

TRACE: Learning 3D Gaussian Physical Dynamics from Multi-view Videos

Supplementary Material

The appendix includes:

- Rendering equation and preliminary for vanilla 3DGS.
- Implementation details of Auxiliary Deformation Field.
- Details of Equivalent Velocity and Acceleration.
- Details for the definition of Cross-Product Matrix.
- Implementation details of translation and rotation system.
- Training and evaluation resources for the model.
- Full quantitative results for both interpolation and extrapolation tasks.
- Quantitative and qualitative results for continual learning on Panoptic Sports dataset.
- Implementation details for the dynamic segmentation.
- Segmentation on Real-world Scene.
- Additional results for continual learning and the details for Particle NeRF dataset.
- Additional details of datasets.
- Analysis on performance change along prediction time span.
- Analysis on performance against motion complexity.
- Additional quantitative results for ablation study in the main context.
- Additional per-scene quantitative & qualitative results for both interpolation and extrapolation tasks.
- Additional qualitative results for motion segmentation.
- Additional qualitative results for extrapolation beyond dataset time span.

5.1. Preliminary for vanilla 3DGS

3D Gaussian Splatting [19] represents a 3D scene by a set of colored 3D Gaussian kernels. Specifically, each Gaussian kernel is parameterized by a 3D position $\mathbf{P} \in \mathcal{R}^3$, an orientation represented by a quaternion \mathbf{r} , and a scaling \mathbf{s} . By transforming the orientation \mathbf{r} and scaling \mathbf{s} into the rotation matrix \mathbf{R} and scaling matrix \mathbf{S} , a 3D covariance matrix Σ can be composed as $\Sigma = \mathbf{R}\mathbf{S}\mathbf{S}^T\mathbf{R}^T$. Then the Gaussian kernel can be evaluated at any location $\mathbf{x} \in \mathcal{R}^3$ in the 3D space:

$$G(\mathbf{x}) = e^{-\frac{1}{2}(\mathbf{x}-\mathbf{P})^T\Sigma^{-1}(\mathbf{x}-\mathbf{P})}. \quad (7)$$

Besides, each Gaussian kernel has an opacity σ indicating its influence in rendering, and a color \mathbf{c} computed from spherical harmonics (SH) for view-dependent appearance.

The rendering of Gaussian kernels on the image consists of two steps. Firstly, the Gaussian kernels are projected onto the image plane, following the differentiable rasterization pipeline proposed in [62]. The 3D position \mathbf{P} and covariance matrix Σ of each Gaussian kernel are projected into 2D position $\mathbf{P}' = \mathbf{J}\mathbf{W}\mathbf{P}$ and covariance matrix $\Sigma' = \mathbf{J}\mathbf{W}\Sigma\mathbf{W}^T\mathbf{J}^T$ respectively, where \mathbf{J} denotes the Jaco-

bian of the approximated projective transformation and \mathbf{W} denotes the transformation from the world to camera coordinates. Secondly, the color of a pixel μ on the image can be rendered by α -blending as follows:

$$C(\mu) = \sum_i T_i \alpha_i \mathbf{c}_i, \quad T_i = \prod_{j=1}^{i-1} (1 - \alpha_j), \quad (8)$$

where α_i is obtained by evaluating the projection of the Gaussian kernel G_i on the pixel μ , i.e., $\alpha_i = \sigma_i e^{-\frac{1}{2}(\mu-\mathbf{P}')^T\Sigma'^{-1}(\mu-\mathbf{P})}$. By adjusting the parameters of Gaussian kernels mentioned above and adaptively controlling the Gaussian density, a high-fidelity representation of a 3D scene can be obtained from multi-view images. We refer readers to [19] for more details.

5.2. Implementation Details of Auxiliary Deformation Field

We leverage an existing deformation field introduced in [55] as our auxiliary deformation field. In particular, the 3D position \mathbf{x}_0 of each canonical Gaussian kernel and the current timestamp t are fed into an MLP-based deformation network, denoted as f_{defo} . The implementation of this MLP is directly adapted from [55], i.e., an MLP with 8 layers in total and 256 hidden sizes for each layer, plus a ResNet layer at layer 4. At the input layer, an 8-degree positional embedding is applied to the 3D position \mathbf{x}_0 and a 5-degree positional embedding onto time t .

Mathematically, the deformation field $f_{defo}(\mathbf{x}, t) : \mathcal{R}^4 \rightarrow \mathcal{R}^{10}$ is defined as

$$(\delta\mathbf{x}, \delta\mathbf{r}, \delta\mathbf{s}) = f_{defo}(\mathbf{x}_0, t), \quad (9)$$

where $\delta\mathbf{x}$ represents the translation of the center of Gaussian kernel, $\delta\mathbf{r}$ represents the rotation for the pose of Gaussian kernel in quaternion representation, and $\delta\mathbf{s}$ is the difference of Gaussian sizes. Note a Gaussian size vector is parametrized by $\mathbf{z} = \log(\mathbf{s})$, so the difference can be defined as $\delta\mathbf{s} = \exp(\delta\mathbf{z})$.

After applying the deformation field onto Gaussian kernels, we can deform the Gaussian kernels from canonical time to time t as:

$$\mathbf{x}_t = \mathbf{x}_0 + \delta\mathbf{x} \quad (10)$$

$$\mathbf{r}_t = \delta\mathbf{r} \circ \mathbf{r}_0 \quad (11)$$

$$\mathbf{s}_t = \exp(\mathbf{z}_0 + \delta\mathbf{z}) = \mathbf{s}_0 \odot \delta\mathbf{s}, \quad (12)$$

where \circ is quaternion multiplication and \odot is element-wise multiplication.

5.3. Details of Equivalent Velocity and Acceleration

Given a rigid particle P , we define the following physical parameters:

- Group #1 - Rotation Center Parameters including: 1) the center's position $P_c \in \mathcal{R}^3$, 2) the center's velocity $v_c \in \mathcal{R}^3$, and 3) acceleration $a_c \in \mathcal{R}^3$ in world coordinate.
- Group #2 - Rigid Particle Rotational Parameters including: 1) the rigid particle's rotation vector $w_p \in \mathcal{R}^3$ with regard to its center P_c , and 2) the rigid particle's angular acceleration $\epsilon_p \in \mathcal{R}^3$.

Then we can derive the composite velocity for this particle as:

$$v_p^t = w_p^t \times (P - P_c^t) + v_c^t = w_p^t \times P + (v_c^t - w_p^t \times P_c^t). \quad (13)$$

Now we define the equivalent velocity and acceleration as:

$$\bar{v}_c^t = v_c^t - w_p^t \times P_c^t, \quad \bar{a}_c^t = a_c^t - \epsilon_p^t \times P_c^t. \quad (14)$$

By definition, the 1st-order equivalence is naturally obeyed by the definition. To be specific, it can be shown as:

$$\bar{v}_p^t = w_p^t \times P + \bar{v}_c^t = v_p^t \quad (15)$$

Now let's prove the 2nd-order equivalence:

$$\bar{v}_p^{t+dt} = (w_p^t + dt \epsilon_p^t) \times P + \bar{v}_c^t + dt \bar{a}_c^t \quad (16)$$

$$= (w_p^t + dt \epsilon_p^t) \times P + v_c^t - w_p^t \times P_c^t \quad (17)$$

$$+ dt (a_c^t - \epsilon_p^t \times P_c^t) \quad (18)$$

$$= (w_p^t + dt \epsilon_p^t) \times P - (w_p^t + dt \epsilon_p^t) \times P_c^t \quad (19)$$

$$+ v_c^t + dt a_c^t \quad (20)$$

$$= (w_p^t + dt \epsilon_p^t) \times (P - P_c^t) + (v_c^t + dt a_c^t) \quad (21)$$

$$= v_p^{t+dt} \quad (22)$$

5.4. Details of Cross-Product Matrix

Here we first introduce the definition of Cross-Product Matrix. Given two vectors $k = [k_1, k_2, k_3]^T \in \mathcal{R}^3$ and $v = [v_1, v_2, v_3] \in \mathcal{R}^3$, one can calculate the cross product of these two vectors $k \times v$. Now we define the Cross-Product Matrix of vector k as $K \in \mathcal{R}^{3 \times 3}$, such that

$$Kv = k \times v \quad (23)$$

$$= \begin{bmatrix} -k_3 v_2 + k_2 v_3 \\ k_3 v_1 - k_1 v_3 \\ -k_2 v_1 + k_1 v_2 \end{bmatrix} = \begin{bmatrix} 0 & -k_3 & k_2 \\ k_3 & 0 & k_1 \\ -k_2 & k_1 & 0 \end{bmatrix} v. \quad (24)$$

Thereby we can get the closed-form for the Cross-Product Matrix of vector $k = [k_1, k_2, k_3]^T$ as

$$K = \begin{bmatrix} 0 & -k_3 & k_2 \\ k_3 & 0 & k_1 \\ -k_2 & k_1 & 0 \end{bmatrix} \quad (25)$$

Now we want to derive the cross-product matrix for vector $w_p^{mid} / \|w_p^{mid}\|$. We just need to plug the vector components into the above Equation 25.

5.5. Implementation details of translation and rotation system

As discussed in the main context, we learn the following two groups of physical parameters for each 3D particle P as function $f_{trd}(x) : \mathcal{R}^3 \rightarrow \mathcal{R}^{13}$:

$$\{(\bar{v}_c, \bar{a}_c), (w_p, \epsilon_p)\} = f_{trd}(P), \quad (26)$$

where angular velocity is defined as $\omega = \|w_p\|_2$ around the rotation axes direction $\hat{k} = w_p / \omega$ following the right-hand rule.

This module is implemented by a simple 8×256 MLPs with 8 layers in total and 256 hidden sizes for each layer. In addition, an 8-degree positional embedding is applied onto the input 3D position x , and ReLU is chosen as the activation function.

5.6. Training and evaluation resources for the model

As the complexity of different scenes varies, the total number of Gaussians learned for each scene varies from 40k to 1.6M. In general, our training time is 1.05 times longer than DefGS (or 4DGS if built on it). For example, on the *bat* of Dynamic Object Dataset, DefGS/4DGS need 25 minutes, while we need 27 minutes, with a slight training cost addition. Since our additional module is a tiny MLPs, we only need 367.4kB larger storage. Our rendering speed is 0.85 times slower than DefGS (or 0.8 times slower than 4DGS if built on it). For example, on the *bat* of Dynamic Object Dataset, they achieve 40 fps and ours 32 fps. We train all our models on a single NVIDIA 3090 24G GPU.

5.7. Full Quantitative Results for both interpolation and extrapolation tasks

We show the complete quantitative results for both novel-view *interpolation* and future *extrapolation* tasks in Table 5 and Table 6.

5.8. Quantitative and qualitative results for continual learning on Panoptic Sports dataset

We also conduct experiments on the Panoptic Sports Dataset. It has 6 realistic non-rigid scenes like human interactions, where each scene has 31 viewpoints and each viewpoint has 150 timestamps/frames in total. This dataset is naturally suitable for our continual learning setting and we evaluate our method as follows: 1) the first 15 timestamps are selected for initial training; 2) for every later round, we load one more timestamp into training set and evaluate future prediction on the future 10 timestamps; 3) we keep training till 135th timestamp. Quantitative results are presented in Figure 6 and qualitative results are shown in Figure 5. We can see that our method succeeds in extremely challenging non-rigid scenes with sudden motions and non-constant forces. By design, our method is also flexible to advanced hierarchical deformation fields, and Gaus-

Table 5. Quantitative results of all methods for both future frame extrapolation and novel view interpolation on Dynamic Object Dataset and Dynamic Indoor Scene Dataset.

	Dynamic Object Dataset						Dynamic Indoor Scene Dataset					
	Interpolation			Extrapolation			Interpolation			Extrapolation		
	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow
T-NeRF[40]	13.163	0.709	0.353	13.818	0.739	0.324	24.944	0.742	0.336	22.242	0.700	0.363
D-NeRF[40]	14.158	0.697	0.352	14.660	0.737	0.312	25.380	0.766	0.300	20.791	0.692	0.349
TiNeuVox[13]	27.988	0.960	0.063	19.612	0.940	0.073	29.982	0.864	0.213	21.029	0.770	0.281
T-NeRF _{PINN}	15.286	0.794	0.293	16.189	0.835	0.230	16.250	0.441	0.638	17.290	0.477	0.618
HexPlane _{PINN}	27.042	0.958	0.057	21.419	0.946	0.067	25.215	0.763	0.389	23.091	0.742	0.401
NSFF[25]	-	-	-	-	-	-	29.365	0.829	0.278	24.163	0.795	0.289
NVFi[21]	29.027	0.970	0.039	27.594	0.972	0.036	<u>30.675</u>	0.877	0.211	29.745	0.876	0.204
DefGS[55]	<u>37.865</u>	<u>0.994</u>	<u>0.007</u>	19.849	0.949	0.045	29.926	<u>0.916</u>	<u>0.130</u>	21.380	0.819	0.188
DefGS _{nvfi}	37.316	<u>0.994</u>	0.008	28.749	<u>0.984</u>	<u>0.013</u>	30.170	<u>0.915</u>	0.133	31.096	<u>0.945</u>	<u>0.077</u>
4DGS[52]	37.285	0.986	0.020	20.354	0.950	0.052	29.381	0.889	0.212	21.107	0.793	0.274
TRACE_{Adgs} (Ours)	35.676	0.985	0.021	<u>30.327</u>	0.983	0.019	28.804	0.862	0.242	30.607	0.896	0.209
TRACE (Ours)	39.305	0.995	0.006	31.597	0.987	0.009	32.088	0.929	0.093	34.824	0.965	0.054

Table 6. Quantitative results of all methods for future frame extrapolation optionally with novel view interpolation on NVIDIA Dynamic Scene Dataset, Dynamic Multipart Dataset

	NVIDIA Dynamic Scene Dataset						Dynamic Multipart Dataset					
	Interpolation			Extrapolation			Interpolation			Extrapolation		
	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow
T-NeRF[40]	23.078	0.684	0.355	21.120	0.707	0.358	9.833	0.567	0.550	10.064	0.576	0.537
D-NeRF[40]	22.827	0.711	0.309	20.633	0.709	0.327	13.279	0.747	0.378	13.344	0.767	0.340
TiNeuVox[13]	28.304	0.868	0.216	24.556	0.863	0.215	29.957	0.966	0.067	20.804	0.923	0.090
T-NeRF _{PINN}	18.443	0.597	0.439	17.975	0.605	0.428	-	-	-	-	-	-
HexPlane _{PINN}	24.971	0.818	0.281	24.473	0.818	0.279	-	-	-	-	-	-
NVFi[21]	<u>27.138</u>	0.844	0.231	<u>28.462</u>	0.876	0.214	27.516	0.960	0.052	25.235	0.955	0.046
DefGS[55]	26.662	<u>0.893</u>	<u>0.127</u>	24.240	0.895	0.140	34.635	0.990	0.019	20.664	0.930	0.067
DefGS _{nvfi}	26.972	0.890	0.128	27.529	<u>0.927</u>	<u>0.102</u>	34.637	0.990	0.018	28.455	0.979	0.017
4DGS[52]	19.411	0.462	0.532	22.510	0.703	0.408	37.021	0.992	<u>0.014</u>	20.564	0.935	0.067
TRACE_{Adgs} (Ours)	18.618	0.432	0.553	22.554	0.721	0.390	<u>35.993</u>	<u>0.991</u>	0.016	<u>32.317</u>	<u>0.985</u>	0.016
TRACE (Ours)	26.861	0.912	0.089	29.341	0.933	0.074	35.768	<u>0.991</u>	0.011	33.481	0.990	0.007

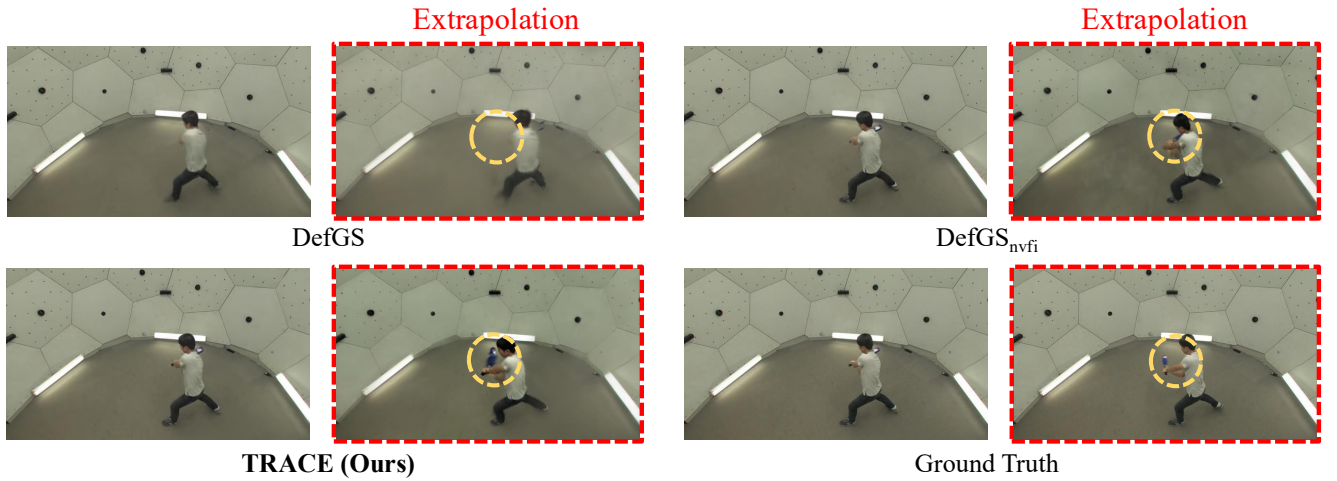


Figure 5. Qualitative Results on Panoptic Sports Dataset.

sians are traced in a Lagrangian representation, thus mass

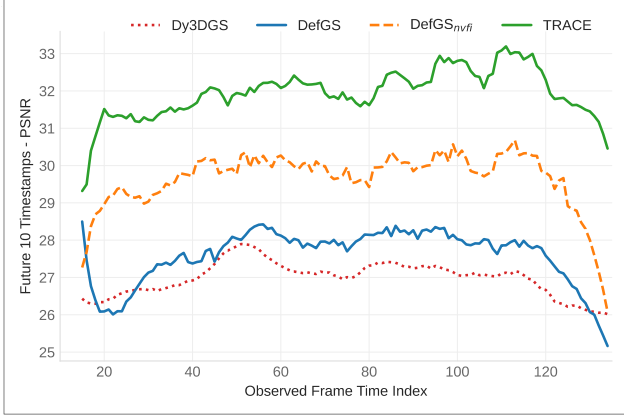


Figure 6. Quantitative Results on Panoptic Sports Dataset.

5.9. Implementation Details for the Segmentation of DefGS / DefGS_{nvfi}

Given a well-trained DefGS or DefGS_{nvfi} model with N canonical Gaussian kernels, we first assign learnable per-Gaussian object codes $\mathbf{O} \in (0, 1)^{N \times K}$ to all Gaussian kernels, where K is the maximum number of objects that is expected to appear in the scene.

After that, we query the position displacements for Gaussians kernels from the well-trained deformation field at time 0 and t respectively, thus obtaining the Gaussians \mathbf{P}_0 at time 0 and \mathbf{P}_t at time t . Then the per-Gaussian scene flows \mathbf{M}_t from time 0 to t is calculated as $\mathbf{M}_t = \mathbf{P}_t - \mathbf{P}_0$.

Lastly, two losses proposed in OGC [47] are computed on the learnable object codes. **1) Dynamic rigid consistency:** For the k^{th} object, we first retrieve its (soft) binary mask \mathbf{O}^k , and feed the tuple $\{\mathbf{P}_0, \mathbf{P}_t, \mathbf{O}^k\}$ into the weighted-Kabsch algorithm to estimate its transformation matrix $\mathbf{T}_k \in \mathbb{R}^{4 \times 4}$ belonging to $SE(3)$ group. Then the dynamic loss is computed as:

$$\ell_{dynamic} = \frac{1}{N} \sum_{\mathbf{p} \in \mathbf{P}_0} \left\| \left(\sum_{k=1}^K o_{\mathbf{p}}^k \cdot (\mathbf{T}_k \circ \mathbf{p}) \right) - (\mathbf{p} + \mathbf{m}_t) \right\|_2$$

where $o_{\mathbf{p}}^k$ represents the probability of being assigned to the k^{th} object for a specific point \mathbf{p} , and $\mathbf{m}_t \in \mathbb{R}^3$ represents the motion vector of \mathbf{p} from time 0 to t . The operation \circ applies the rigid transformation to the point. This loss aims to discriminate objects with different motions. **2) Spatial smoothness:** For each point \mathbf{p} in \mathbf{P}_0 , we first search H nearest neighboring points. Then the smoothness loss is defined as:

$$\ell_{smooth} = \frac{1}{N} \sum_{\mathbf{p} \in \mathbf{P}_0} \left(\frac{1}{H} \sum_{h=1}^H \|o_{\mathbf{p}} - o_{\mathbf{p}_h}\|_1 \right) \quad (27)$$

where $o_{\mathbf{p}} \in (0, 1)^K$ represents the object assignment of the center point \mathbf{p} , and $o_{\mathbf{p}_h} \in (0, 1)^K$ represents the object assignment of its h^{th} neighboring point. This loss aims to

avoid the over-segmentation issues. More details are provided in [47].

In our experiments for DefGS and DefGS_{nvfi}, the maximum number of predicted objects K is set to be 8. A softmax activation is applied on per-Gaussian object codes. During optimization, we adopt the Adam optimizer with a learning rate of 0.01 and optimize object codes for 1000 iterations until convergence.

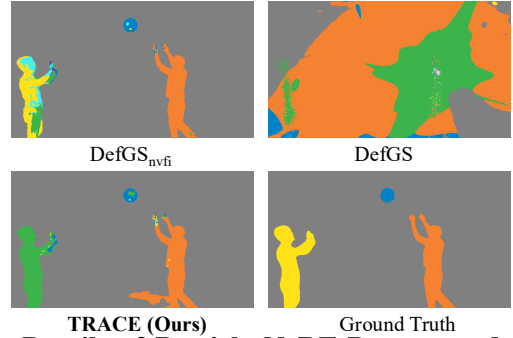
5.10. Segmentation on Real-world Scene

We select the basketball scene from Panoptic Sports Dataset, and use manual prompts on Track-Anything to get pseudo ground truth. We render 2D masks on all $31 \times 150 = 4650$ views for evaluation. Results are shown in Table 7 and Figure 7.

Table 7. Segmentation results on basketball scene of Panoptic Sports Dataset.

	AP \uparrow	PQ \uparrow	F1 \uparrow	Pre \uparrow	Rec \uparrow	mIoU \uparrow
DefGS	5.64	15.30	20.51	15.53	30.23	32.05
DefGS _{nvfi}	43.52	53.60	64.02	52.95	80.93	68.25
TRACE (Ours)	53.40	59.46	72.06	64.64	81.40	66.19

Figure 7. Qualitative results of segmentation.



5.11. Details of Particle NeRF Dataset and Additional Results for Continual Learning

We first introduce details for Particle NeRF datasets. This dataset comprises 6 scenes, and each has 40 views. We reserve 36 views for training and 4 views for novel view evaluation. The details for motions and time splits for each scene are listed following:

- **cloth:** This scene includes a piece of square cloth folded by external forces.
- **wheel:** This scene includes a rolling wheel with a constant angular velocity.
- **spring:** This scene includes a box tied onto a spring, and undergoing a harmonic oscillation motion.
- **robot:** This scene shows a robot arm rotating its poses.
- **robot-task:** This scene shows a robot arm putting a box onto a sliding platform.
- **pendulums:** This scene shows two pendulums and each of them undergoes harmonic oscillation motion.

For the first three scenes, we define the 140-th frame as virtual time 1.0, and for the latter three scenes we define the 70-th frame as virtual time 1.0.

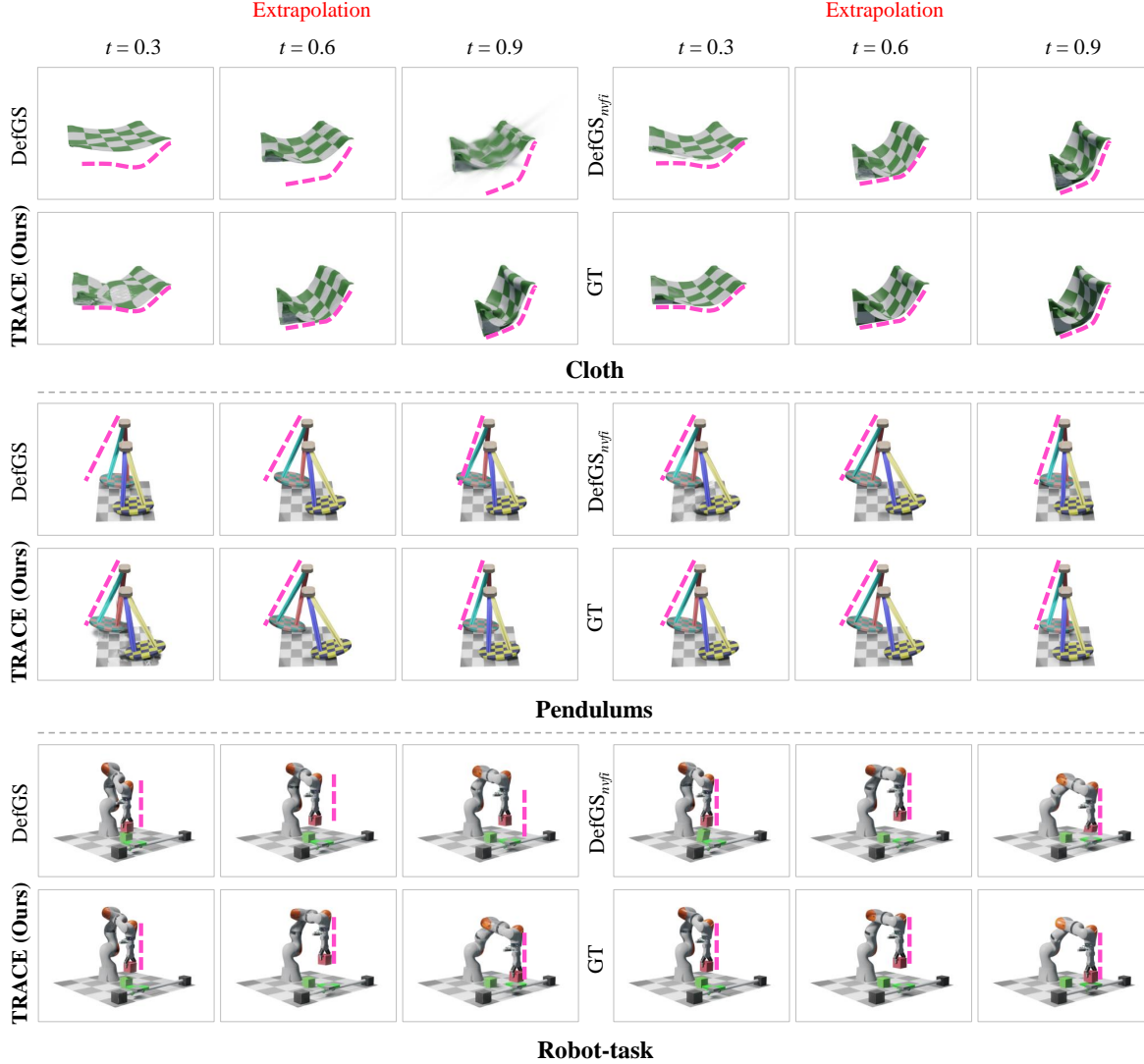


Figure 8. Qualitative **extrapolation** results for continual learning on Particle NeRF dataset.

Figure 8 shows the qualitative future frame extrapolation results. Table 8 shows the quantitative past-time interpolation results.

Table 8. Quantitative **interpolation** results (PSNR) of continual learning.

Observed Time	0.15	0.30	0.45	0.60	0.75	Average
Extrap Till	0.30	0.45	0.60	0.75	0.90	
DefGS[55]	38.665	37.446	36.730	36.229	35.820	36.978
DefGS _{nyfi}	37.984	36.499	35.233	35.168	34.896	35.956
TRACE (Ours)	35.213	34.705	33.164	33.567	33.414	34.013

5.12. Additional details of our new dataset

Dynamic Multipart dataset: This dataset comprises 4 distinct objects *, including a variety of challenging mo-

* All objects are purchased from SketchFab, licensed under the SketchFab Standard License: <https://sketchfab.com/licenses>, and are all allowed for AI generation model usage

tions. Details of the 4 dynamic objects are:

- **Foldingchair:** A folding chair is given. This chair is composed of three parts. The whole motion is unfolding this chair, so all three parts are undergoing different rotating motions.
- **Hypoerbolic Slot:** This is an extremely hard case, where a stick is rotating through a hypoerbolic slot. Note that, only the stick in this hypoerbolic shape is dynamic, this introduces more challenges in motion extrapolation.
- **Satellite:** This object is a satellite with two wing doors opening and one main door opening, all rotating in different directions.
- **Stove:** A home stove is given. The motion is mainly closing its top cover plate.

5.13. Additional Quantitative Results for Ablation Study for the main context

Here we show the total results for the ablation study in Table 9, both for interpolation and extrapolation.

Table 9. Ablation studies on three datasets.

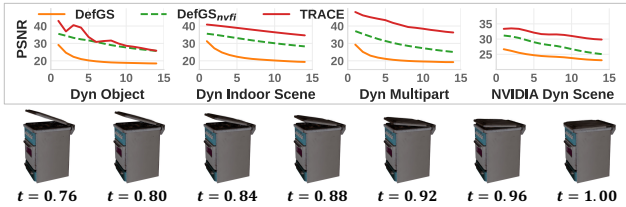
Interpolation													
				Dynamic Multipart			Dynamic Object			Dynamic Indoor Scene			
order f_{defo} physcis equiv				PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	
(1) δt	2	✓	✓	✓	35.370	0.991	0.018	39.305	0.995	0.006	32.184	0.928	0.121
(1) $2\delta t$	2	✓	✓	✓	35.768	0.991	0.011	38.518	0.995	0.005	32.088	0.929	0.093
(1) $3\delta t$	2	✓	✓	✓	35.666	0.991	0.011	37.718	0.994	0.005	32.044	0.929	0.094
(2) $2\delta t$	1	✓	✓	✓	35.369	0.991	0.018	37.859	0.994	0.006	32.250	0.927	0.122
(2) $3\delta t$	3	✓	✓	✓	35.832	0.991	0.011	38.669	0.995	0.004	32.111	0.929	0.093
(3) $2\delta t$	2	✗	✓	✓	18.906	0.895	0.133	-	-	-	-	-	-
(4) $2\delta t$	2	✓	✗	✓	35.526	0.992	0.011	-	-	-	-	-	-
(5) $2\delta t$	2	✓	✓	✗	35.225	0.991	0.011	-	-	-	-	-	-

Extrapolation													
				Dynamic Multipart			Dynamic Object			Dynamic Indoor Scene			
order f_{defo} physcis equiv				PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	
(1) δt	2	✓	✓	✓	32.852	0.989	0.010	31.597	0.987	0.009	33.831	0.958	0.078
(1) $2\delta t$	2	✓	✓	✓	33.481	0.990	0.007	31.511	0.988	0.006	34.824	0.965	0.054
(1) $3\delta t$	2	✓	✓	✓	33.003	0.989	0.008	29.787	0.983	0.011	34.778	0.965	0.054
(2) $2\delta t$	1	✓	✓	✓	33.125	0.989	0.010	28.522	0.984	0.012	34.576	0.963	0.076
(2) $3\delta t$	3	✓	✓	✓	33.312	0.989	0.008	31.044	0.987	0.007	34.086	0.961	0.056
(3) $2\delta t$	2	✗	✓	✓	19.206	0.907	0.120	-	-	-	-	-	-
(4) $2\delta t$	2	✓	✗	✓	29.602	0.983	0.012	-	-	-	-	-	-
(5) $2\delta t$	2	✓	✓	✗	29.986	0.985	0.012	-	-	-	-	-	-

5.14. Analysis on Performance Change along Prediction Time Span

We present the extrapolation results for per future timestamp in Figure 9. We can see that, though our method is clearly better than baselines, the performance of all methods decreases as the extrapolation goes further. To alleviate this issue, we advocate incorporating new observations in a continual learning manner.

Figure 9. The top row shows extrapolation results over future timestamps and the bottom shows qualitative results of our extrapolation.

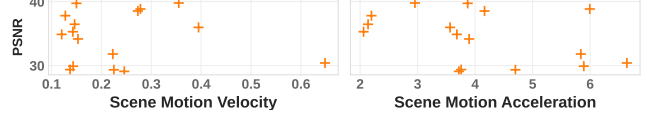


5.15. Analysis on Performance against Motion Complexity

For each scene, we calculate its scene motion complexity as follows: 1) we uniformly sample 40K points on each image of the first timestamp; 2) we use BootsTAPIR[11] to track these points over all timestamps and static points are removed; 3) we calculate optical flows for dynamic points between every two adjacent timestamps; 4) the optical flows are divided by the time interval and then normalized by image size, getting normalized image motion velocity which will be averaged out on the whole scene to obtain *Scene Motion Velocity*; 5) we further calculate the difference between

two adjacent normalized image motion velocities, getting acceleration which will be averaged out on the whole scene to obtain *Scene Motion Acceleration*. Figure 10 shows the relationship where each cross represents one scene out of our 4 datasets. We can see that we achieve similar results over different levels of motion complexity, highlighting the generality of our method.

Figure 10. Extrapolation over different scene motion complexities.



5.16. Additional Quantitative Results on Dynamic Object Dataset

Here we show the total per-scene results on Dynamic Object Dataset in Table 10 and qualitative results in Figures 14, 15&16, both for interpolation and extrapolation.

5.17. Additional Quantitative Results on Dynamic Indoor Scene Dataset

Here we show the total per-scene results in Dynamic Indoor Scene Datasets in Table 11 and qualitative results in Figures 16&17, both for interpolation and extrapolation.

5.18. Additional Quantitative Results on NVIDIA Dynamic Scene Dataset

Here we show the total per-scene results on NVIDIA Dynamic Scene Dataset in Table 12 and qualitative results in Figure 19, both for interpolation and extrapolation.

5.19. Additional Quantitative Results on Dynamic Multipart Dataset

Here we show the total per-scene results on our Dynamic Multipart Dataset in Table 13 and qualitative results in Figure 18, both for interpolation and extrapolation.

5.20. Additional qualitative results for extrapolation beyond dataset time spans

We list some meaningful longer extrapolation results from each dataset here in Figure 24. In our dataset, the training period lasts from $t = 0$ to $t = 0.75$ and the extrapolation period lasts from $t = 0.75$ to $t = 1.0$. Here we show the qualitative results till $t = 1.5$, which is already twice the training period. We can see that our method can still obtain physically meaningful future frame prediction in particularly high quality.

5.21. Additional Qualitative Results for Object/Part Segmentation

Figures 11, 12 and 13 show more qualitative results for the autonomous object or part segmentation based on the learned physical parameters via the simple K-means clustering algorithm.

Table 10. Per-scene quantitative results on Dynamic Object dataset.

Methods	Falling Ball						Bat					
	Interpolation			Extrapolation			Interpolation			Extrapolation		
	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow
T-NeRF[40]	14.921	0.782	0.326	15.418	0.793	0.308	13.070	0.836	0.234	13.897	0.834	0.230
D-NeRF[40]	15.548	0.665	0.435	15.116	0.644	0.427	14.087	0.845	0.212	15.406	0.887	0.175
TiNeuVox[13]	35.458	0.974	0.052	20.242	0.959	0.067	16.080	0.908	0.108	16.952	0.930	0.115
T-NeRF _{PINN}	17.687	0.775	0.368	17.857	0.829	0.265	16.412	0.903	0.197	18.983	0.930	0.132
HexPlane _{PINN}	32.144	0.965	0.065	20.762	0.951	0.081	23.399	0.958	0.057	21.144	0.951	0.064
NVFi[21]	35.826	0.978	0.041	31.369	0.978	0.041	23.325	0.964	0.046	25.015	0.968	0.042
DefGS[55]	37.535	0.995	0.009	20.442	0.976	0.033	38.750	0.997	0.004	17.063	0.936	0.072
DefGS _{nvfi}	38.606	0.996	0.010	24.873	0.985	0.015	38.075	0.997	0.004	28.950	0.980	0.015
TRACE_{Adgs} (Ours)	27.481	0.943	0.083	29.951	0.974	0.050	38.354	0.997	0.004	28.093	0.980	0.015
TRACE (Ours)	41.394	0.998	0.005	42.713	0.998	0.004	39.384	0.997	0.003	26.449	0.979	0.015

Methods	Fan						Telescope					
	Interpolation			Extrapolation			Interpolation			Extrapolation		
	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow
T-NeRF[40]	8.001	0.308	0.646	8.494	0.392	0.593	13.031	0.615	0.472	13.892	0.670	0.417
D-NeRF[40]	7.915	0.262	0.690	8.624	0.370	0.623	13.295	0.609	0.469	14.967	0.700	0.385
TiNeuVox[13]	24.088	0.930	0.104	20.932	0.935	0.078	31.666	0.982	0.041	20.456	0.921	0.067
T-NeRF _{PINN}	9.233	0.541	0.508	9.828	0.606	0.443	14.293	0.739	0.366	15.752	0.804	0.298
HexPlane _{PINN}	22.822	0.921	0.079	19.724	0.919	0.080	25.381	0.948	0.066	23.165	0.932	0.074
NVFi[21]	25.213	0.948	0.049	27.172	0.963	0.037	26.487	0.959	0.048	27.101	0.963	0.046
DefGS[55]	35.858	0.985	0.017	20.932	0.948	0.038	37.502	0.996	0.003	20.684	0.927	0.048
DefGS _{nvfi}	35.217	0.984	0.019	26.648	0.972	0.023	37.568	0.996	0.003	34.096	0.994	0.005
TRACE_{Adgs} (Ours)	36.052	0.985	0.019	34.622	0.988	0.012	37.668	0.994	0.006	33.338	0.989	0.010
TRACE (Ours)	35.969	0.985	0.015	34.964	0.990	0.008	40.441	0.997	0.003	31.145	0.986	0.006

Methods	Shark						Whale					
	Interpolation			Extrapolation			Interpolation			Extrapolation		
	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow
T-NeRF[40]	13.813	0.853	0.223	15.325	0.882	0.193	16.141	0.860	0.212	15.880	0.860	0.203
D-NeRF[40]	17.727	0.903	0.150	19.078	0.936	0.092	16.373	0.898	0.154	14.771	0.883	0.171
TiNeuVox[13]	23.178	0.971	0.059	19.463	0.950	0.050	37.455	0.994	0.016	19.624	0.943	0.063
T-NeRF _{PINN}	17.315	0.878	0.177	18.739	0.921	0.115	16.778	0.927	0.141	15.974	0.919	0.127
HexPlane _{PINN}	28.874	0.976	0.040	22.330	0.961	0.047	29.634	0.981	0.035	21.391	0.961	0.053
NVFi[21]	32.072	0.984	0.024	28.874	0.982	0.021	31.240	0.986	0.025	26.032	0.978	0.029
DefGS[55]	37.802	0.994	0.006	19.924	0.957	0.034	39.740	0.997	0.004	20.048	0.951	0.046
DefGS _{nvfi}	37.327	0.994	0.006	29.240	0.987	0.007	37.101	0.996	0.005	28.686	0.986	0.012
TRACE_{Adgs} (Ours)	39.485	0.996	0.006	26.997	0.980	0.010	35.017	0.995	0.008	28.961	0.987	0.017
TRACE (Ours)	40.537	0.997	0.005	27.280	0.982	0.008	38.104	0.997	0.004	27.030	0.985	0.012

5.22. Additional Qualitative Results for Segmentation on Dynamic Indoor Scene Dataset

Figures 20, 21, 22, & 23 shows qualitative results for the rendered mask on Dynamic Indoor Scene dataset.

Table 11. Per-scene quantitative results on Dynamic Indoor Scene dataset.

Methods	Gnome House						Chessboard					
	Interpolation			Extrapolation			Interpolation			Extrapolation		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
T-NeRF[40]	26.094	0.716	0.383	23.485	0.643	0.419	25.517	0.796	0.294	20.228	0.708	0.365
D-NeRF[40]	27.000	0.745	0.319	21.714	0.641	0.367	24.852	0.774	0.308	19.455	0.675	0.384
TiNeuVox[13]	30.646	0.831	0.253	21.418	0.699	0.326	33.001	0.917	0.177	19.718	0.765	0.310
T-NeRF _{PINN}	15.008	0.375	0.668	16.200	0.409	0.651	16.549	0.457	0.621	17.197	0.472	0.618
HexPlane _{PINN}	23.764	0.658	0.510	22.867	0.658	0.510	24.605	0.778	0.412	21.518	0.748	0.428
NSFF[25]	31.418	0.821	0.294	25.892	0.750	0.327	32.514	0.810	0.201	21.501	0.805	0.282
NVFi[21]	30.667	0.824	0.277	30.408	0.826	0.273	30.394	0.888	0.215	27.840	0.872	0.219
DefGS[55]	32.041	0.918	0.132	21.703	0.775	0.207	27.355	0.912	0.147	20.032	0.808	0.218
DefGS _{nvfi}	32.881	0.919	0.132	33.630	0.953	0.077	26.200	0.907	0.156	26.730	0.917	0.110
TRACE_{Adgs} (Ours)	32.055	0.878	0.243	34.033	0.901	0.218	31.101	0.931	0.155	28.337	0.898	0.201
TRACE (Ours)	32.971	0.922	0.106	36.479	0.962	0.065	35.203	0.961	0.064	35.271	0.972	0.050

Methods	Factory						Dining Table					
	Interpolation			Extrapolation			Interpolation			Extrapolation		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
T-NeRF[40]	26.467	0.741	0.328	24.276	0.722	0.344	21.699	0.716	0.338	20.977	0.725	0.324
D-NeRF[40]	28.818	0.818	0.252	22.959	0.746	0.303	20.851	0.725	0.319	19.035	0.705	0.341
TiNeuVox[13]	32.684	0.909	0.148	22.622	0.810	0.229	23.596	0.798	0.274	20.357	0.804	0.258
T-NeRF _{PINN}	16.634	0.446	0.624	17.546	0.480	0.609	16.807	0.486	0.640	18.215	0.548	0.595
HexPlane _{PINN}	27.200	0.826	0.283	24.998	0.792	0.312	25.291	0.788	0.350	22.979	0.771	0.355
NSFF[25]	33.975	0.919	0.152	26.647	0.855	0.196	19.552	0.665	0.464	22.612	0.770	0.351
NVFi[21]	32.460	0.912	0.151	31.719	0.908	0.154	29.179	0.885	0.199	29.011	0.898	0.171
DefGS[55]	33.629	0.943	0.096	22.820	0.839	0.169	27.680	0.890	0.145	20.965	0.855	0.157
DefGS _{nvfi}	33.643	0.943	0.097	33.049	0.954	0.062	27.957	0.891	0.145	30.975	0.955	0.060
TRACE_{Adgs} (Ours)	23.668	0.747	0.385	27.584	0.846	0.275	28.391	0.892	0.184	32.472	0.940	0.141
TRACE (Ours)	33.019	0.943	0.079	35.293	0.965	0.050	27.159	0.891	0.124	32.253	0.961	0.052

Table 12. Quantitative results of our method and baselines on the NVIDIA Dynamic Scene dataset.

	Truck						Skating					
	Interpolation			Extrapolation			Interpolation			Extrapolation		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
T-NeRF[40]	18.673	0.548	0.447	18.176	0.567	0.447	27.483	0.820	0.263	24.063	0.846	0.269
D-NeRF[40]	17.660	0.554	0.431	16.905	0.544	0.445	27.994	0.869	0.187	24.361	0.873	0.208
TiNeuVox[13]	27.230	0.846	0.229	24.887	0.848	0.209	29.377	0.889	0.202	24.224	0.878	0.220
T-NeRF _{PINN}	15.241	0.413	0.540	14.959	0.395	0.552	21.644	0.780	0.338	20.990	0.814	0.303
HexPlane _{PINN}	25.494	0.768	0.337	24.991	0.768	0.325	24.447	0.867	0.225	23.955	0.868	0.232
NVFi[21]	27.276	0.840	0.235	28.269	0.855	0.220	26.999	0.848	0.227	28.654	0.896	0.208
DefGS[55]	28.327	0.885	0.115	24.947	0.875	0.131	24.997	0.900	0.138	23.532	0.914	0.148
DefGS _{nvfi}	28.169	0.884	0.114	28.481	0.922	0.088	25.774	0.896	0.141	26.577	0.931	0.115
TRACE_{Adgs} (Ours)	19.330	0.435	0.551	21.073	0.610	0.481	17.905	0.428	0.554	24.034	0.831	0.298
TRACE (Ours)	28.070	0.878	0.107	28.855	0.919	0.077	25.651	0.946	0.070	29.827	0.946	0.070

Table 13. Per-scene quantitative results on Dynamic Multipart dataset.

Methods	Folding Chair						Hyperbolic Slot					
	Interpolation			Extrapolation			Interpolation			Extrapolation		
	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow
T-NeRF[40]	10.146	0.598	0.537	10.260	0.586	0.548	7.437	0.424	0.749	7.098	0.404	0.739
D-NeRF [40]	11.681	0.717	0.437	13.177	0.765	0.357	7.279	0.485	0.714	7.547	0.468	0.695
TiNeuVox[13]	34.160	0.984	0.039	13.391	0.808	0.199	28.637	0.955	0.083	25.436	0.973	0.040
NVFi[21]	27.748	0.962	0.049	23.433	0.940	0.063	25.487	0.944	0.057	25.757	0.956	0.039
DefGS[55]	37.319	0.995	0.009	13.682	0.820	0.169	31.780	0.983	0.030	25.631	0.981	0.020
DefGS _{nvi}	37.269	0.994	0.009	25.404	0.962	0.022	32.506	0.985	0.025	29.351	0.988	0.012
TRACE_{4dgs} (Ours)	36.707	0.993	0.012	26.993	0.967	0.033	33.507	0.987	0.020	37.502	0.994	0.008
TRACE (Ours)	40.031	0.996	0.005	29.588	0.982	0.010	33.041	0.986	0.017	36.437	0.995	0.005

Methods	Satellite						Stove					
	Interpolation			Extrapolation			Interpolation			Extrapolation		
	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow
T-NeRF[40]	14.614	0.754	0.307	14.468	0.751	0.328	7.134	0.490	0.605	8.429	0.562	0.531
D-NeRF[40]	17.991	0.930	0.100	17.252	0.926	0.102	16.165	0.856	0.262	15.400	0.908	0.205
TiNeuVox[13]	33.061	0.983	0.035	28.627	0.978	0.032	23.969	0.943	0.109	15.760	0.934	0.087
NVFi[21]	29.644	0.973	0.029	30.075	0.975	0.027	27.186	0.959	0.072	21.675	0.950	0.054
DefGS[55]	36.832	0.993	0.007	27.622	0.979	0.016	32.607	0.989	0.029	15.721	0.941	0.063
DefGS _{nvi}	36.640	0.993	0.007	34.282	0.990	0.007	21.134	0.988	0.029	24.781	0.977	0.027
TRACE_{4dgs} (Ours)	35.722	0.992	0.010	30.804	0.986	0.012	38.037	0.992	0.023	33.969	0.991	0.012
TRACE (Ours)	36.938	0.993	0.007	32.093	0.988	0.007	33.062	0.990	0.015	35.805	0.994	0.007



Figure 11. Qualitative results for Object/Part Segmentation on Dynamic Object dataset.

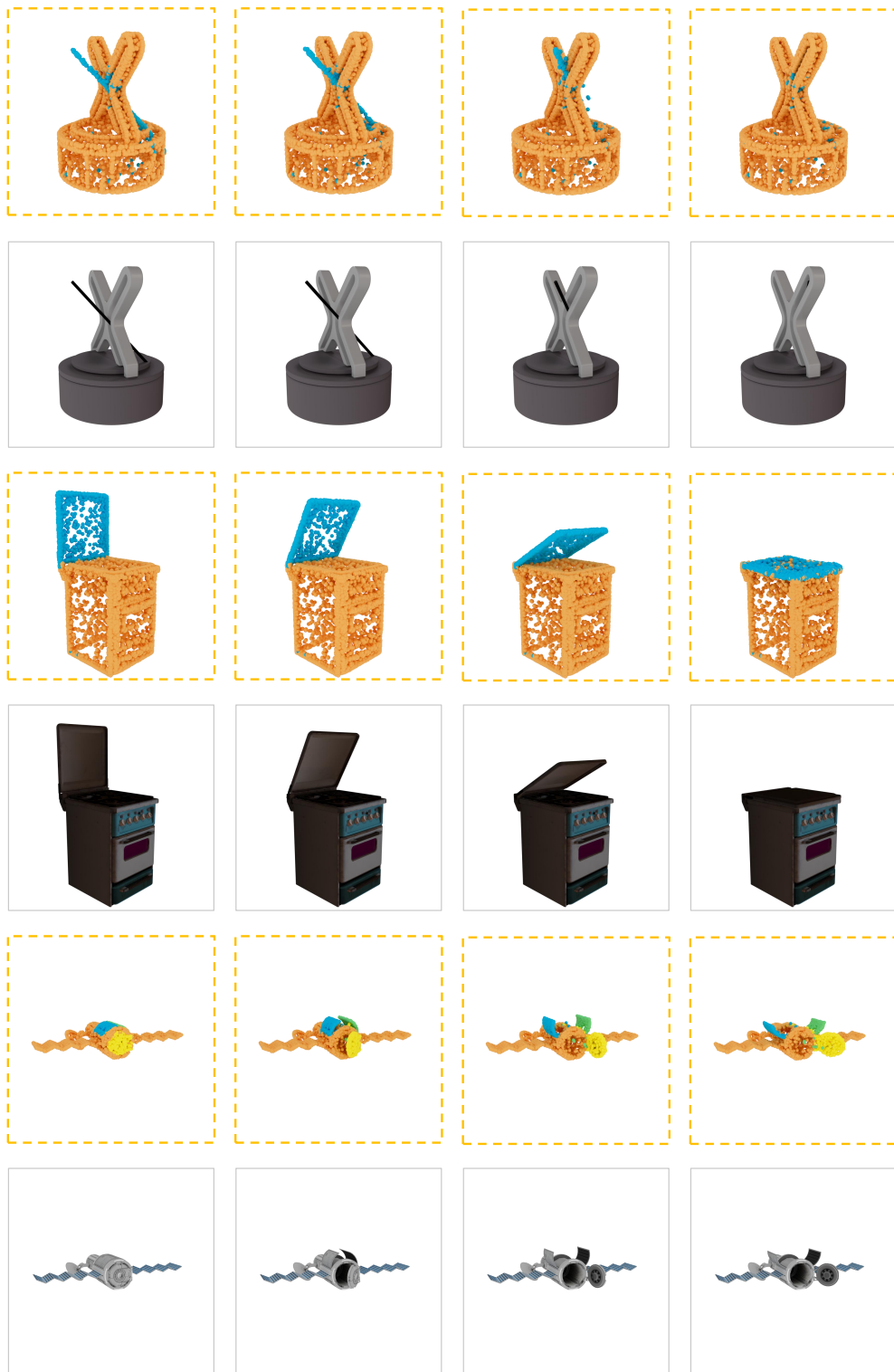


Figure 12. Qualitative results for Object/Part Segmentation on Dynamic Multipart dataset.

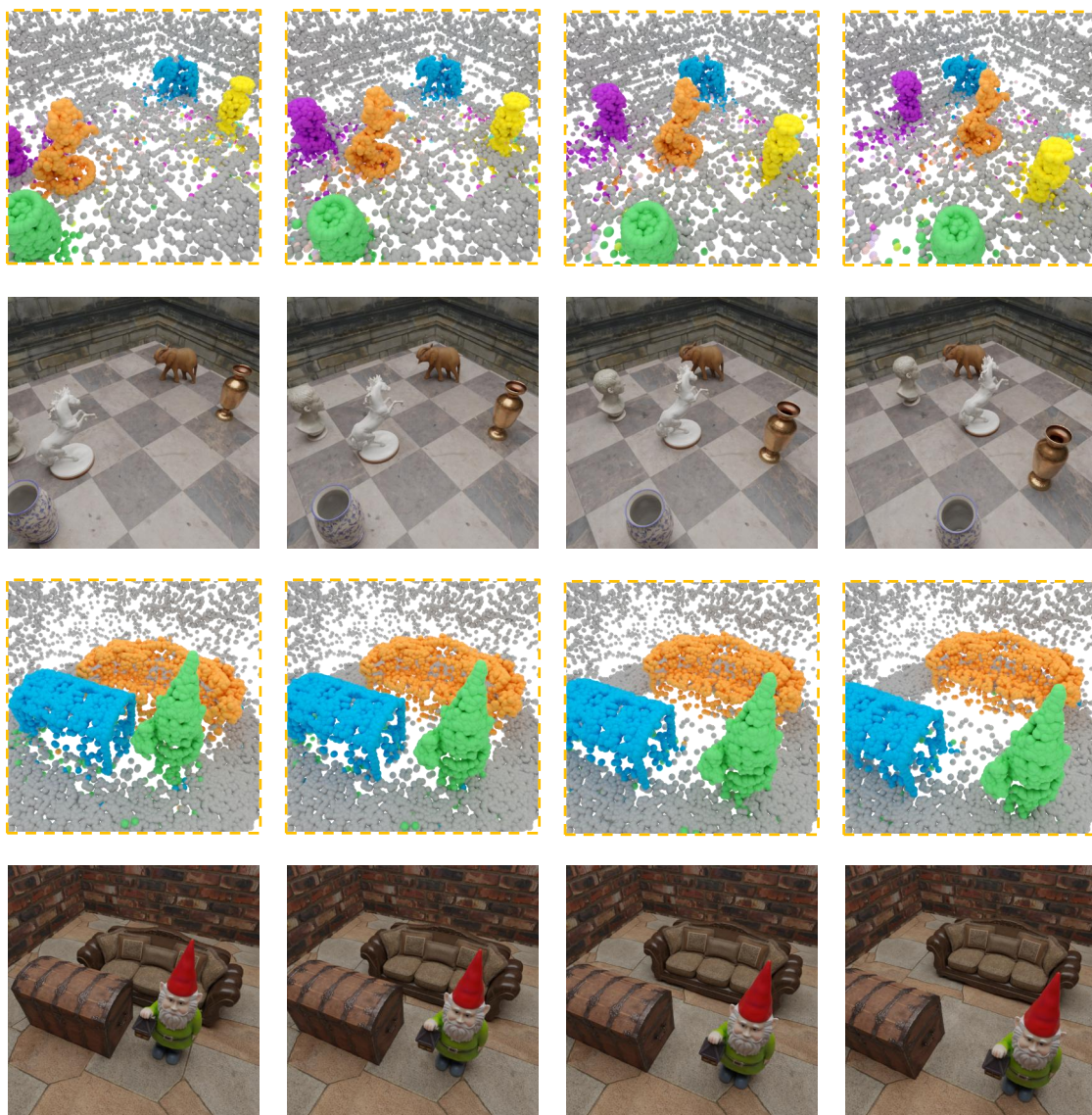


Figure 13. Qualitative results for Object/Part Segmentation on Dynamic Indoor Scene dataset.

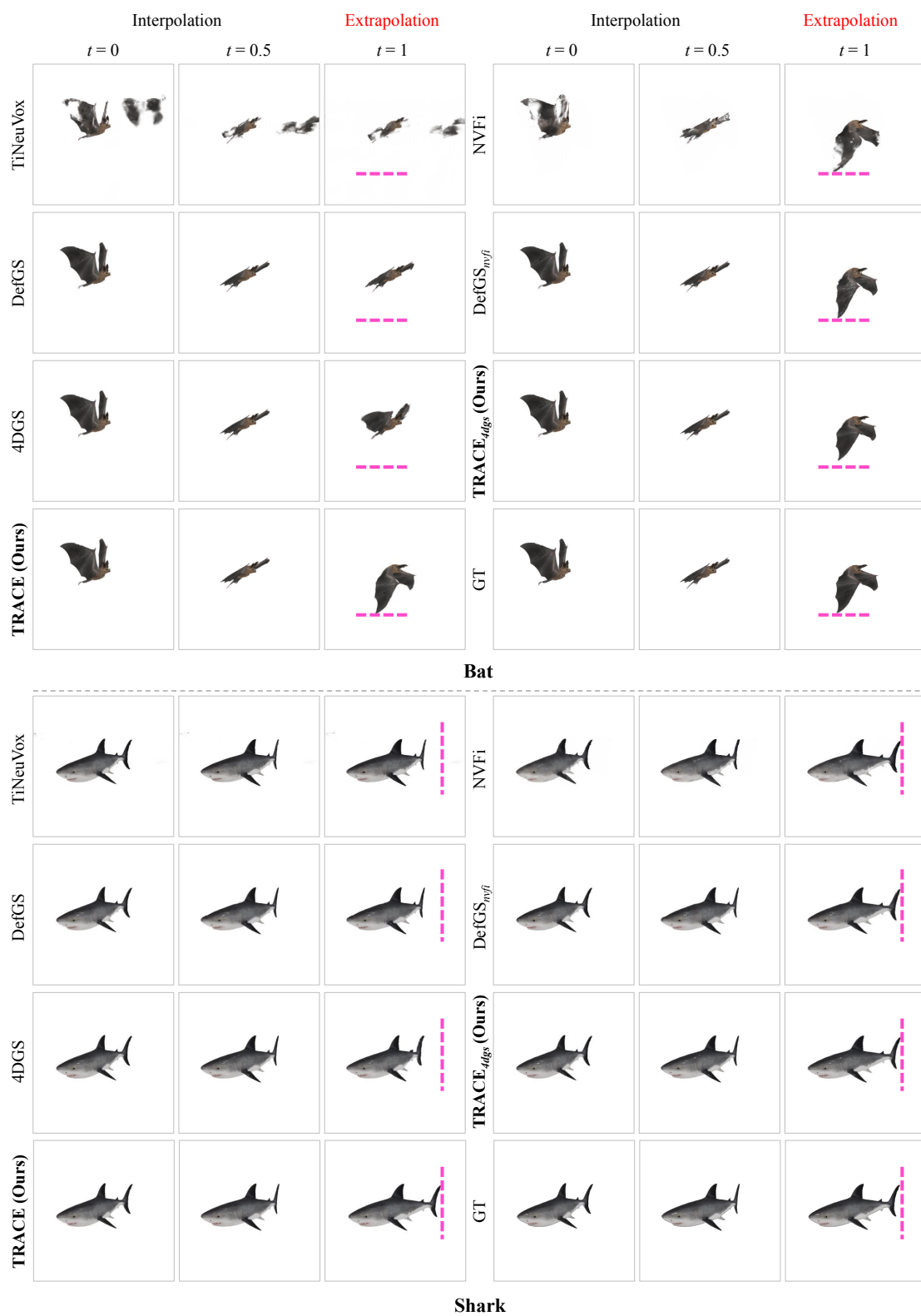
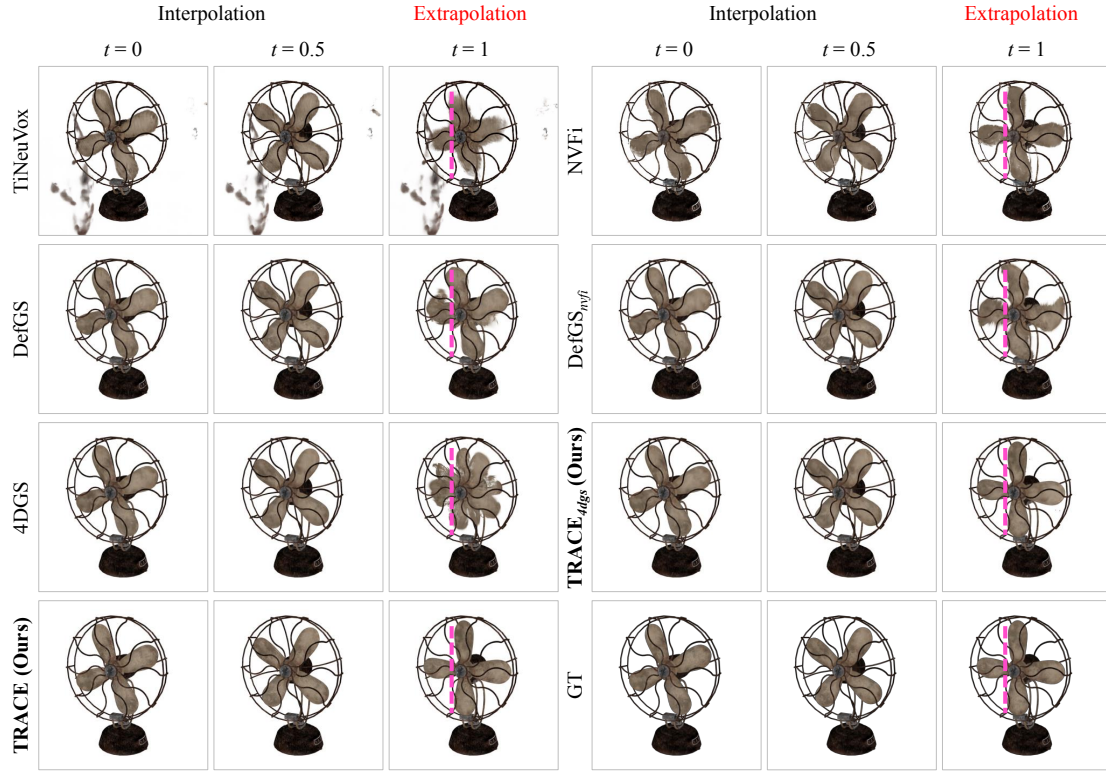


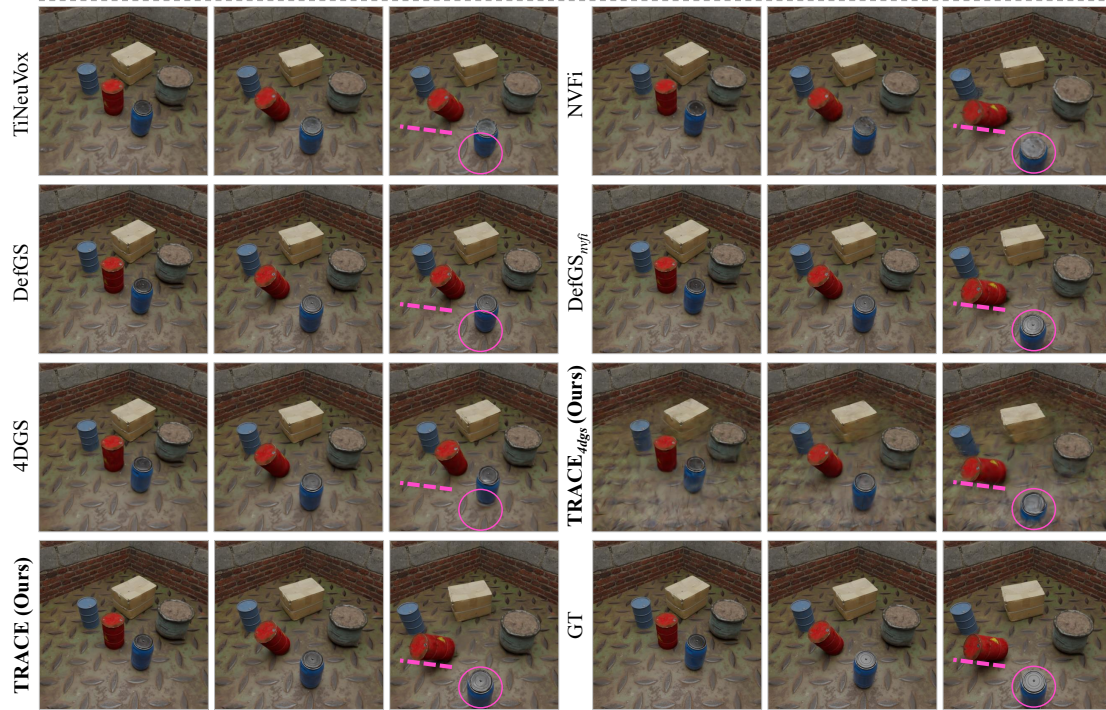
Figure 14. Qualitative results of RGB view synthesis for interpolation and **extrapolation** tasks on Dynamic Object dataset.



Figure 15. Qualitative results of RGB view synthesis for interpolation and **extrapolation** tasks on Dynamic Object dataset.

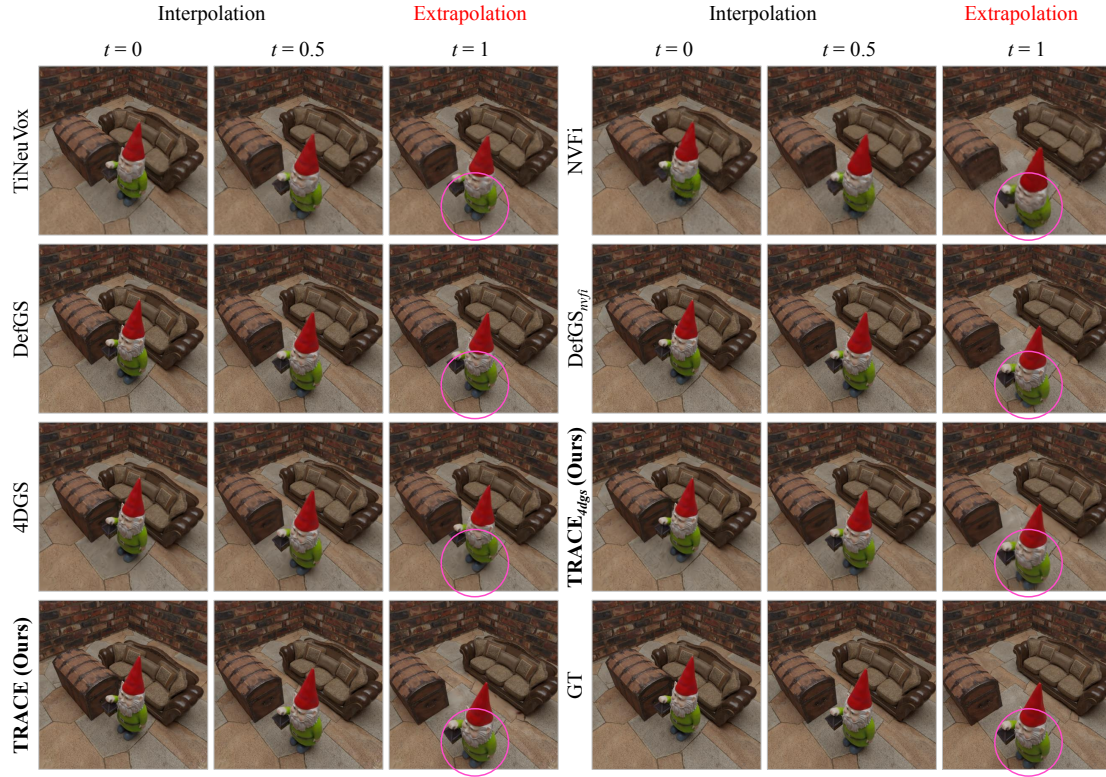


Fan

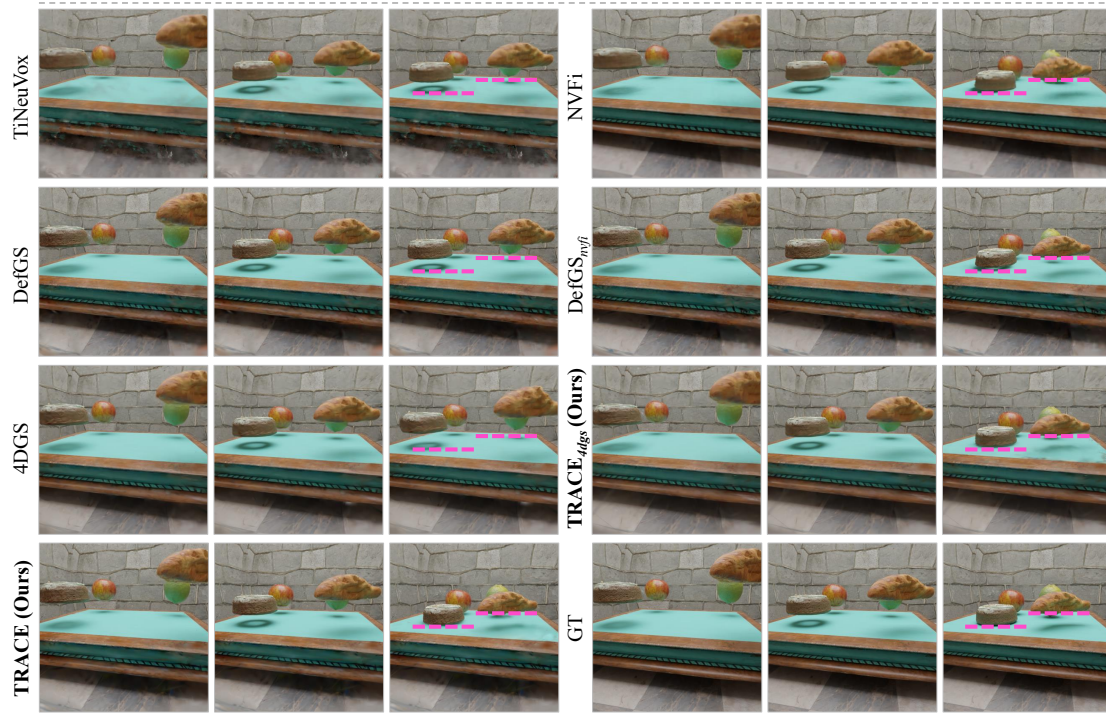


Factory

Figure 16. Qualitative results of RGB view synthesis for interpolation and **extrapolation** tasks on Dynamic Object and Dynamic Indoor Scene datasets.



Gnome House



Dining Table

Figure 17. Qualitative results of RGB view synthesis for interpolation and **extrapolation** tasks on Dynamic Indoor Scene dataset.

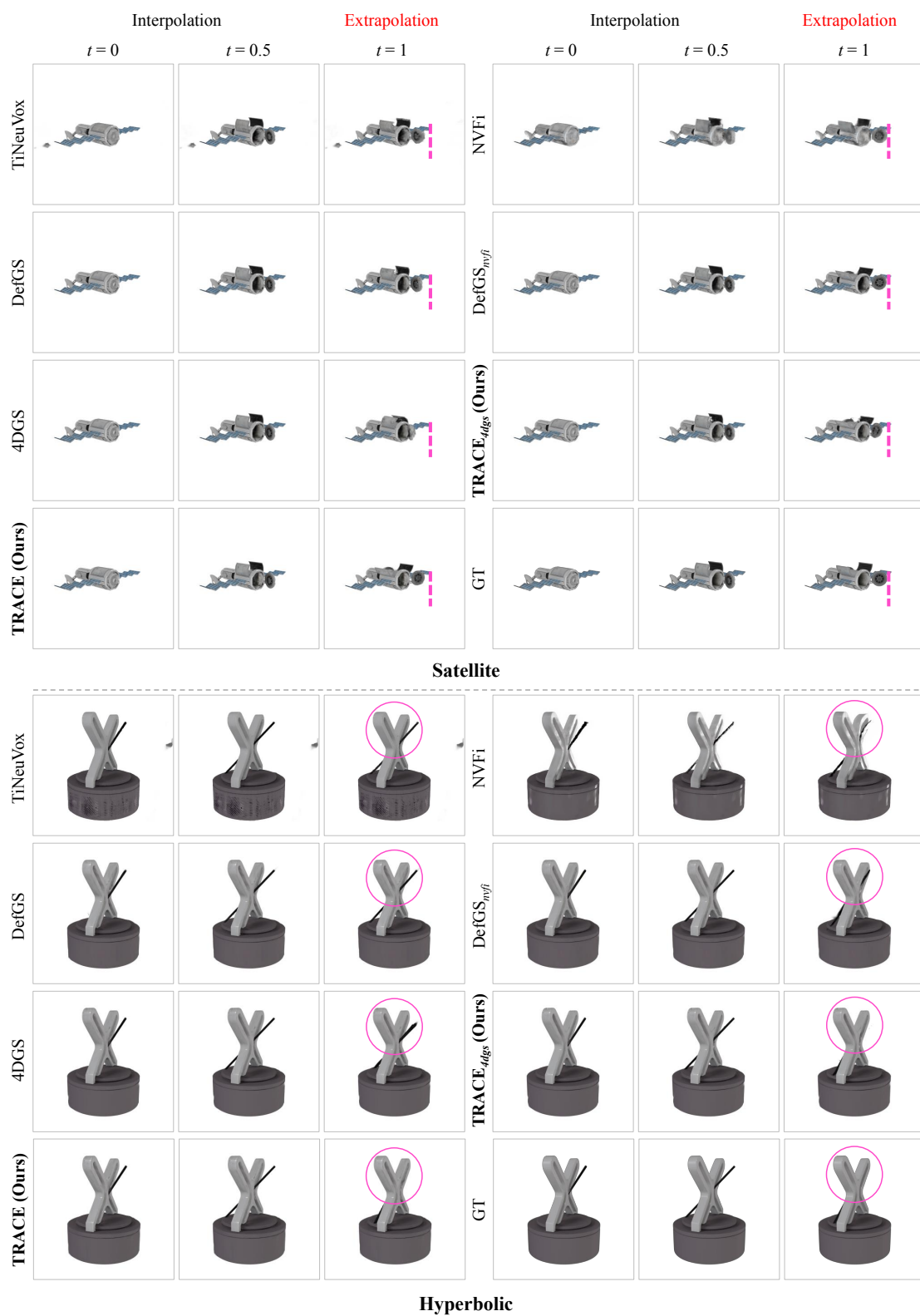


Figure 18. Qualitative results of RGB view synthesis for interpolation and **extrapolation** tasks on Dynamic Multipart dataset.



Figure 19. Qualitative results of RGB view synthesis for interpolation and **extrapolation** tasks on “Skating” scene of NVIDIA Dynamic Scene dataset.

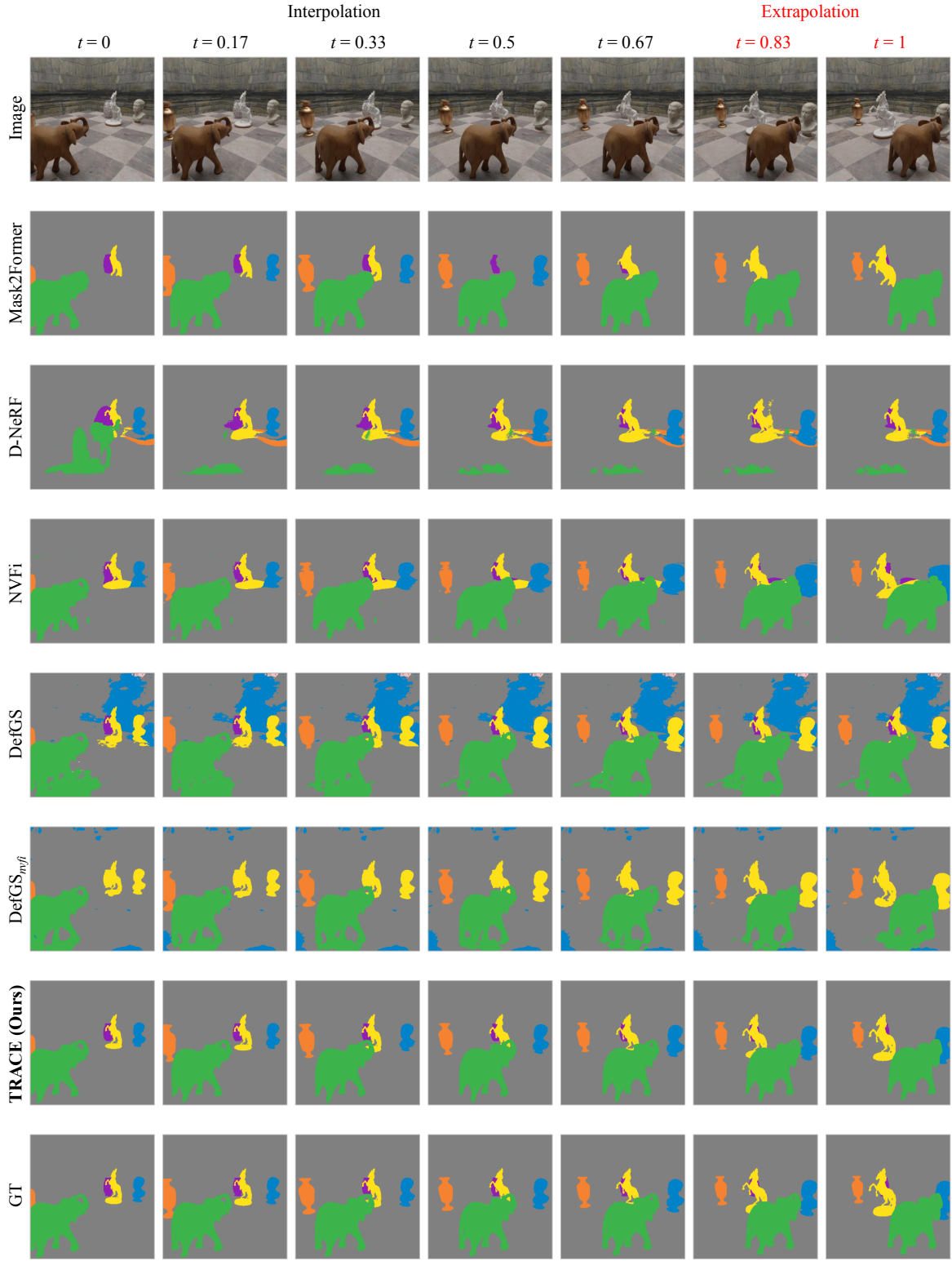


Figure 20. Qualitative results for object segmentation on “Chessboard” of Dynamic Indoor Scene dataset.

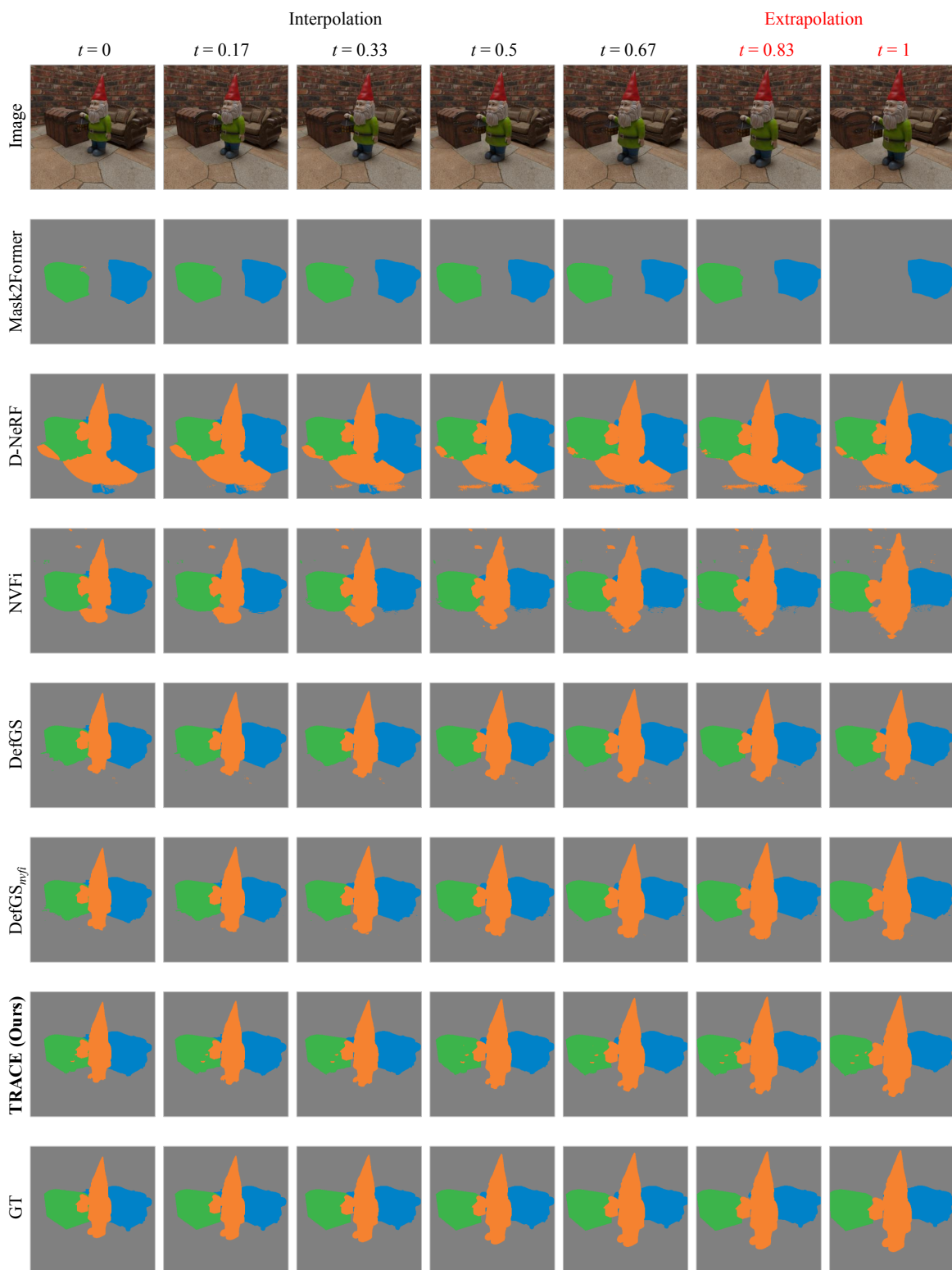


Figure 21. Qualitative results for object segmentation on “Gnome House” of Dynamic Indoor Scene dataset.

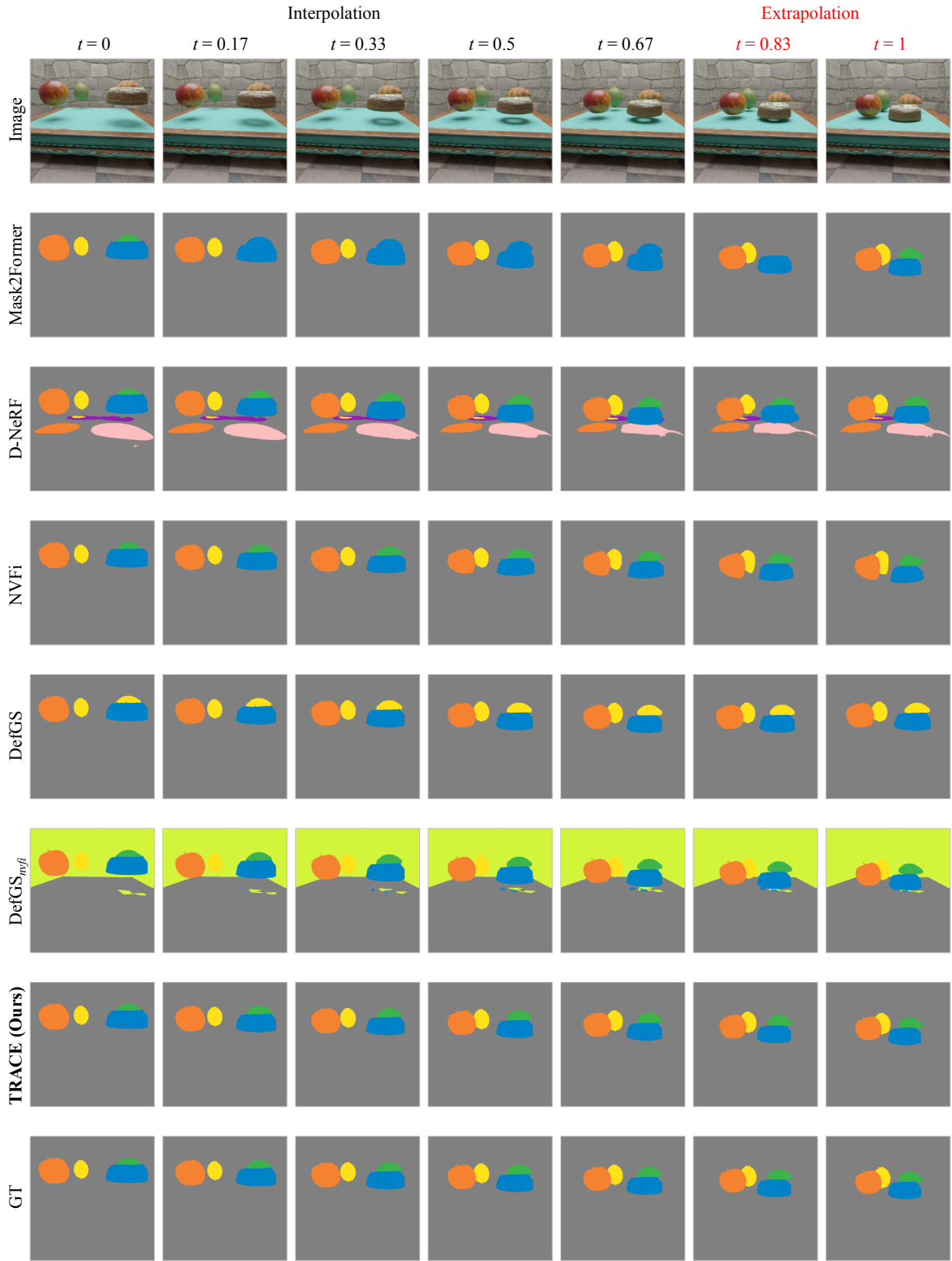


Figure 22. Qualitative results for object segmentation on “Dining Table” of Dynamic Indoor Scene dataset.

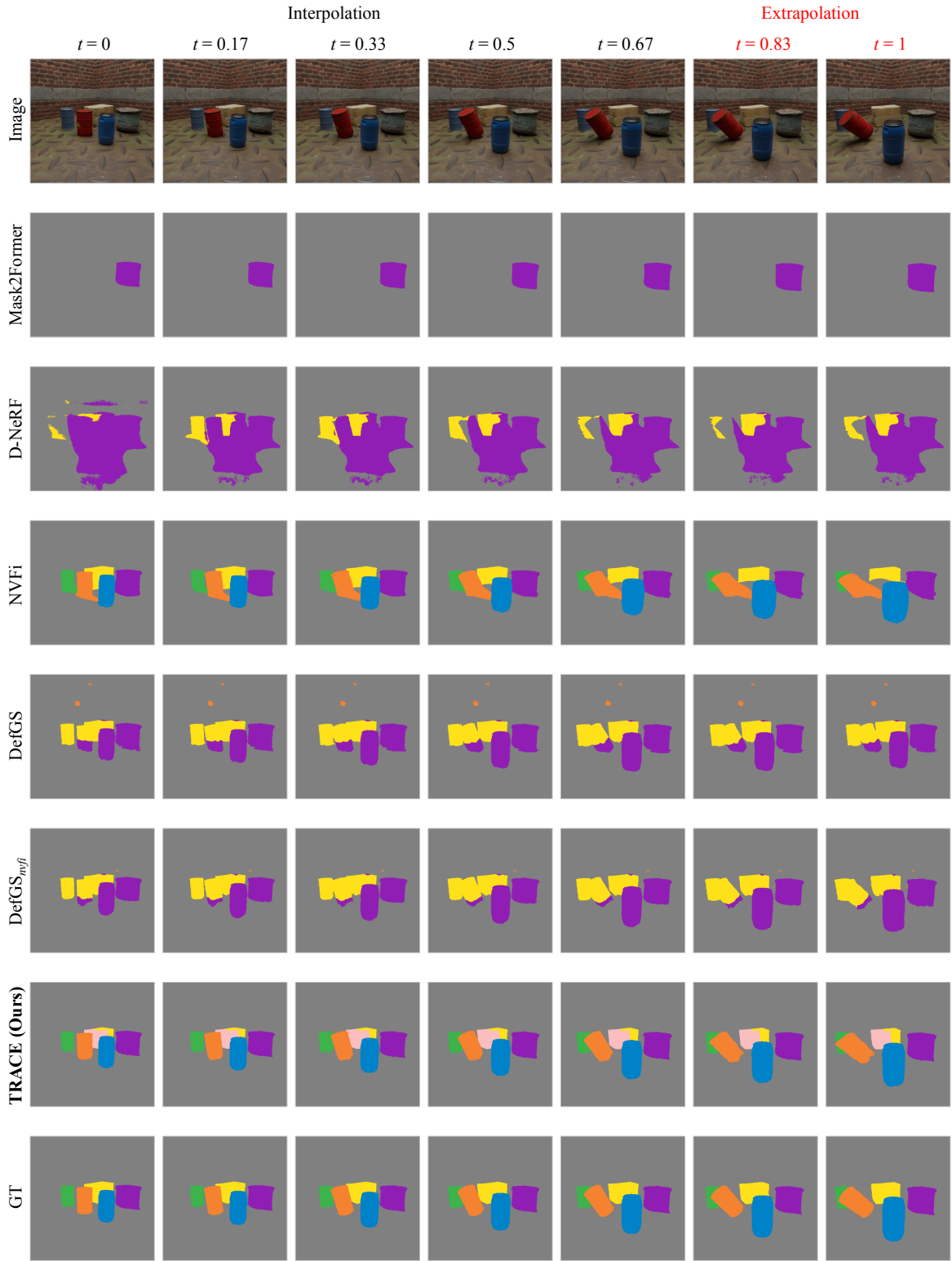


Figure 23. Qualitative results for object segmentation on “Factory” of Dynamic Indoor Scene dataset.

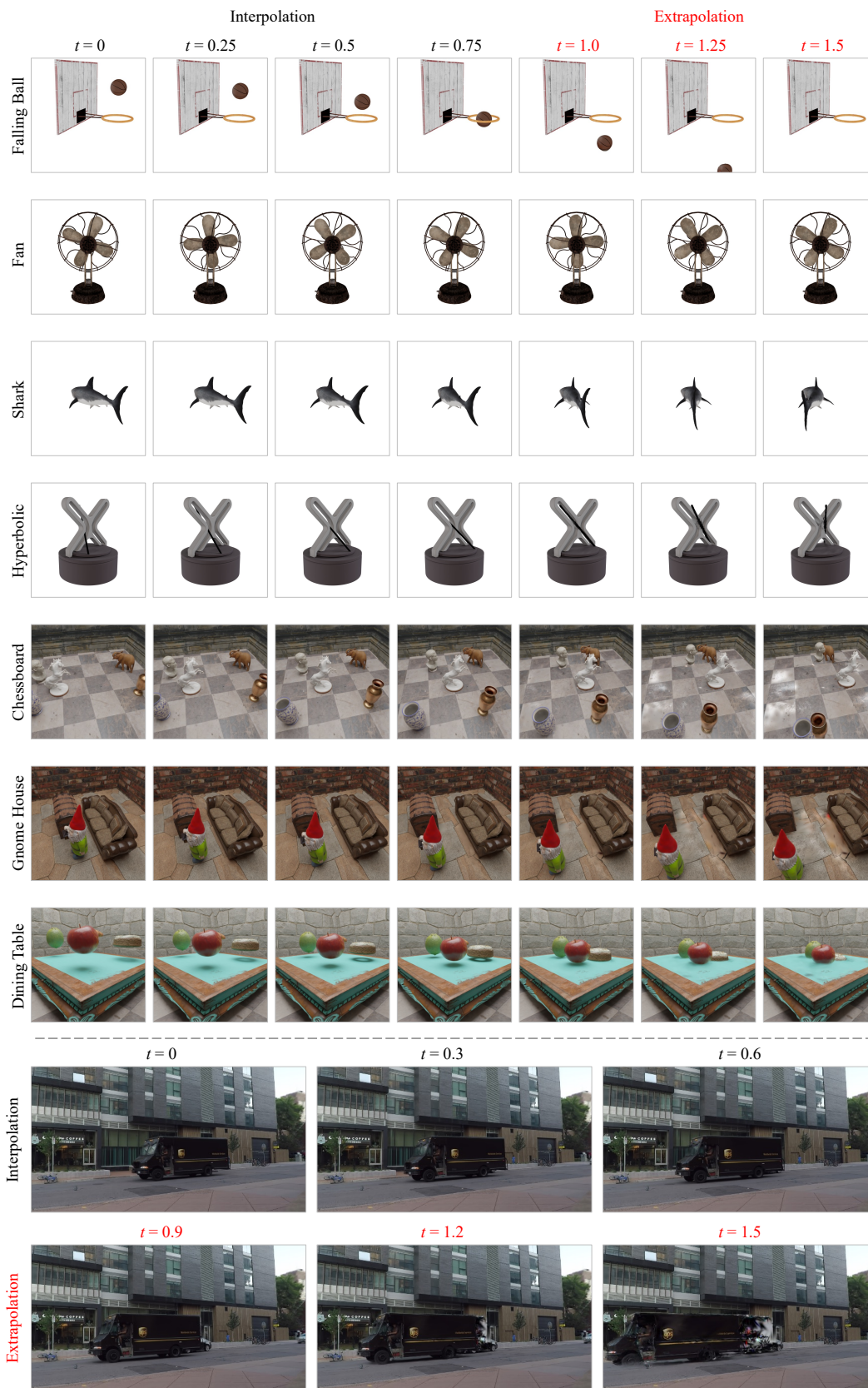


Figure 24. Qualitative results of RGB view synthesis for longer extrapolation from our method.