iLRM: An Iterative Large 3D Reconstruction Model

Supplementary Material

A. Additional Implementation Details

We initialize model weights using a zero-mean normal distribution with a standard deviation of 0.02. Bias terms are omitted in all Linear and normalization layers. The model is trained using the AdamW [12] optimizer with hyperparameters $\beta_1 = 0.9$ and $\beta_2 = 0.95$. A weight decay of 0.05 is applied to all parameters except the weights of LayerNorm [2]. We use a cosine learning rate schedule with a peak learning rate of 2e-4 and a warmup of 2500 iterations. Our training setup largely follows the configuration proposed in [8, 22].

For the RealEstate10K (RE10K) [23] dataset, the 8-view half-resolution viewpoint setting *(8, H, F)* is trained on 8 H100 GPUs with a total batch size of 256 for 50,000 iterations. Similarly, the 4-view half-resolution viewpoint setting *(4, H, F)* is trained on 8 RTX 4090 GPUs with a total batch size of 128 for 100,000 iterations. The mini-batch cross-attention variants were also trained with the equivalent computational budgets for each viewpoint setting. Lastly, the 2-view full-resolution viewpoint setting *(2, F, F)*, which serves as the reference point, is trained on 8 H100 GPUs for 200,000 iterations.

There are two variants in the DL3DV dataset [11]. **1)** For comparison with MVSplat [5] and DepthSplat [18], we initialize from the pretrained *(8, H, F)* model trained on the RE10K dataset, and finetune it on 8 H100 GPUs with a total batch size of 96 for 100,000 iterations for LR (256×448), and additional 50,000 iterations for HR (512×960). During training, the number of input viewpoints is randomly sampled between 6 and 11 to expose the model to varying numbers of viewpoints/images. Following this stage, the model is further finetuned under the high-resolution setting (512×960). **2)** For comparison with LongLRM [24], which incorporates an undistortion preprocessing step, we adopt the training protocol described in the original work, using 8 H200 GPUs. The training resolution is scheduled in a curriculum of 256×256, 512×512, and 540×960.

Gaussian representations. After the final self-attention layer, the viewpoint features are decoded into Gaussian parameters using a single linear layer with an output dimension of 16. The Gaussian positions, denoted as μ, consist of 5 channels: 2 for the spatial xy offset and 3 for depth, z. The final depth is obtained by averaging the 3 depth channels. Opacity (α) is represented by a single channel. Covariance (Σ) is derived from 3 channels of scale and 4 channels of rotation. Finally, color (c) is represented using 3 channels. Higher-order spherical harmonics coefficients are not used in our method. The post-activation functions for each parameter follow the design of GS-LRM [22], except for the spatial xy offset, for which we constrain the range to lie within a single pixel of viewpoint resolution. We utilize gsplat [21], an open-source library for Gaussian Splatting [9] for a rasterizer. In the post-prediction optimization, we use a learning rate of $6e^{-4}$ for the positions and $1e^{-3}$ for the other attributes.

Camera pose normalization. We normalize camera poses to align the scene into a consistent coordinate system and scale. First, we compute the average position and viewing directions (forward, down, and right) from the input camera extrinsics. These are used to build a new reference pose, which centers and aligns the scene. All camera extrinsics are then transformed into this reference frame. Finally, we scale the entire scene so that the largest camera distance is 1, ensuring the scene fits within a normalized space [22].

B. Additional Architectural Details

We provide the detailed figure of our token uplifting module in Fig. 1. Note that, to balance the model's representational capacity and computational efficiency, the length of the low-resolution viewpoint embeddings does not exceed that of the high-resolution image features.

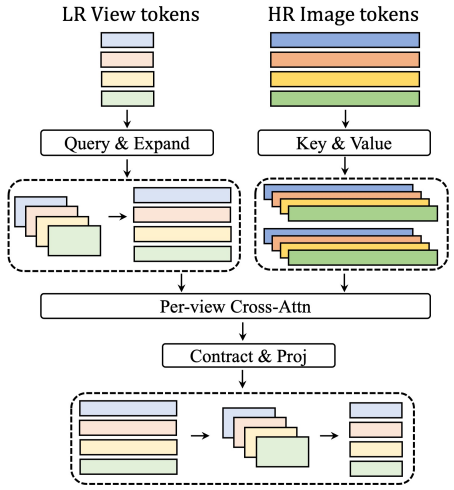

Figure 1. Architectural details of token uplifting.

Tokenization and normalization. After tokenizing the viewpoints and multi-view images using linear layers, both types of tokens are passed through a LayerNorm [2]. In each cross-attention layer, only the viewpoint tokens are further processed with a pre-normalization layer. Additionally, after the query and key linear projections, both tokens are passed through an extra normalization layer, referred to as the QK-Norm [7].

C. Additional Evaluation Details

When we utilize more input viewpoints (more than two in RealEstate10K [23] experiment compared to the baselines), we sample additional viewpoints/images evenly between the two endpoint indices, ensuring that these samples do not overlap with the target indices. For cross-dataset generalization on the DL3DV dataset, we use a baseline of 12 frames.

In wide-baseline setting, every 8th image in the sequence is reserved for the test split, while K-means clustering on camera positions and viewing directions is applied to the remaining images to select input views that ensure wide scene coverage [24].

	PSNR ↑	SSIM ↑	LPIPS ↓
Baseline	**29.24**	**0.907**	**0.109**
w/o self-attention	23.33	0.755	0.220
w/ group-attention	29.02	0.904	0.112
w/ random init.	28.90	0.902	0.112
w/ LR-feature init.	28.35	0.894	0.121

Table 1. Additional ablations on model architecture.

D. Additional Ablations on Model Architecture

We provide additional ablation studies and analyses in Tab. 1 under the same configuration as the ablations on model architecture in main script. All variants are trained under half-resolution 4 viewpoints setting $(4, H, F)$, with a batch size of 16 on a single RTX 4090 GPU.

1) Self-attention. To ensure a fair comparison, we replaced all self-attention layers with cross-attention layers rather than simply removing them, maintaining a comparable parameter count. The performance dropped significantly, highlighting the essential role of self-attention in capturing global dependencies and enhancing multi-view awareness among viewpoint embeddings. Without self-attention, the model struggles to integrate contextual information across different viewpoints, resulting in poor convergence and reconstruction quality.

2) Group-attention This variant replaces the per-viewpoint cross-attention mechanism with a group-attention approach, where all viewpoint tokens and image tokens are concatenated and jointly processed through a cross-attention block. Unlike our default design, group-attention introduces global interactions across all views. While this mechanism can increase the expressive capacity between multiple viewpoints, it incurs quadratic complexity with respect to the number of views. However, the increased computational cost does not yield performance gains, suggesting that separating the roles—using cross-attention for localized image-view interactions and self-attention for global refinement across viewpoints—leads to a more efficient and effective architecture, which is also validated as Alternating-Attention in VGGT [16].

3) Different initialization. We also investigate the different initialization methods of scene representation. For the random initizliation, we used a learnable embedding initialized with zero mean and 0.02 standard deviation, whereas for the LR feature variant, we used features extracted from low-resolution images. In the PSNR training curve, the LR feature variant rises more quickly in the early stages but is later surpassed by the random initialization variant. We believe this is because, in our iterative cross-attention architecture, high-resolution image features and camera information are continually provided by the cross-attention blocks. As a result, the learnable embedding can offer more flexible and informative parameters for guiding iterative updates, whereas the LR image features may introduce redundant and less discriminative information that limits long-term performance gains. Moreover, the use of LR features may bias the early attention stages toward coarse processing, which can hinder the model's ability to fully refine fine details in later stages.

E. Computational Costs of Training

We provide a detailed theoretical calculation of the FLOPs for our mini-batch cross-attention mechanism. In this analysis, we limit the computation to a per-view, single cross-attention operation, excluding our token uplifting strategy (as it introduces a constant cost across all variations). Given a viewpoint token of shape (L_v, D) and an image token of shape (L_i, D), where L_v and L_i denote the token lengths and D is the hidden dimension, the FLOPs for the cross-attention operation are computed as:

$$4D^2(L_v + L_i) + 4L_vL_iD.$$

Assuming a hidden dimension of $D = 768$, an image resolution of 256×256, and a viewpoint resolution of 128×128, with a patch size of 8×8, the token lengths are computed as $L_i = 1024$ for the image tokens and $L_v = 256$ for the viewpoint tokens, based on the experimental configuration used in the RealEstate10K [23] dataset.

Thus, the computation becomes: baseline: **3.83 GFLOPs**; half cross-attention: **1.71 GFLOPs**; quarter cross-attention: **0.81 GFLOPs**.

F. Additional Quantitative Results

Wide-coverage baselines. We additionally evaluate our method on the recently released DL3DV [11] evaluation split, which comprises 51 scenes in our experiments. Our method achieves better reconstruction quality, faster inference, and stronger longer-context generalization, while its

compact 3D scene representations additionally enable fast rendering, as shown in Tab. 2.

Method	Views	Time ↓	PSNR ↑	SSIM ↑	LPIPS ↓
3D-GS [9]	32	8min	25.09	0.838	0.175
Long-LRM [24]	32	0.84sec	23.54	0.776	0.270
Ours	$(32, H, F)$	**0.53sec**	**23.93**	**0.800**	**0.259**
Long-LRM (Unseen)	16	0.50sec	20.65	0.707	0.328
Ours (Unseen)	$(16, H, F)$	**0.19sec**	**21.63**	**0.746**	**0.316**
Long-LRM (Unseen)	40	1.05sec	23.76	0.785	0.262
Ours (Unseen)	$(40, H, F)$	**0.76sec**	**24.21**	**0.809**	**0.250**
Long-LRM (Unseen)	48	1.38sec	23.88	0.795	0.255
Ours (Unseen)	$(48, H, F)$	**1.04sec**	**24.45**	**0.818**	**0.242**

Table 2. Quantitative comparisons on the undistorted DL3DV evaluation dataset (540×960). We utilized flash attention v3 [15] using a H100 GPU.

Depth estimation. In addition to novel view synthesis, we further evaluate the rendered depth maps, which serve as a indicator measure for underlying geometric accuracy, on the DL3DV [11], Tanks&Temples (TNT) [10], and Mip-NeRF360 (Mip360) [3] dataset. Since these datasets do not provide ground-truth depth, we adopt the recent state-of-the-art monocular depth estimator, MoGe-2 [17], as a proxy to obtain pseudo depth. For each target view, we predict depth from the target image and its focal length, mask out invalid values in both rendered and pseudo depths, and then compute relative depth accuracy by comparing them using standard scale-invariant depth metrics. Tab. 3 shows that our method produces depth maps that are more consistent with the MoGe-2 predictions than those of the baseline, even though the baseline is additionally regularized using a pretrained depth estimation model [19] during training. We regard this evaluation as an indirect indicator of geometric quality in the absence of ground-truth depth. We also provide qualitative visualizations of the rendered depth maps in Fig. 2, where our method produces sharper, more detailed depth boundaries and fewer artifacts compared to the baseline. We attribute these improvements to our compact representation, which reduces redundancy and artifacts while better preserving fine geometric details.

Method	Views	DL3DV-Benchmark		DL3DV-Eval		TNT&Mip360	
		Abs Rel ↓	RMSE ↓	Abs Rel ↓	RMSE ↓	Abs Rel ↓	RMSE ↓
Long-LRM [24]	16	0.718	1.427	0.768	1.631	0.603	1.199
Ours	$(16, H, F)$	**0.670**	**1.174**	**0.693**	**1.354**	**0.515**	**0.852**
Long-LRM	32	0.759	1.571	0.805	1.788	0.645	1.345
Ours	$(32, H, F)$	**0.709**	**1.310**	**0.739**	**1.513**	**0.543**	**0.967**

Table 3. Quantitative comparison of depth estimation. We use the large MoGe-2 (ViT-L) variant as the pseudo-depth estimator.

Geometry estimation We evaluate the Chamfer Distance (CD) and F1-score with geometry datasets using NRGBD [1] and ETH3D [14]. For iLRM, ground-truth pointmaps are downsampled by half to match the generated point clouds. With Mip-NeRF360 dataset, we visualized the vertices after building mesh and removing flying points

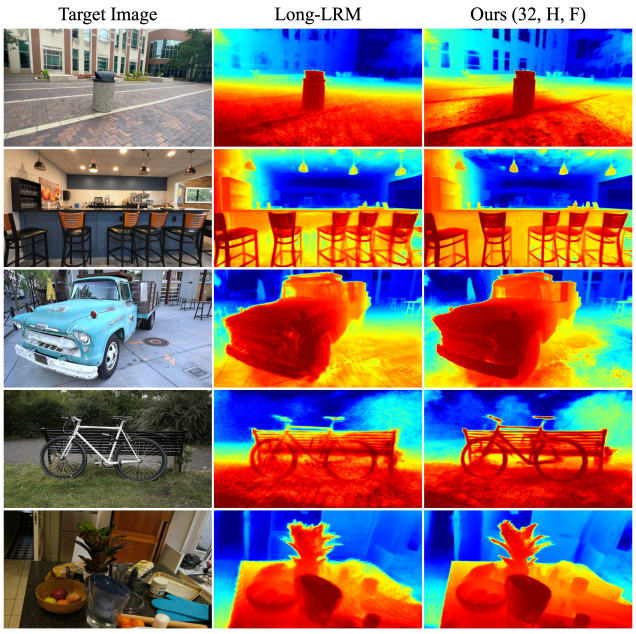

Figure 2. Qualitative comparison of rendered depth maps. Examples from DL3DV (top two rows), Tanks&Temples (third row), and Mip-NeRF360 (bottom two rows) are shown.

(relative tolerance threshold of 0.04). iLRM shows better geometry with fewer points in Tab. 4 and Fig. 3.

Method	NRGBD (16-view)		NRGBD (32-view)		ETH3D (16-view)		ETH3D (32-view)	
	CD ↓	F1-score ↑	CD ↓	F1-score ↑	CD ↓	F1-score ↑	CD ↓	F1-score ↑
Long-LRM	0.53	0.52	**0.43**	0.59	2.75	0.32	2.69	0.39
Ours	**0.50**	**0.60**	0.52	**0.69**	**2.06**	**0.50**	**1.15**	**0.54**

Table 4. Pointmap estimation comparisons with a input resolution of 540 × 960 (# views). The F1-score threshold was set to 0.1.

Figure 3. Zero-shot colored vertices visualization.

Post-prediction optimization While the performance of zero-shot novel view synthesis (TNT, Mip360) in Tab. 5 is comparable to the baseline, our finer geometric estimation promote faster convergence during post-prediction optimization which reflects more reliable geometry in the initial 3D Gaussians. In 10-epoch optimization, our method already outperforms the baseline while using less than half

of the time, and with 20-epoch, still faster than the baseline, it achieves even higher accuracy. We attribute these gains to our compact scene representation and its stronger capacity for capturing underlying geometric structure. We provide qualitative comparisons on 10-epoch optimization in Fig. 4.

Method	Views	Time ↓	Tanks&Temples [10]			Mip-NeRF360 [3]		
			PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
3D-GS [9]	32	8min	18.48	0.720	0.260	22.95	0.694	0.250
Long-LRM [24]	32	0.84sec	**18.59**	0.614	**0.367**	21.08	0.484	**0.445**
Ours	$(32, H, F)$	**0.53sec**	18.58	**0.631**	0.385	**21.09**	**0.495**	0.466
Long-LRM_{10}	32	11sec	19.23	0.663	0.348	22.05	0.554	0.414
Ours_{10}	$(32, H, F)$	**4.5sec**	19.42	0.689	0.350	22.49	0.601	0.414
Ours_{20}	$(32, H, F)$	8.6sec	**19.62**	**0.704**	**0.338**	**22.85**	**0.622**	**0.397**

Table 5. Quantitative comparisons on the Tanks&Temples and Mip-NeRF360 dataset (540×960). We adopt the settings reported in their paper for the post-prediction optimization of Long-LRM (learning rates of $5e^{-4}$ for position and $1e^{-3}$ for color). We utilized flash attention v3 [15] using a H100 GPU.

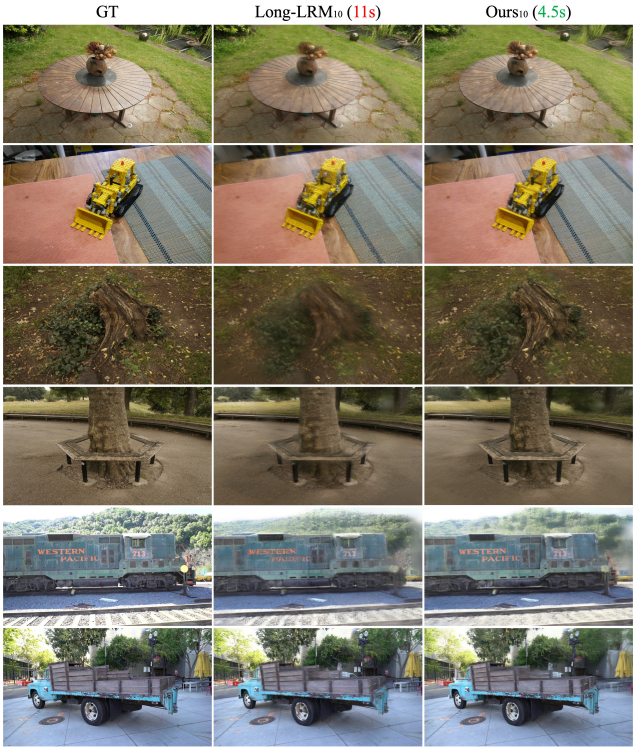

Figure 4. Qualitative comparison on Mip-NeRF360 (first four rows) and Tanks&Temples (bottom two rows) after 10-epoch of post-prediction optimization.

Input robustness. To assess the robustness of our model to imperfect camera pose estimates, we evaluate under translational camera pose perturbations on the 32-view DL3DV dataset. Gaussian noise is added to the camera translation vectors, and we report performance across varying noise to examine the degradation in rendering quality (Tab. 6).

Inference time. We compare the inference time of our model across different numbers of input views at a resolution of 540×960, using a single H100 GPU with flash attention v3 [15]. As shown in Tab. 7, our method achieves lower latency than Long-LRM across all comparable settings. Notably, Long-LRM runs out of memory at 256 input views, whereas our model still completes inference within a practical time budget, indicating better scalability of the proposed compact viewpoint representation to large number of views.

stand. dev.	0		0.001		0.005	
	PSNR	LPIPS	PSNR	LPIPS	PSNR	LPIPS
Long-LRM	23.97	0.267	23.27 (-0.70)	0.287 (+0.020)	20.49 (-3.48)	0.381 (+0.114)
iLRM	**24.30**	**0.256**	**23.82 (-0.48)**	**0.270 (+0.014)**	**21.49 (-2.81)**	**0.352 (+0.096)**

Table 6. Robustness evaluation under translational camera pose perturbations on the 32-view DL3DV (540×960) dataset.

Method	16	32	64	96	128	256
Long-LRM [24]	0.5	0.84	2.08	3.90	6.39	Out-of-memory
Ours	**0.19**	**0.53**	**1.66**	**3.37**	**5.61**	**20.92**

Table 7. Quantitative comparisons of inference time across different numbers of input views. All times are measured in seconds.

Varying baseline range. We compare our model against recent generalizable 3D reconstruction methods [5, 13, 18] on the RealEstate10K [23] dataset, with a particular focus on handling varying degrees of camera overlap [20]. These overlap categories are determined using the dense feature matching method RoMA [6]. As shown in Tab. 8, our method, which efficiently handles a large number of input viewpoints/images, achieves superior performance compared to existing approaches, especially in challenging cases with small viewpoint overlap.

Same number of Gaussians. We also validate the strength of our decoupling strategy in leveraging high-resolution images as visual cues while generating efficient and compact 3D Gaussians. As discussed in our motivation, previous methods [4, 5, 13, 18, 22] require downsampling the input images to reduce the number of generated Gaussians, inherently coupling image resolution with representation density. To demonstrate the flexibility of our approach, we conduct an experiment in which all methods generate the same number of Gaussians using 4 viewpoints at half resolution. Specifically, the baseline methods [5, 18] follow a $(4, H, H)$ configuration, where both the number of viewpoints and image resolution are reduced. In contrast, our method adopts a $(4, H, F)$ setting, where we preserve high-resolution image inputs while generating low-resolution Gaussians, thanks to our decoupled design. As shown in Tab. 9, our method surpasses the baselines in performance while requiring fewer computational resources in training, and faster inference speed, highlighting the practical advantages of our design. This result demonstrates the efficiency and the representational ability of our architecture, which effectively utilizes high-resolution visual cues, leading to superior reconstruction quality under the same output density without requiring

Method	Small			Medium			Large			Average		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
MVSplat [5]	20.37	0.725	0.250	23.81	0.814	0.172	27.47	0.885	0.115	24.01	0.812	0.175
DepthSplat [18]	22.82	0.798	0.193	25.38	0.851	0.145	28.32	0.900	0.104	25.59	0.852	0.145
Gen-Den [13]	21.10	0.744	0.234	24.57	0.828	0.162	28.26	0.895	0.108	24.77	0.826	0.164
Ours $(2, F, F)$	**23.82**	**0.813**	**0.184**	**26.54**	**0.864**	**0.139**	**29.43**	**0.910**	**0.103**	**26.70**	**0.864**	**0.140**
Ours $(4, H, F)$	27.65	0.887	0.127	29.13	0.908	0.108	30.73	0.926	0.092	29.22	0.908	0.108
Ours-MC $(4, H, F)$	27.41	0.882	0.131	28.87	0.904	0.111	30.44	0.927	0.095	28.96	0.904	0.112
Ours $(8, H, F)$	29.44	0.912	0.106	30.51	0.925	0.093	31.77	0.937	0.080	30.61	0.925	0.092
Ours-MC $(8, H, F)$	29.15	0.908	0.108	30.20	0.922	0.094	31.46	0.935	0.082	30.30	0.922	0.094

Table 8. Quantitative comparisons on the RE10K dataset under varying view overlap conditions.

Method	Params (M)	Train GPU (#)	PSNR ↑	SSIM ↑	LPIPS ↓	# Gaussians	Time (s)	Memory (GB)
MVSplat [5]	12	H100 (1)	27.53	0.889	0.116	65,536	0.048	**0.65**
DepthSplat [18]	354	H100 (1)	28.08	0.898	0.107	65,536	0.062	2.49
Ours	185	RTX 4090 (1)	29.24	0.907	0.109	65,536	**0.027**	1.22
Ours	185	RTX 4090 (2)	**29.82**	**0.916**	**0.101**	65,536	**0.027**	1.22

Table 9. Quantitative comparisons under the same number of Gaussians on the RE10K dataset. Inference time and memory consumption are measured only during the Gaussian generation stage, excluding the rendering process on a RTX 4090 GPU.

expensive hardware. To train the baseline methods effectively with a large batch size (similar to ours), we run them on a single H100 GPU. Our method and MVSplat [5] are trained with a batch size of 16, while DepthSplat [18] is trained with a batch size of 12 due to memory constraints.

Quarter resolution baselines. To evaluate different viewpoint configurations—specifically the resolution of each viewpoint—we additionally compare a quarter-resolution variant of the viewpoint inputs. All experiments in Tab. 10 are conducted using a single RTX 4090 GPU with a batch size of 16, 12 update layers, and trained for 100,000 iterations. While lowering the resolution of viewpoint inputs leads to a moderate drop in reconstruction quality, it significantly reduces the number of generated Gaussians, showing trade-off between accuracy and efficient representations.

Method	PSNR ↑	SSIM ↑	LPIPS ↓	# Gaussians
Ours $(8, H, F)$	**30.39**	**0.923**	**0.095**	131,072
Ours $(4, H, F)$	29.24	0.907	0.109	65,536
Ours $(8, Q, F)$	27.36	0.868	0.152	32,768
Ours $(4, Q, F)$	26.40	0.843	0.177	**16,384**

Table 10. Quantitative comparisons of different viewpoint configurations on the RE10K dataset. Q denotes quarter resolution compared to the original image resolution.

G. Additional Qualitative Results

We present additional qualitative results in Fig. 5 for the RealEstate10K (RE10K) [23] dataset and in Fig. 6, 7, 8 and 9 for the DL3DV [11], Tanks&Temples [10], and MipNeRF360 [3] dataset. Also, we provide additional attention visualization in Fig. 10. Further details for each example are provided in the corresponding captions in figures.

Figure 5. Qualitative comparison on the RE10K dataset (2 input images except for "Ours(4, H, F)" and "Ours(8, H, F)", 256×256).

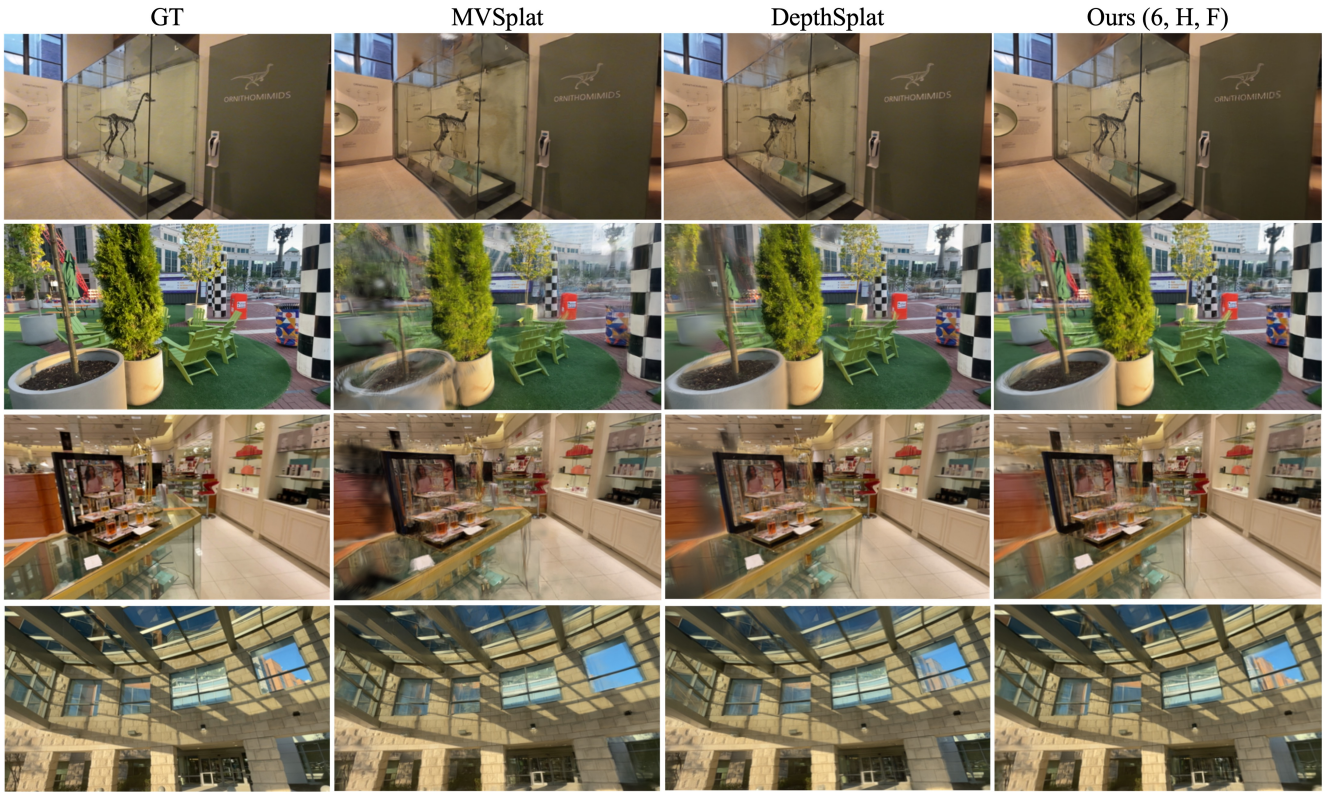

Figure 6. Qualitative comparison on the DL3DV dataset under the 50-frame baseline setting (6 input images, 256×448).

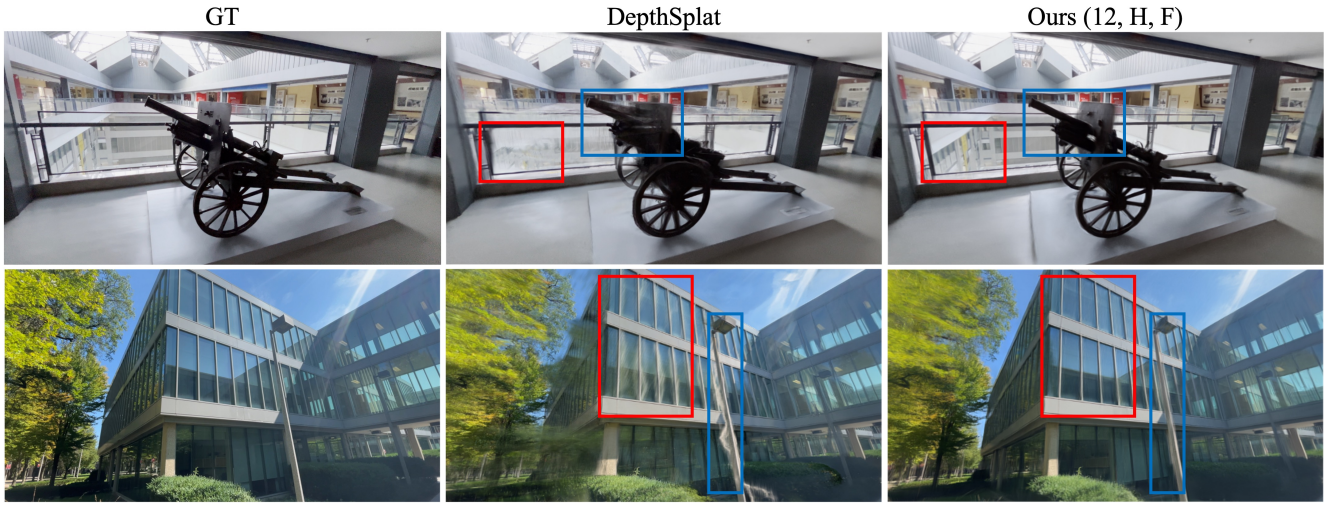

Figure 7. Qualitative comparison on the DL3DV dataset under the 100-frame baseline setting (12 input images, 512×960).

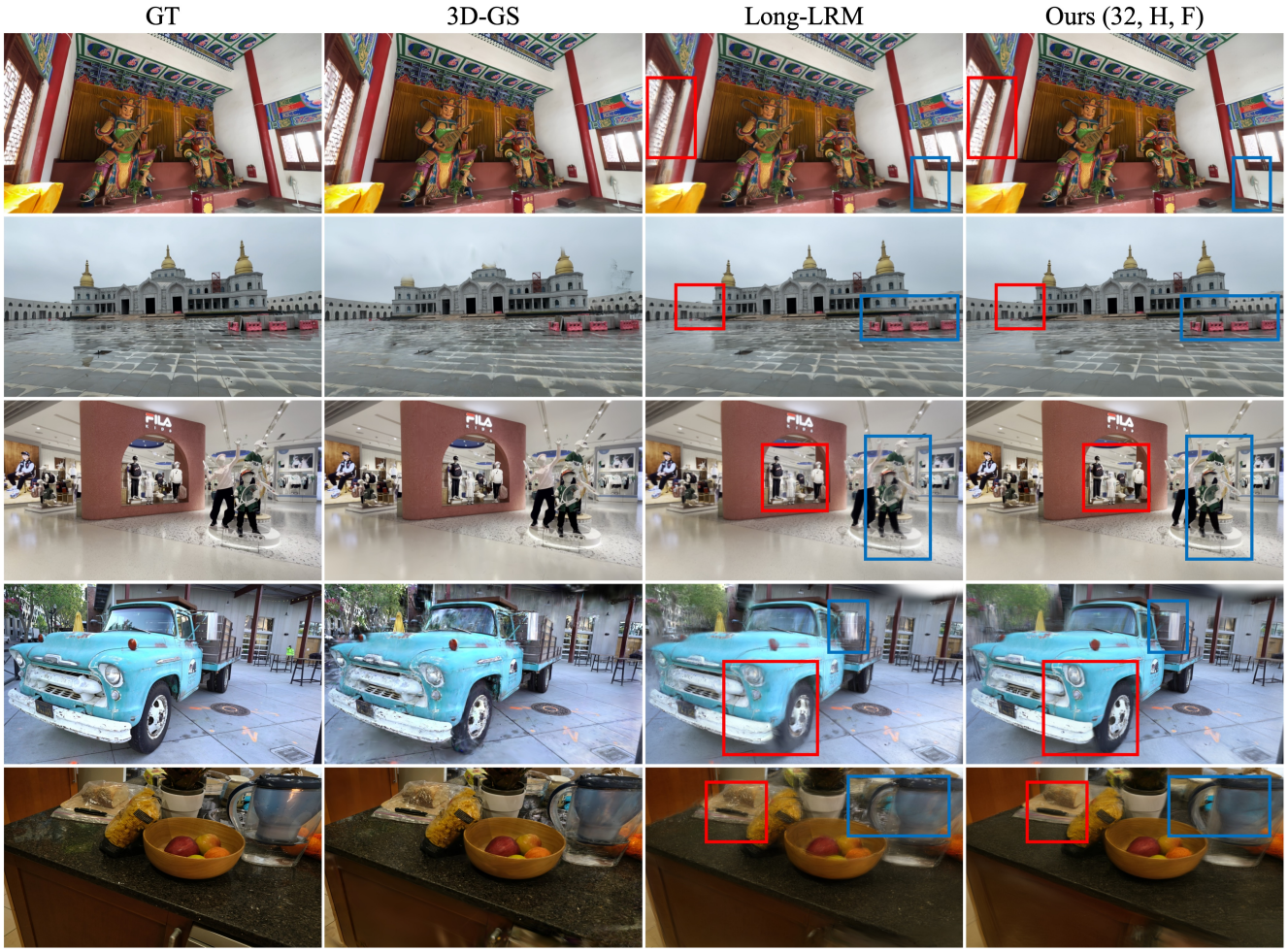

Figure 8. Qualitative comparison under the wide-baseline setting (32 input images, 540×960, zero-shot). DL3DV (top three rows), Tanks&Temples (fourth row), and Mip-NeRF360 (bottom row) are shown.

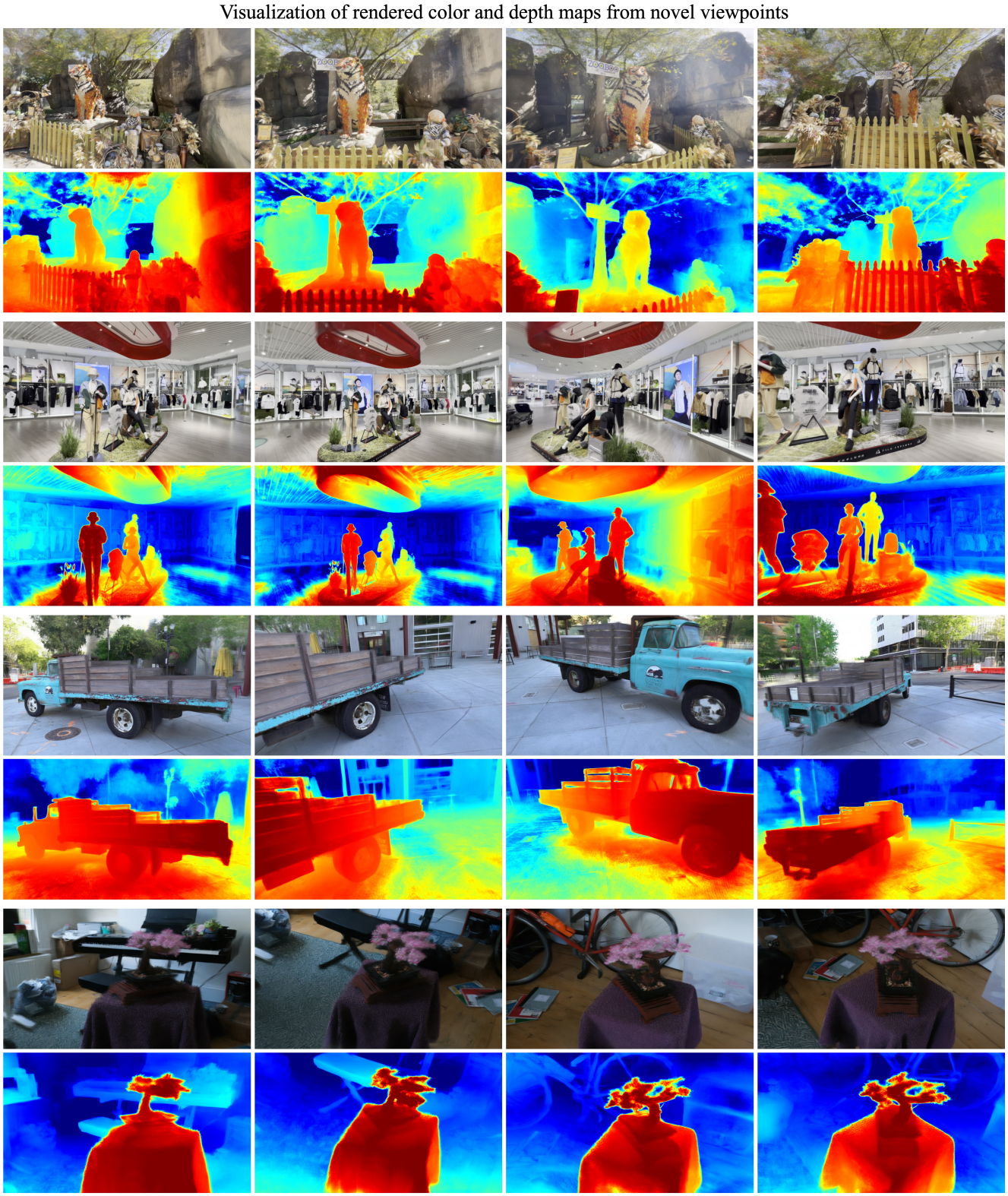

Figure 9. Qualitative visualization of rendered color and depth maps from novel viewpoints (32 input images, 540×960, zero-shot). Scenes from DL3DV (top two example), Tanks&Temples (third example), and Mip-NeRF360 (bottom example) are shown.

Figure 10. For the colored query patches in the reference viewpoint (red, yellow, green), we visualize top-3 attended tokens from other viewpoints throughout the iterative refinement process. For relatively easy cases with small camera motion or distinctive regions, the model identifies the correct correspondences in the early layers, whereas for more challenging cases with larger viewpoint changes, the attention gradually converges to geometrically consistent regions as the refinement progresses. Scenes from DL3DV (top two rows), Tanks&Temples (third row), and Mip-NeRF360 (bottom row) are shown.

References

[1] Dejan Azinović, Ricardo Martin-Brualla, Dan B Goldman, Matthias Nießner, and Justus Thies. Neural rgb-d surface reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6290–6301, 2022. 3

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 1

[3] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5470–5479, 2022. 3, 4, 5

[4] David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. 4

[5] Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images. In *European Conference on Computer Vision*, 2025. 1, 4, 5

[6] Johan Edstedt, Qiyu Sun, Georg Bökman, Mårten Wadenbäck, and Michael Felsberg. Roma: Robust dense feature matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19790–19800, 2024. 4

[7] Alex Henry, Prudhvi Raj Dachapally, Shubham Pawar, and Yuxuan Chen. Query-key normalization for transformers. *arXiv preprint arXiv:2010.04245*, 2020. 1

[8] Haian Jin, Hanwen Jiang, Hao Tan, Kai Zhang, Sai Bi, Tianyuan Zhang, Fujun Luan, Noah Snavely, and Zexiang Xu. Lvsm: A large view synthesis model with minimal 3d inductive bias. *arXiv preprint arXiv:2410.17242*, 2024. 1

[9] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 2023. 1, 3, 4

[10] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36(4), 2017. 3, 4, 5

[11] Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, et al. Dl3dv-10k: A large-scale scene dataset for deep learning-based 3d vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. 1, 2, 3, 5

[12] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 1

[13] Seungtae Nam, Xiangyu Sun, Gyeongjin Kang, Younggeun Lee, Seungjun Oh, and Eunbyung Park. Generative densification: Learning to densify gaussians for high-fidelity generalizable 3d reconstruction. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 26683–26693, 2025. 4, 5

[14] 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 *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2538–2547, 2017. 3

[15] Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. Flashattention-3: Fast and accurate attention with asynchrony and low-precision. *Advances in Neural Information Processing Systems*, 37: 68658–68685, 2024. 3, 4

[16] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5294–5306, 2025. 2

[17] Ruicheng Wang, Sicheng Xu, Yue Dong, Yu Deng, Jianfeng Xiang, Zelong Lv, Guangzhong Sun, Xin Tong, and Jiaolong Yang. Moge-2: Accurate monocular geometry with metric scale and sharp details. *arXiv preprint arXiv:2507.02546*, 2025. 3

[18] Haofei Xu, Songyou Peng, Fangjinhua Wang, Hermann Blum, Daniel Barath, Andreas Geiger, and Marc Pollefeys. Depthsplat: Connecting gaussian splatting and depth. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 16453–16463, 2025. 1, 4, 5

[19] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. 3

[20] Botao Ye, Sifei Liu, Haofei Xu, Xueting Li, Marc Pollefeys, Ming-Hsuan Yang, and Songyou Peng. No pose, no problem: Surprisingly simple 3d gaussian splats from sparse unposed images. *arXiv preprint arXiv:2410.24207*, 2024. 4

[21] Vickie Ye, Ruilong Li, Justin Kerr, Matias Turkulainen, Brent Yi, Zhuoyang Pan, Otto Seiskari, Jianbo Ye, Jeffrey Hu, Matthew Tancik, et al. gsplat: An open-source library for gaussian splatting. *Journal of Machine Learning Research*, 26(34):1–17, 2025. 1

[22] Kai Zhang, Sai Bi, Hao Tan, Yuanbo Xiangli, Nanxuan Zhao, Kalyan Sunkavalli, and Zexiang Xu. Gs-lrm: Large reconstruction model for 3d gaussian splatting. In *European Conference on Computer Vision*, 2025. 1, 4

[23] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *arXiv preprint arXiv:1805.09817*, 2018. 1, 2, 4, 5

[24] Chen Ziwen, Hao Tan, Kai Zhang, Sai Bi, Fujun Luan, Yicong Hong, Li Fuxin, and Zexiang Xu. Long-lrm: Long-sequence large reconstruction model for wide-coverage gaussian splats. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025. 1, 2, 3, 4