HiMoR: Monocular Deformable Gaussian Reconstruction with Hierarchical Motion Representation

Supplementary Material

1. Additional results

iPhone dataset. We show qualitative comparison of temporal consistency at novel view on the iPhone dataset [2] in Fig. 4, 5, 6, 7. Our method demonstrates better overall temporal consistency.

Nvidia dataset. We show per-scene quantitative results of novel view synthesis on the Nvidia dataset [8] in Tab. 1, and qualitative results in Fig. 1.

Ablation studies. We show qualitative results of ablation studies in Fig. 2.

2. Additional training details

2.1. Loss functions and weights

Here, we provide a detailed explanation for each term of the loss in main text Eq. 7.

RGB loss $\mathcal{L}_{rgb}$, mask loss $\mathcal{L}_{mask}$, and depth loss $\mathcal{L}_{depth}$, ensure that the rendered image, foreground mask, and depth to match their respective ground truth in a pixel-wise manner. The RGB loss $\mathcal{L}_{rgb}$ is a combined MSE (mean squared error) loss, D-SSIM [11] loss, and LPIPS [15] loss between the rendered image and the ground truth, weighted 0.8, 0.2, and 0.01, respectively. Mask loss $\mathcal{L}_{mask}$ computes MSE between rendered mask and the mask predicted via [12] with weight of 1.0. For the depth loss $\mathcal{L}_{depth}$, we include an MSE term weighted 0.5, and following [10], apply a regularization to the gradient of the rendered depth, weighted 1.0.

The tracking loss $\mathcal{L}_{track}$ supervises the rendered tracks to match the unprojected 2D tracks from [1]. Following [10], we compute $\mathcal{L}_{track}$ as a combination of $\mathcal{L}_{track-2d}$ and $\mathcal{L}_{track-depth}$, with respective weights of 2.0 and 0.1. $\mathcal{L}_{track-2d}$ is the MSE loss between between rendered 2D tracks and the 2D tracks from [1] on normalized pixel coordinates, while $\mathcal{L}_{track-depth}$ is the MSE loss between the reprojected depths of the rendered tracks and the metric aligned depths from [13].

The rigidity loss $\mathcal{L}_{rigid}$ is calculated as:

$$\begin{aligned}\mathcal{L}_{rigid} =& \\ &\sum_{i\in\mathcal{N}}\sum_{j\in\mathrm{KNN}(i)}\big(\big|\|\boldsymbol{x}_{i,t}-\boldsymbol{x}_{j,t}\|-\|\boldsymbol{x}_{i,t+\Delta}-\boldsymbol{x}_{j,t+\Delta}\|\big| \\ &+\|\mathbf{T}_{j,t}^{-1}\boldsymbol{x}_{i,t}-\mathbf{T}_{j,t+\Delta}^{-1}\boldsymbol{x}_{i,t+\Delta}\|\big),\end{aligned} \tag{1}$$

where $\boldsymbol{x}_{*,t}$ and $\mathbf{T}_{*,t}$ denote node's position at time t and its deformation from the canonical frame to time t, respectively. Δ denotes time interval within the sampled batch. The set of nodes is denoted by $\mathcal{N}$, while $\mathrm{KNN}(i)$ refers to the K-nearest neighbors of the i^{th} node, determined by curve distance. The initial weight of $\mathcal{L}_{rigid}$ for first-level nodes is 0.5. Once second-level nodes are activated, the weight for first-level nodes is increased to 2.5, while second-level nodes are assigned a weight of 0.5.

We also include regularization terms for the acceleration of motion bases, the acceleration of rendered tracks, and the scale of Gaussians as [10], with respective weights of 0.1, 2.0, and 0.01. In addition, the radius of nodes are regularized to be no larger than the average distance to its three nearest neighbors. The weight of this regularization is set to 0.0001.

2.2. Optimization

We use Adam [4] to optimize HiMoR and Gaussians in canonical frame jointly. The learning rates for mean, opacity, scale, rotation, and color of each Gaussian are set to 1.6×10^{-4}, 1×10^{-2}, 5×10^{-3}, 1×10^{-3}, and 1×10^{-2}, respectively. The adaptive density control of Gaussians proposed in original 3DGS paper [3] is also applied. The learning rate for motion bases is set to 1.6×10^{-4}. The learning rates for position, radius, and motion coefficients of each node are set to 1.6×10^{-5}, 5×10^{-4}, and 1×10^{-2}, respectively. We train our method using a single V100 GPU with 32GB of VRAM. The total training time is approximately $3-6$ hours for the iPhone dataset [2], and about 1 hour for the Nvidia dataset [8].

3. Evaluation details

For iPhone dataset [2], we use the preprocessed dataset provided by [10]. For Nvidia dataset [8], we use scripts provided by [10] to generate foreground masks, 2D tracks, and monocular depth maps. The monocular depth maps are aligned with per-frame scale and shift factor computed using the point cloud estimated by running COLMAP [7] on all 12 calibrated cameras as [9]. The results of each baseline are produced using the respective public code with its original settings.

4. Formulation of CLIP-I and CLIP-T

CLIP-I is the cosine similarity between the CLIP embeddings [6] of the rendered image and the ground truth, while CLIP-T is the cosine similarity between frames with certain interval to assess temporal consistency. CLIP-I and CLIP-T

Method	Balloon1			Balloon2			Jumping			Playground		
	PSNR ↑	SSIM↑	LPIPS ↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
T-NeRF [2]	23.152	0.7498	0.11402	23.475	0.8322	0.09172	20.011	0.6922	0.15557	16.902	0.5438	0.20163
HyperNeRF [5]	22.576	0.7354	0.09916	23.933	0.8501	0.07770	20.279	0.7148	0.14790	16.521	0.5283	0.20321
Deformable 3DGS [14]	15.912	0.2753	0.46370	15.126	0.3071	0.41394	16.685	0.4525	0.34131	12.878	0.3193	0.37822
Marbles [9]	23.377	0.7843	0.07810	23.422	0.8046	0.08843	19.997	0.6545	0.14920	16.944	0.5757	0.15507
SoM [10]	23.692	0.7919	0.06336	23.037	0.8155	0.07606	19.906	0.6692	0.17004	16.976	0.5745	0.15110
Ours	23.901	0.7978	0.06200	23.483	0.8261	0.07540	20.042	0.6910	0.15978	17.125	0.5784	0.14942

Method	Skating			Truck			Umbrella			Mean		
	PSNR ↑	SSIM↑	LPIPS ↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
T-NeRF [2]	27.177	0.8937	0.05666	27.363	0.8588	0.06407	24.609	0.6528	0.11780	23.241	0.7462	0.11450
HyperNeRF [5]	27.085	0.9080	0.05555	27.632	0.8663	0.06536	24.631	0.6584	0.10668	23.237	0.7516	0.10794
Deformable 3DGS [14]	19.310	0.5504	0.27465	18.209	0.4334	0.18731	17.263	0.2807	0.31782	16.483	0.3741	0.33956
Marbles [9]	27.488	0.8905	0.05698	27.127	0.8580	0.05558	24.309	0.6714	0.08876	23.238	0.7541	0.09603
SoM [10]	27.530	0.9141	0.05514	27.569	0.8738	0.04830	24.318	0.6567	0.09221	23.290	0.7565	0.09374
Ours	27.561	0.9151	0.05803	27.400	0.8761	0.04862	24.295	0.6504	0.10269	23.401	0.7621	0.09370

Table 1. Quantitative results of novel view synthesis on the Nvidia dataset [8].

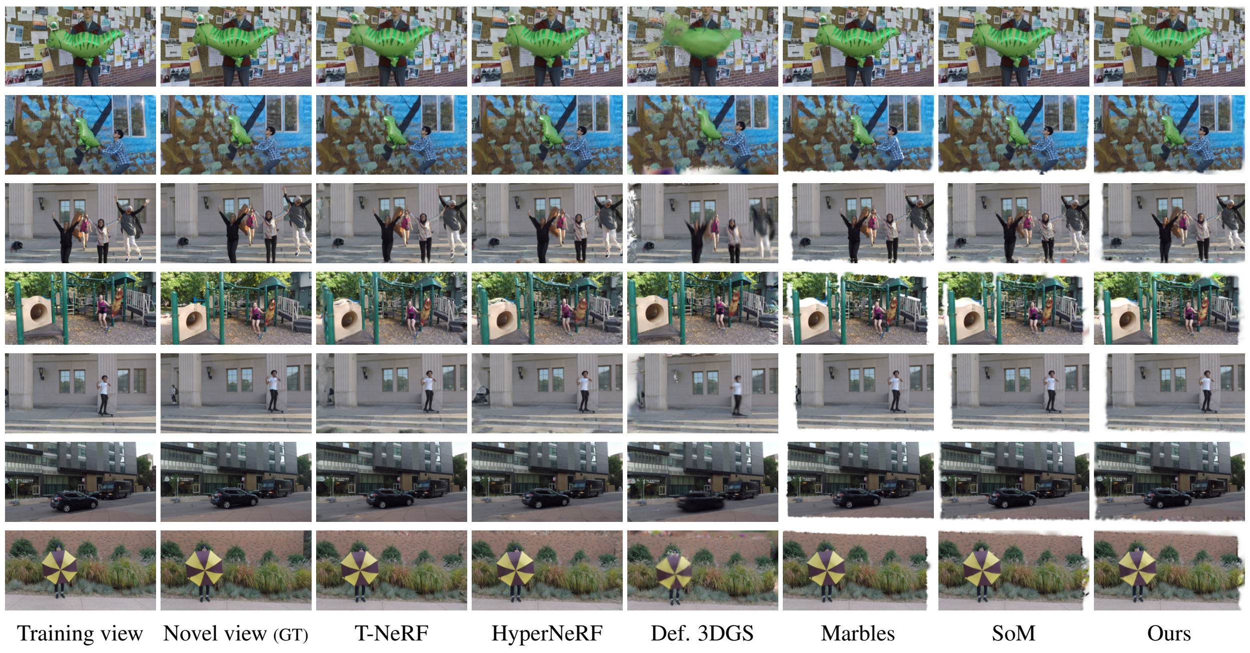

Figure 1. Qualitative results of novel view synthesis on the Nvidia dataset [8]. From the top are "Balloon1", "Balloon2", "Jumping", "Playground", "Skating", "Truck", and "Umbrella".

are defined as follows:

$$\mathrm{Sim}\left(\boldsymbol{v}_i, \boldsymbol{v}_j\right) = \frac{\boldsymbol{v}_i \cdot \boldsymbol{v}_j}{\|\boldsymbol{v}_i\| \|\boldsymbol{v}_j\|}, \tag{2}$$

$$\mathrm{CLIP-I} = \mathrm{Sim}\left(\mathcal{E}(\hat{I}_t), \mathcal{E}(I_t)\right), \tag{3}$$

$$\mathrm{CLIP-T} = \mathrm{Sim}\left(\mathcal{E}(\hat{I}_t), \mathcal{E}(\hat{I}_{t+\Delta})\right), \tag{4}$$

where $\mathcal{E}(\cdot)$ denotes the CLIP encoder, I_t denotes the ground truth image, and $\hat{I}_t$ refers to the rendered image. Δ specifies the temporal interval between frames for CLIP-T.

5. Additional ablations

We provide additional ablations studies on the number of motion bases and the number of nodes, and an explanation about the number of levels. In our default setting "[10, 5] (ours)", the initial number of first-level nodes is 50, which increases to approximately 130 after densification. The number of motion bases shared among first-level nodes is set to 10. When activating second-level nodes, 10 nodes are assigned to each first-level node and 5 motion bases are

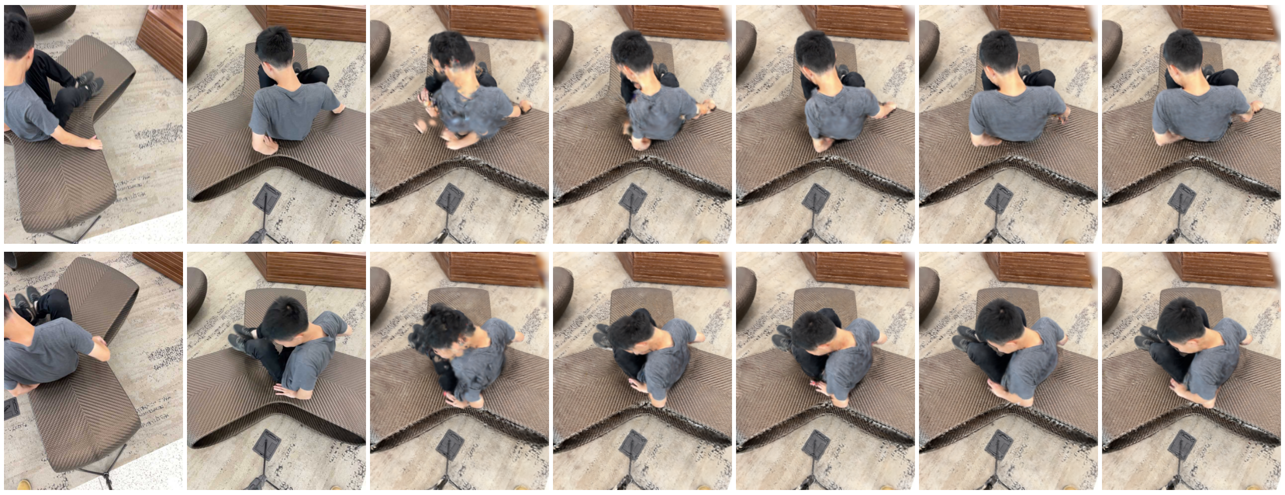

Figure 2. Qualitative results of ablation studies.

Method	CLIP-I↑	CLIP-T↑	LPIPS ↓	PCK-T↑
[20, 5]	0.8828	0.9654	0.3627	0.9147
[5, 5]	0.8842	0.9639	0.3696	0.8910
[10, 8]	0.8909	0.9640	0.3667	0.9171
[10, 2]	0.8795	0.9646	0.3675	0.9068
[10, 5] (half)	0.8853	0.9656	0.3664	0.8971
[10, 5] (double)	0.8824	0.9638	0.3646	0.9201
[10, 5] (ours)	0.8853	0.9658	0.3696	0.9158

Table 2. Ablation study on the number of motion bases and nodes. The values inside "[]" represent the number of motion bases for first-level and second-level nodes, respectively. For example, "[10, 5]" indicates that first-level nodes share 10 motion bases, while second-level nodes under the same parent share 5 motion bases. "(half)" denotes that the number of first-level nodes is halved compared to "(ours)," while "(double)" denotes that the number of first-level nodes is doubled compared to "(ours)." Note that, the number of second-level nodes assigned to each first-level node is unchanged.

shared among those nodes under the same parent node.

Number of motion bases. We evaluate the sensitivity of performance to the number of motion bases. The results are shown in the first four rows of Tab. 2. Increasing the number of motion bases for first-level nodes has minor impact on overall performance, while reducing motion bases for first-level nodes leads to a loss of detail in the training views, as shown in Fig. 3. For second-level nodes, using more motion bases improves the quality of novel view synthesis increased but compromises temporal consistency. When the motion bases for second-level nodes is fewer, it can introduce some subtle distortions, resulting in a lower CLIP-I.

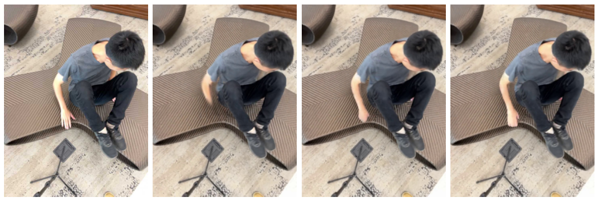

Figure 3. Qualitative results of additional ablation studies on the number of motion bases and nodes. Using fewer motion bases at the first level or fewer nodes in total result in a loss of detail in the training view (*i.e.*, hand).

Based on the above analysis, we find our default setting to be reasonable.

Number of nodes. We vary the number of first-level nodes to exam the impact of the number of nodes. As predefined number of second-level nodes and motion bases are assigned to each first-level nodes, the number of first-level nodes determines the complexity of HiMoR. The results are shown in "[10, 5] (half)" and "[10, 5] (double)" of Tab. 2When the number of first-level nodes is halved compared to original settings (ours), PCK-T decreases, suggesting a lost of details in the training views as Fig. 3. Conversely, doubling the number of first-level nodes results in reduced CLIP-I and CLIP-T, likely due to the increased degrees of freedom.

Number of levels. We only use two levels of nodes in our experiment due to the limited number of foreground Gaussian. Current child nodes initialization strategy involves performing K-Means clustering to the relative deformation of the Gaussians within each parent node's radius. However, when the number of Gaussian inside parent node's radius is limited (*i.e.*, less than the number of K-Means clusters), current child nodes initialization strategy may fail. Moreover, increasing the quantity of nodes might not always prove beneficial as the number of Gaussians per node can be significantly small, making the node-based deformation close to per-Gaussian deformation. However, from the algorithmic perspective, we believe that the additional levels of nodes can become more effective in larger scale scenarios or more complex movements.

References

[1] Carl Doersch, Yi Yang, Mel Vecerik, Dilara Gokay, Ankush Gupta, Yusuf Aytar, Joao Carreira, and Andrew Zisserman. TAPIR: Tracking any point with per-frame initialization and temporal refinement. In *CVPR*, pages 10061–10072, 2023. 1

[2] Hang Gao, Ruilong Li, Shubham Tulsiani, Bryan Russell, and Angjoo Kanazawa. Monocular dynamic view synthesis: A reality check. In *NeurIPS*, 2022. 1, 2, 5, 6, 7, 8

[3] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4), 2023. 1

[4] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, San Diega, CA, USA, 2015. 1

[5] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M. Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *ACM Trans. Graph.*, 40(6), 2021. 2

[6] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763, 2022. 1

[7] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016. 1

[8] Jae Shin Yoon, Kihwan Kim, Orazio Gallo, Hyun Soo Park, and Jan Kautz. Novel View Synthesis of Dynamic Scenes With Globally Coherent Depths From a Monocular Camera . In *CVPR*, pages 5335–5344, 2020. 1, 2

[9] Colton Stearns, Adam W Harley, Mikaela Uy, Florian Dubost, Federico Tombari, Gordon Wetzstein, and Leonidas Guibas. Dynamic gaussian marbles for novel view synthesis of casual monocular videos. In *SIGGRAPH Asia 2024 Conference Papers*, pages 1–11, 2024. 1, 2

[10] Qianqian Wang, Vickie Ye, Hang Gao, Jake Austin, Zhengqi Li, and Angjoo Kanazawa. Shape of motion: 4d reconstruction from a single video. *arXiv preprint arXiv:2407.13764*, 2024. 1, 2

[11] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 1

[12] Jinyu Yang, Mingqi Gao, Zhe Li, Shang Gao, Fangjing Wang, and Feng Zheng. Track anything: Segment anything meets videos. *arXiv preprint arXiv:2304.11968*, 2023. 1

[13] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. In *CVPR*, 2024. 1

[14] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In *CVPR*, 2024. 2

[15] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 1

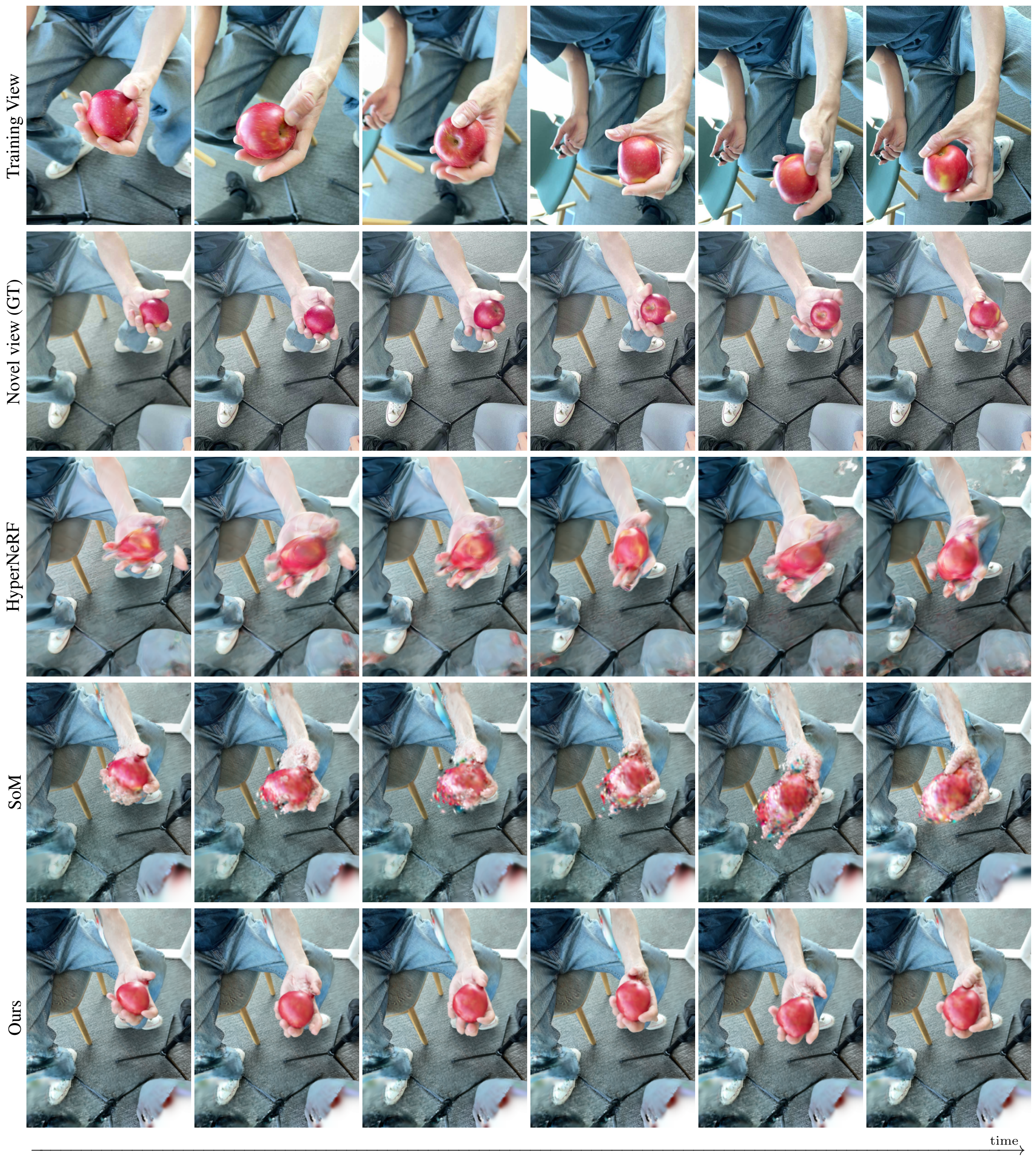

Figure 4. Qualitative comparison of temporal consistency at novel view on the scene “Apple” of iPhone dataset [2]. The time interval of adjacent images is ten frames.

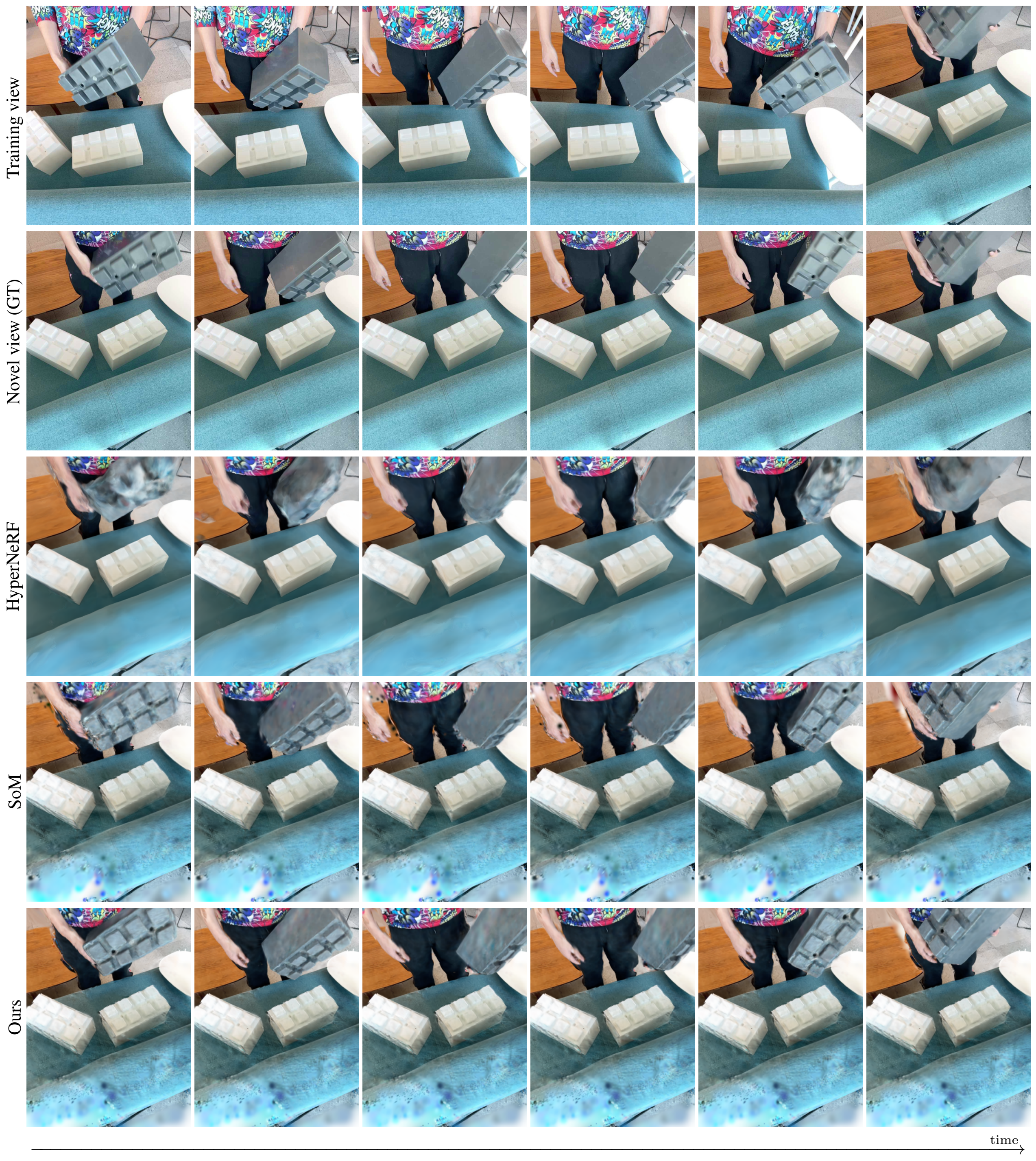

Figure 5. Qualitative comparison of temporal consistency at novel view on the scene “Block” of iPhone dataset [2]. The time interval of adjacent images is ten frames.

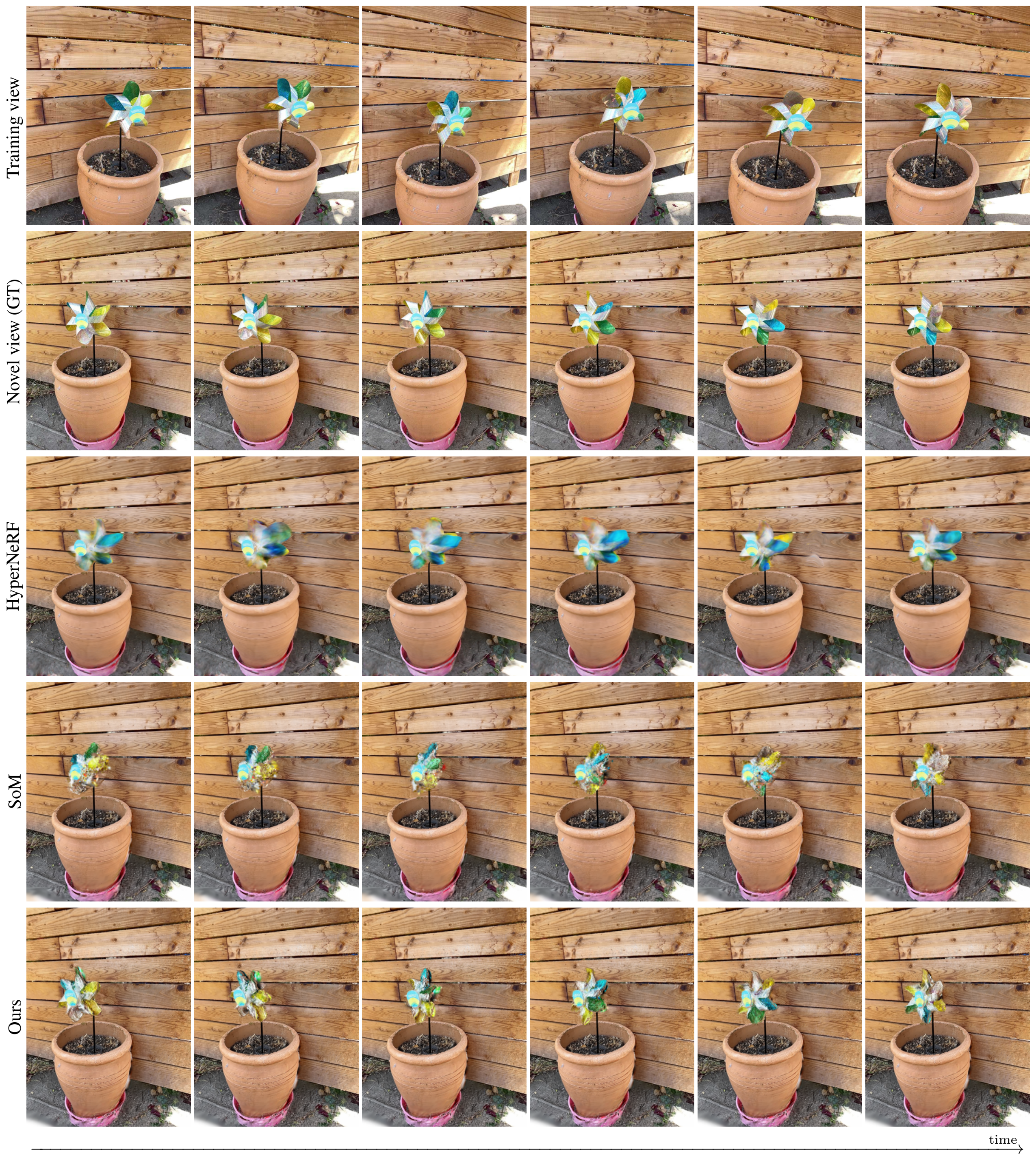

Figure 6. Qualitative comparison of temporal consistency at novel view on the scene “Paper-windmill” of iPhone dataset [2]. The time interval of adjacent images is ten frames.

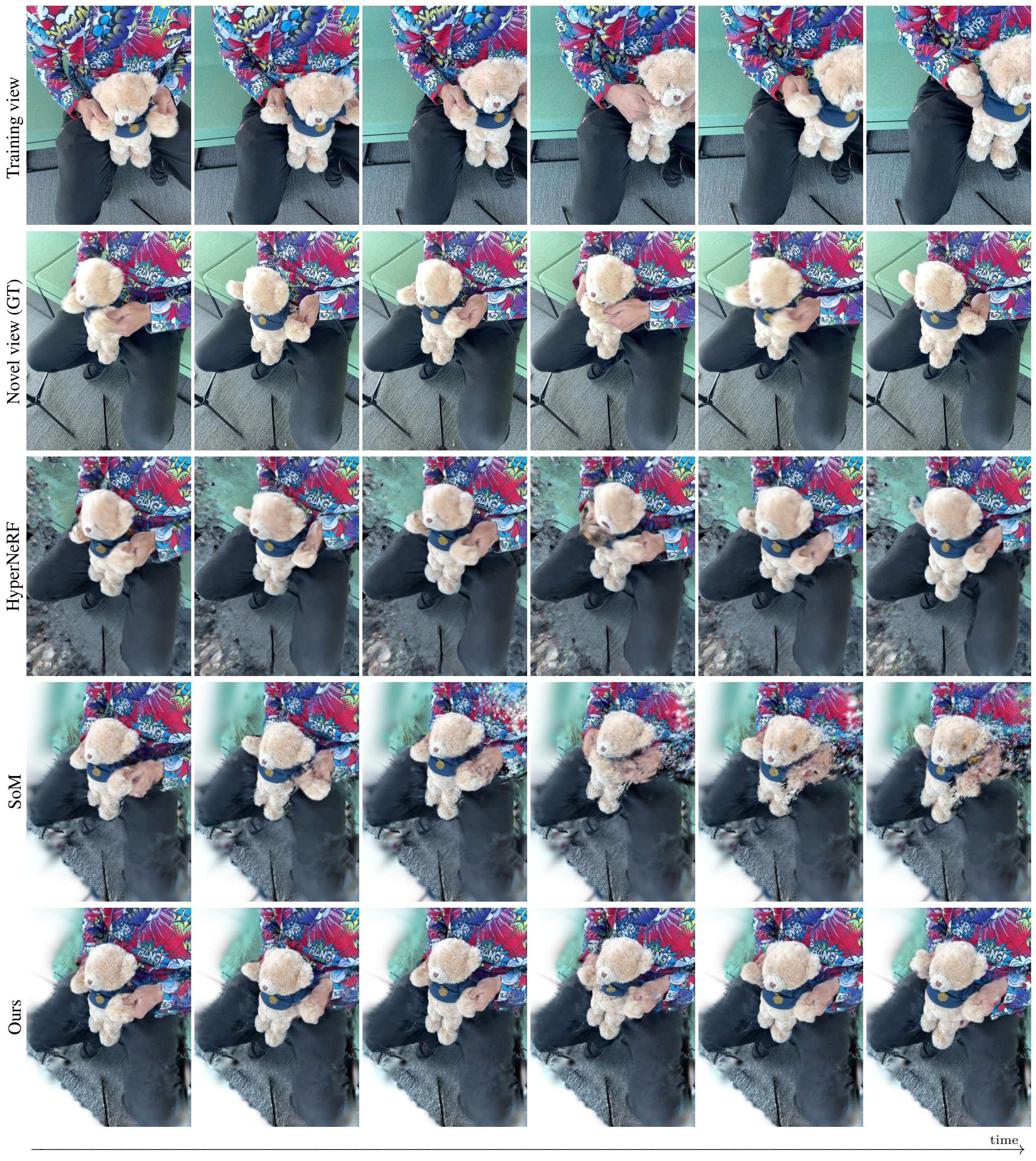

Figure 7. Qualitative comparison of temporal consistency at novel view on the scene “Teddy” of iPhone dataset [2]. The time interval of adjacent images is ten frames.