown in Fig. 4, 5 and 6, we show the render images of test view in the three different splits of Tank & Temple dataset. We illustrate the test view of *7-Scenes* and *Mip-NeRF 360* dataset in Fig. 7 and 8, respectively. We aslo suppliment in Fig. 9 our regressed camera poses and point clouds.

	Method	360	desk	desk2	floor	plant	room	rpy	teddy	xyz	Avg
Calib.	ORB-SLAM3 [11]	×	0.017	0.210	×	0.034	×	×	×	**0.009**	N/A
	DeepV2D [75]	0.243	0.166	0.379	1.653	0.203	0.246	0.105	0.316	0.064	0.375
	DeepFactors [14]	0.159	0.170	0.253	0.169	0.305	0.364	0.043	0.601	0.035	0.233
	DPV-SLAM [41]	0.112	0.018	0.029	0.057	0.021	0.330	0.030	0.084	0.010	0.076
	DPV-SLAM++ [41]	0.132	0.018	0.029	0.050	0.022	0.096	0.032	0.098	0.010	0.054
	GO-SLAM [100]	0.089	**0.016**	0.028	0.025	0.026	0.052	**0.019**	0.048	0.010	0.035
	DROID-SLAM [76]	0.111	0.018	0.042	**0.021**	**0.016**	**0.049**	0.026	0.048	0.012	0.038
	MASt3R-SLAM [45]	**0.049**	**0.016**	**0.024**	0.025	0.020	0.061	0.027	**0.041**	**0.009**	**0.030**
Uncalib.	DROID-SLAM* [76]	0.202	0.032	0.091	0.064	0.045	0.918	0.056	0.045	0.012	0.158
	MASt3R-SLAM* [45]	0.070	0.035	0.055	0.056	0.035	0.118	0.041	0.114	0.020	0.060
	VGGT-SLAM (Sim(3)) [43]	0.123	0.040	0.055	0.254	0.022	0.088	0.041	0.032	0.016	0.074
	VGGT-SLAM (SL(4)) [43]	0.071	0.025	0.040	0.141	0.023	0.102	0.030	0.034	0.014	0.053
	SAIL-Recon (Offline)	0.070	0.024	0.042	0.107	0.031	0.113	0.020	0.037	0.012	0.051

Table 8. **Root mean square error (RMSE) of absolute trajectory error (ATE) on TUM RGB-D [66] (unit: m)**. Gray rows denote results obtained with calibrated camera intrinsics, while entries marked with * indicate evaluation in the uncalibrated setting. We color result in: Best, Second, and Third. Note that our method is actually a offline SfM method.

		Frames	CMP (D)	Reality Capture	DROID-SLAM† [76]	ACE0	Ours	Frames	CMP (SRB)	CMP+ ACE0	Reality Capture	DROID-SLAM† [76]	ACE0	Ours
Training	Barn	410	24.0	21.2	19.0	16.5	23.5	19.3k	26.3	25.1	16.9	13.5	17.7	25.1
	Catpr.	383	17.1	15.9	16.6	16.9	16.8	11.4k	18.7	18.8	17.9	18.9	18.6	17.5
	Church	507	18.3	17.6	14.3	17.2	17.0	19.3k	18.5	17.3	-	11.5	16.5	15.8
	Ignatius	264	20.1	17.7	17.8	19.8	19.5	7.8k	20.9	20.7	18.6	19.1	20.7	20.7
	MtgRm.	371	18.6	18.1	15.6	18.0	19.5	11.1k	20.8	20.3	18.2	17.1	16.6	20.4
	Truck	251	21.1	19.0	18.3	20.1	20.9	7.5k	23.4	23.1	19.1	20.6	23.0	23.5
	Average	364	19.9	18.2	16.9	18.1	19.5	14.6k	21.4	20.9	18.2	16.8	18.9	20.5
	Time		1h	3min	5min	1.1h	3.5min		8h	1.8h	14h	18min	2.2h	58min
Intermediate	Family	152	19.5	18.8	17.6	19.0	20.6	4.4k	21.3	21.3	19.8	19.8	18.0	21.3
	Francis	302	21.6	20.7	20.7	20.1	21.8	7.8k	22.5	22.7	20.4	21.8	21.7	22.8
	Horse	151	19.2	19.0	16.3	19.5	20.1	6.0k	22.6	22.3	20.7	19.2	21.7	21.9
	LightH.	309	16.6	16.5	13.6	17.5	18.2	8.3k	19.5	20.5	16.6	18.9	18.6	19.7
	PlayGd.	307	19.1	19.2	11.4	18.7	20.3	7.7k	21.2	21.0	16.5	11.3	20.4	21.7
	Train	301	16.8	15.4	13.8	16.2	16.2	12.6k	19.8	18.5	14.4	15.6	18.5	18.5
	Average	254	18.8	18.3	15.6	18.5	19.5	7.8k	21.1	21.0	18.1	17.8	19.8	21.0
	Time		32min	2min	3min	1.3h	3min		5h	1h	11h	14min	2.2h	30min
Advanced	Audtrm.	302	19.6	12.2	16.7	18.7	20.3	13.6k	21.4	19.8	-	16.6	20.0	21.0
	BallRm.	324	16.3	18.3	13.1	17.9	14.8	10.8k	18.0	15.6	-	10.4	18.9	16.9
	CortRm.	301	18.2	17.2	12.3	17.1	17.4	12.6k	18.7	17.8	-	10.2	16.3	17.4
	Palace	509	14.2	11.7	10.8	10.7	14.3	21.9k	15.3	12.3	-	18.6	11.0	13.3
	Temple	302	18.1	15.7	11.8	9.7	17.8	17.5k	19.6	16.1	-	11.9	14.8	18.3
	Average	348	17.3	15.0	12.9	14.8	16.9	15.6k	18.6	16.3	-	11.5	16.2	17.4
	Time		1h	2min	4min	1h	3.5min		10h	2.1h		27min	2.8h	59min

Table 9. **Tanks & Temples.** We show the pose accuracy via view synthesis with Nerfacto [70] as PSNR in dB, and the reconstruction time. We color code in: Best, Second, and Third. †Method needs sequential inputs.

		Pseudo GT			All Frames				200 Frames			
	Frames	Kinect Fusion	COLMAP (default)	COLMAP (fast)	DROID-SLAM†	ACE0 (default)	ACE0 (KF Init.)	Ours (50F)	BARF [37]	NoPE-NeRF	ACE0 (default)	Ours (50F)
Chess	6k	19.6	23.6	23.5	19.3	23.3	23.0	23.4	12.8	12.6	22.7	21.8
Fire	4k	19.2	22.6	22.6	13.0	22.3	22.3	22.8	12.7	11.8	22.1	24.4
Heads	2k	17.0	18.8	18.9	17.6	18.8	19.1	18.5	10.7	11.8	19.9	20.2
Office	10k	18.9	21.4	21.6	failed	21.1	21.5	20.9	11.9	10.9	19.8	19.4
Pumkin	6k	19.9	24.1	23.8	18.3	24.1	23.8	24.5	19.6	14.2	24.7	25.0
RedKitchen	12k	17.6	21.4	21.4	10.9	20.8	20.9	19.9	11.6	11.2	18.9	20.0
Stairs	3k	19.0	16.7	21.0	13	17.7	19.9	20.6	15.8	15.9	18.8	20.8
Average	6.5k	18.7	21.2	21.8	N/A	21.2	21.5	21.5	13.6	12.6	21.0	21.8
Avg. Time		realtime	38h	13h	18min	1h	7min	25min	8.5h	47h	27min	3min

Table 10. **7-Scenes.** We show the pose accuracy via view synthesis with Nerfacto [70] as PSNR in dB, and the reconstruction time. We color code in: Best, Second, and Third. Our method takes 50 frames as anchor images only, achieves SOTA performance. For some competitors, we had to sub-sample the images due to their computational complexity (right side). †Method needs sequential inputs.

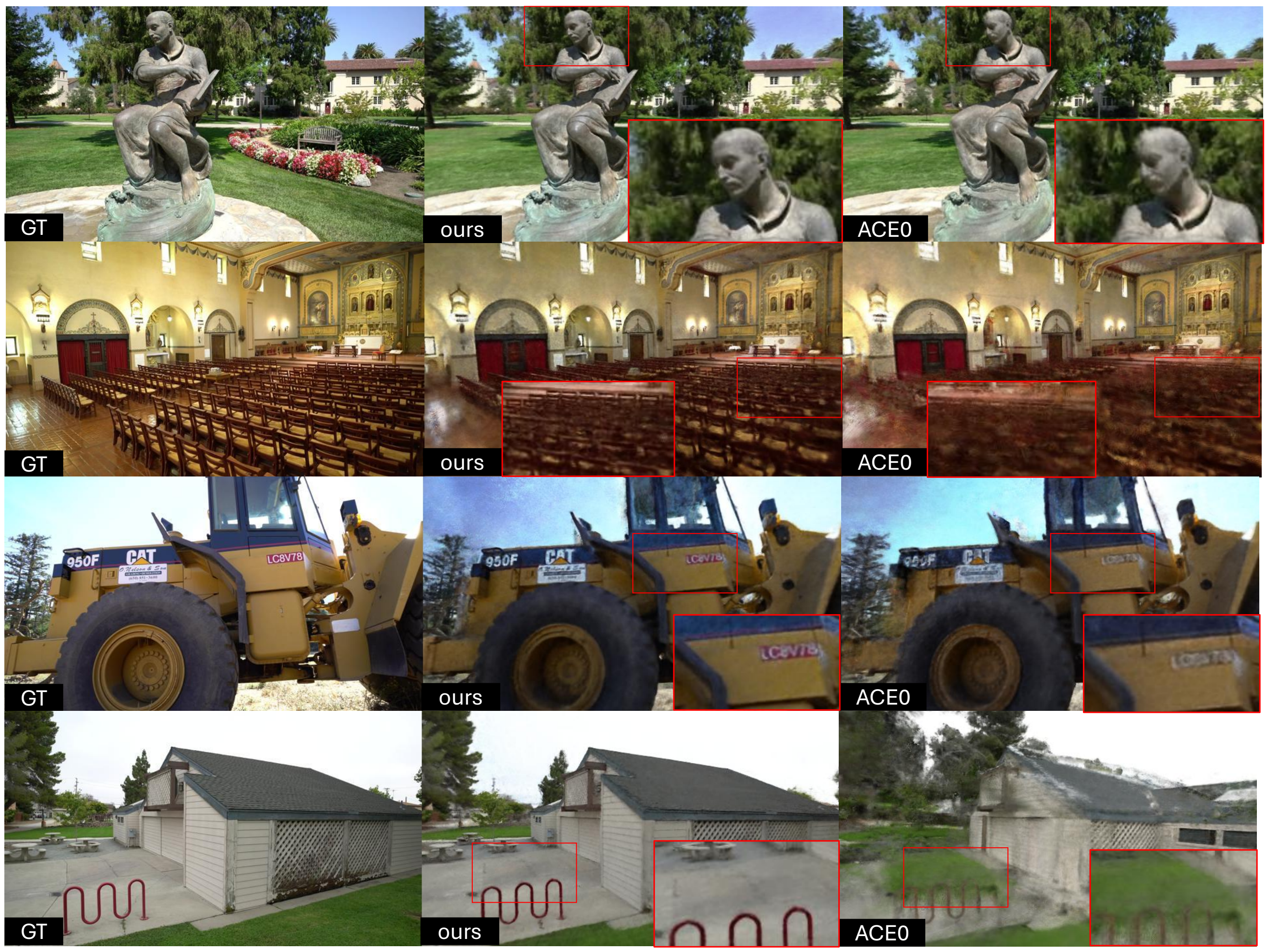

Figure 4. **Visualization on Tank & Temple *training* split.**

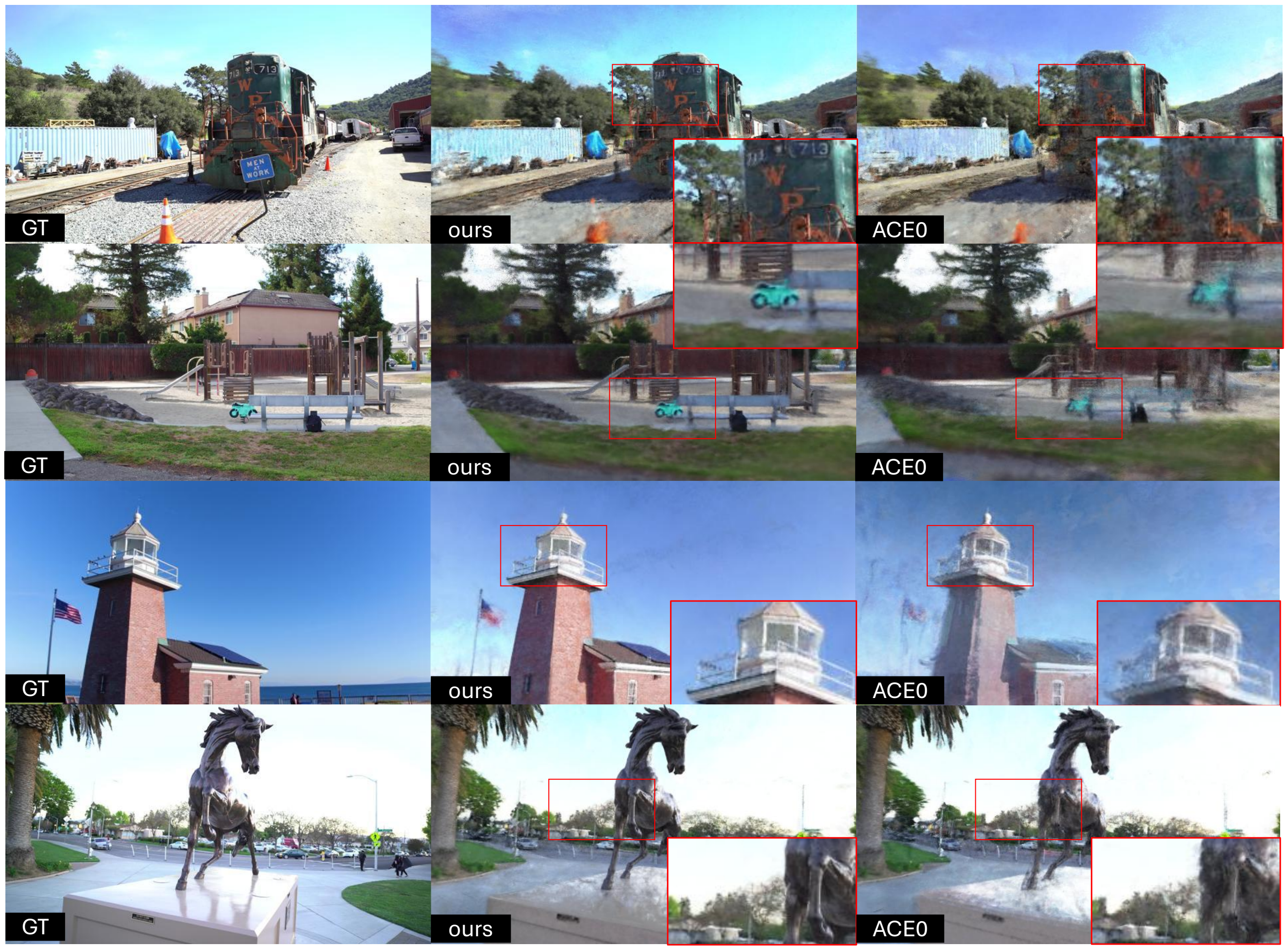

Figure 5. **Visualization on Tank & Temple *intermediate* split.**

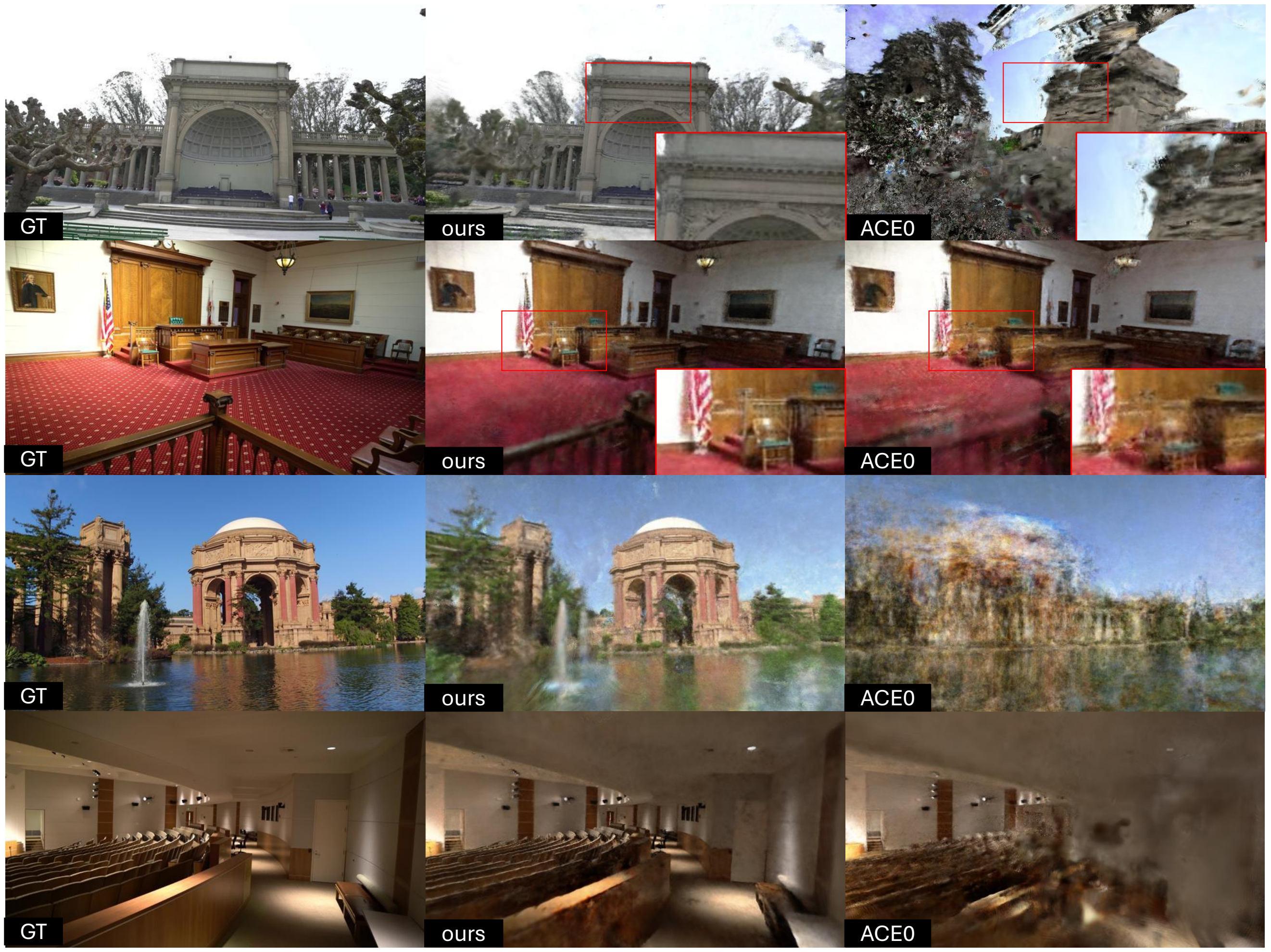

Figure 6. **Visualization on Tank & Temple *advanced* split.**

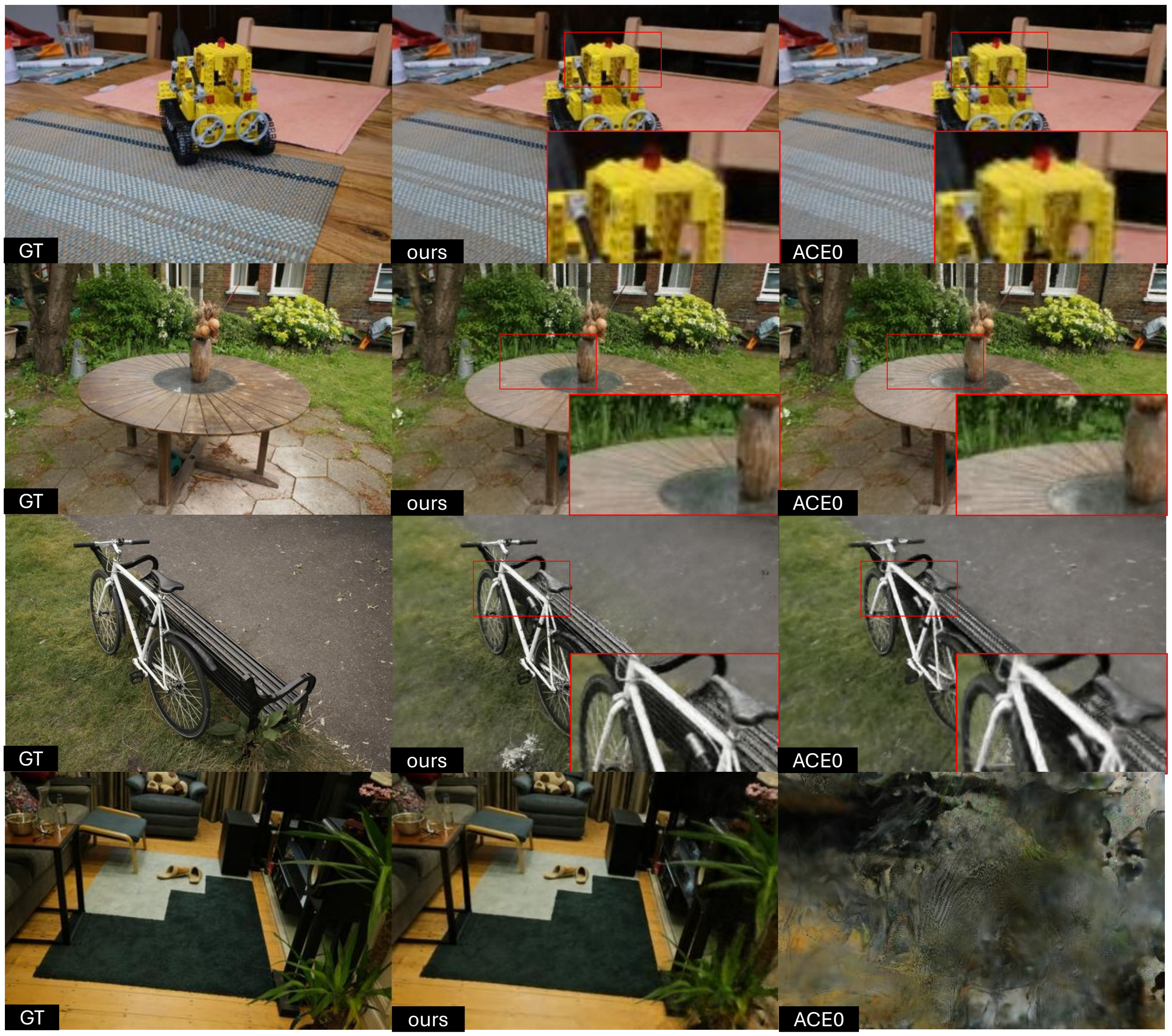

Figure 7. **Visualization on Mip-NeRF 360 dataset.**

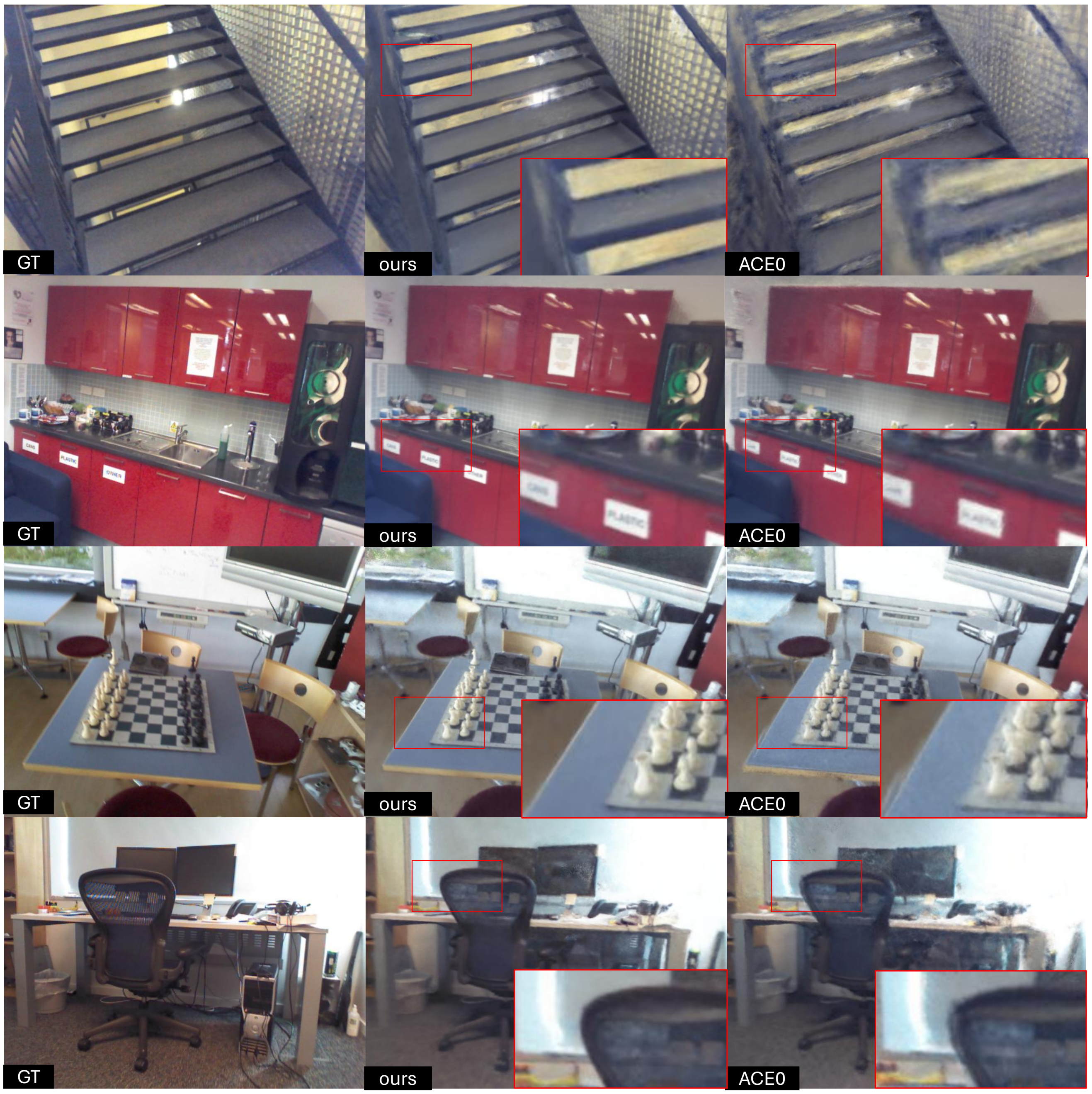

Figure 8. **Visualization on 7-Scenes dataset.**

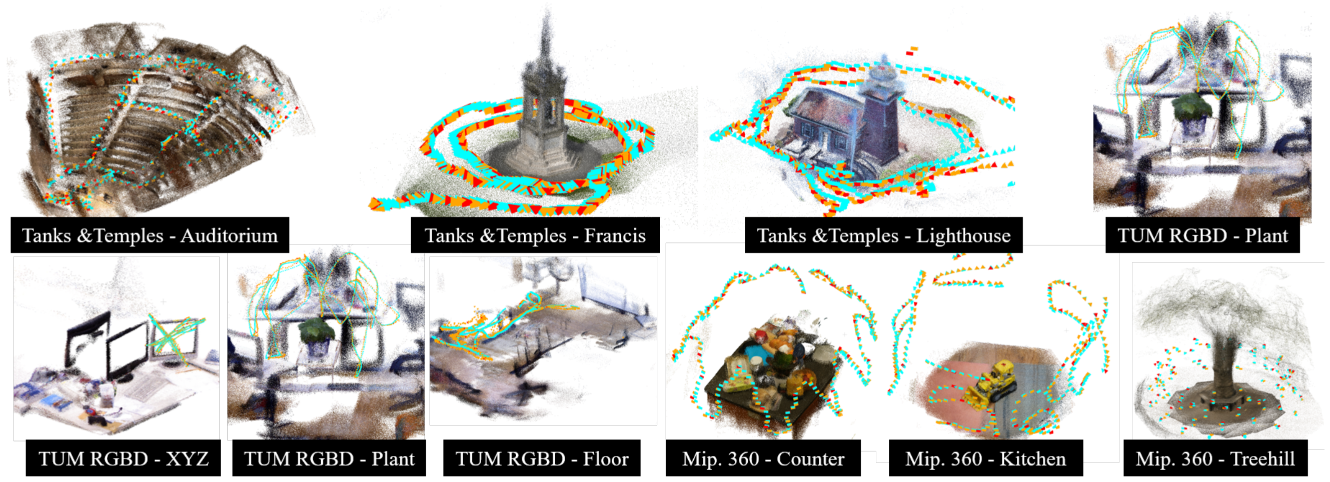

Figure 9. **Regressed Camera Poses and Point Clouds.** We visualize the camera poses and point clouds predicted by SAIL-Recon across various datasets. COLMAP or ground-truth camera poses are shown as blue frustums, while regressed camera poses are shown in yellow, with red indicating anchor images.