

ParticleGS: Learning Neural Gaussian Particle Dynamics from Videos for Prior-free Physical Motion Extrapolation

Supplementary Material

1. Appendix

The appendix includes:

- A Toy Experiment: Learning Dynamics vs. Memorizing Trajectories.
- Implementation Details.
- Additional Qualitative Comparisons.
- Full Quantitative Results.

2. A Toy Experiment: Learning Dynamics vs. Memorizing Trajectories

To intuitively understand the limitations of time-conditioned representations and the necessity of our physics-based approach, we conducted a toy experiment on a 2D damped harmonic oscillator system (a spiral trajectory), as shown in Fig. 1. We trained two models on the initial segment of the spiral (corrupted by random noise, indicated by green dots) and tested their ability to predict the future evolution (indicated by red dots).

Analysis of Time-Conditioned Models. As illustrated in Fig. 1(a), the baseline method, which models position as a direct function of time ($x = \mathcal{F}(t)$), successfully interpolates the training points. However, it fails to extrapolate to the future. This is because the baseline essentially performs curve fitting on the time coordinate, memorizing the correlation between a specific timestamp t and a position x within the observed window. Once t extends beyond this range, the network lacks the underlying physical context to predict the future, reverting to linear behavior that violates the system’s inherent dynamics.

Analysis of Neural ODEs. In contrast, Fig. 1(b) demonstrates that the Neural ODE-based approach accurately predicts the future spiral motion. This success stems from the fact that Neural ODEs do not map time directly to position. Instead, they approximate the governing equation of the system: $\frac{dx}{dt} = f(x, t)$. While the position $x(t)$ changes constantly, the physical law f governing the change remains consistent. By learning this derivative function from the observed past, the model can integrate the same law to generate physically plausible future states. This observation motivates ParticleGS to move from memorizing temporal deformations to learning the physical laws driving the system.

3. Implementation Details

Input Representation. To capture the comprehensive state of each Gaussian particle, we construct a 14-dimensional feature vector as the input to our network. Specifically, this

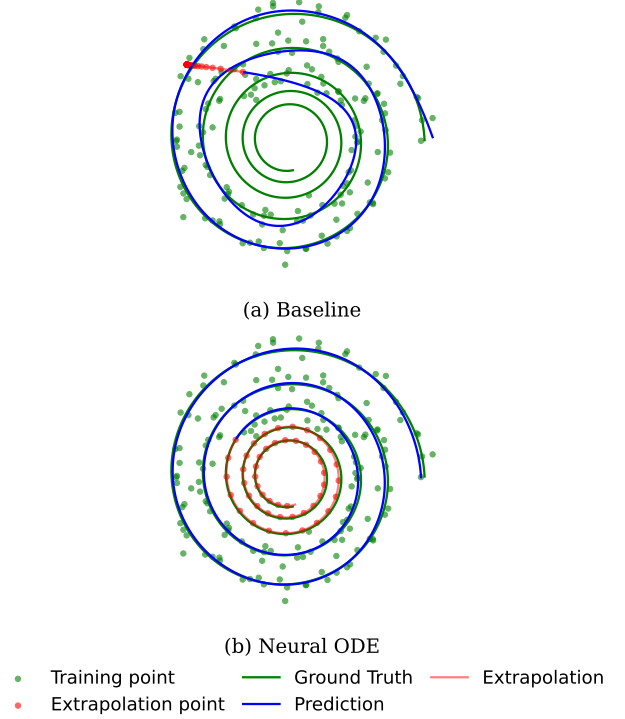


Figure 1. **Toy experiment on extrapolation capabilities.** We visualize the predicted trajectories on a 2D damped spiral. Green dots represent observed training data, while red dots indicate the unobserved future. **(a) Baseline:** The time-conditioned baseline fits the training data well but fails during extrapolation, producing a linear trajectory that deviates from the physical spiral. **(b) Neural ODE:** By modeling the underlying derivatives, the Neural ODE accurately captures the spiral dynamics, maintaining the correct trajectory even in the unobserved region.

vector is a concatenation of the position $x \in \mathbb{R}^3$, rotation quaternion $r \in \mathbb{R}^4$, scaling $s \in \mathbb{R}^3$, opacity $\alpha \in \mathbb{R}^1$, and the 0-th order of the Spherical Harmonics coefficients $sh \in \mathbb{R}^3$. This compact representation enables the encoder to extract both geometric and photometric cues effectively.

Dynamics Latent Space Encoder f_{encoder} : The encoder is designed to extract particle-level static features and system-level dynamic fields. It consists of 4 stacked attention layers, each with a hidden dimension of 256, employing GELU as the activation function. To handle the unstructured nature of 3D Gaussian clouds, we construct local Gaussian patches using Farthest Point Sampling (FPS) and k -nearest neighbors (k-NN). We employ FPS to select representative anchor points from the Gaussian cloud. For

synthetic datasets, we sample $N_k = 2048$ keypoints. For real-world datasets which typically contain more complex geometries and background noise, we increase the sampling density to $N_k = 4096$. For each keypoint, we query its $k = 32$ nearest neighbors (k-NN) based on Euclidean distance to form a local patch. To adapt to the changing Gaussian distribution, we regenerate these neighborhood patches immediately after every densification step or at a fixed interval of 500 iterations.

This hierarchical design ensures efficient information aggregation across the scene.

Neural ODE Dynamics Evolver f_{evolver} : The evolver is the core component for modeling continuous-time dynamics. We implement f_{evolver} using a network composed of two stacked 3-layer MLP blocks equipped with residual connections to facilitate gradient flow. All hidden layers maintain a dimension of 128 and use GELU activation. We utilize the Runge-Kutta 4 (RK4) method as the ODE solver to integrate the learned derivative function.

Gaussian Kernel Space Decoder f_{decoder} : We use a 6-layer MLP with a hidden dimension of 256 for all layers, using GELU as the activation function.

Physical State Dimensions. We set the dimension of the static features L to 16 and the dimension of each dynamic field G to 16.

Optimization. We train our framework using the Adam optimizer and adopt a progressive training strategy. The total training iterations are set to 40,000 for synthetic datasets and 60,000 for real-world datasets. For 0-3k iterations: We adopt Geometry Warm-up and optimize only the static Gaussian parameters to obtain a reliable canonical geometry. For 3k-7k iterations: We adopt Dynamics Warm-up, where we freeze the canonical Gaussians and optimize the ParticleGS network with a gradually increasing temporal window. For 7k-40k/60k iterations: Both the canonical Gaussians and the neural networks are jointly optimized to refine fine-grained details until the end of training.

4. Rodrigues' Rotation Formula

In our framework, the decoder predicts a rotation vector $R \in \mathbb{R}^3$ (representing the axis-angle notation) to model the rotational dynamics. To apply this rotation to the 3D Gaussians, we convert R into a valid rotation matrix $\text{Rod}(R) \in \mathbb{R}^{3 \times 3}$ using Rodrigues' rotation formula. Given the rotation vector $R = [r_x, r_y, r_z]^T$, the rotation angle θ is defined as its Euclidean norm:

$$\theta = \|R\|_2. \quad (1)$$

If θ is non-zero, the normalized rotation axis $K = [k_x, k_y, k_z]^T$ is computed as $K = R/\theta$. We first define the skew-symmetric matrix $[K]_{\times}$ associated with the unit

vector K :

$$[K]_{\times} = \begin{bmatrix} 0 & -k_z & k_y \\ k_z & 0 & -k_x \\ -k_y & k_x & 0 \end{bmatrix}. \quad (2)$$

Rodrigues' rotation formula then expresses the rotation matrix $\text{Rod}(R)$ as:

$$\text{Rod}(R) = I + \sin \theta [K]_{\times} + (1 - \cos \theta) [K]_{\times}^2, \quad (3)$$

where I is the 3×3 identity matrix.

5. Additional Qualitative Comparisons

We provide additional visual comparisons across various datasets in Fig. 2 through Fig. 7. The **red bounding boxes** in the figures mark the regions with ground truth motion, serving as a visual reference for the object's actual position and movement trajectory. The **bottom section** of each sub-figure displays details of specific regions.

6. Full Quantitative Results

We report the complete quantitative results for both reconstruction and extrapolation tasks. Detailed metrics, including PSNR, SSIM, and LPIPS, are provided for every scene in the datasets to ensure a thorough evaluation.

6.1. Full Results on Dynamic Object Dataset

Table 1 presents the detailed quantitative comparisons on the Dynamic Object Dataset, which includes scenes such as Falling Ball, Bat, Fan, Telescope, Shark, and Whale.

6.2. Full Results on Dynamic Indoor Scene Dataset

The results for the Dynamic Indoor Scene Dataset are summarized in Table 2. This dataset features complex background geometries and lighting conditions, represented by scenes like Gnome House, Chessboard, Factory, and Dining Table.

6.3. Full Results on Dynamic Multipart Dataset

We also evaluate our method on the Dynamic Multipart Dataset, which includes objects exhibiting distinct part-level motion patterns, such as Folding Chair, Satellite, Stove, and Hyperbolic. The full quantitative results are shown in Table 3.

6.4. Full Results on FreeGave-GoPro Dataset

Finally, Table 4 details the performance on the FreeGave-GoPro Dataset, covering real-world captured scenes such as Box, Collision, Wrist Rest, Hammer, and Pen.

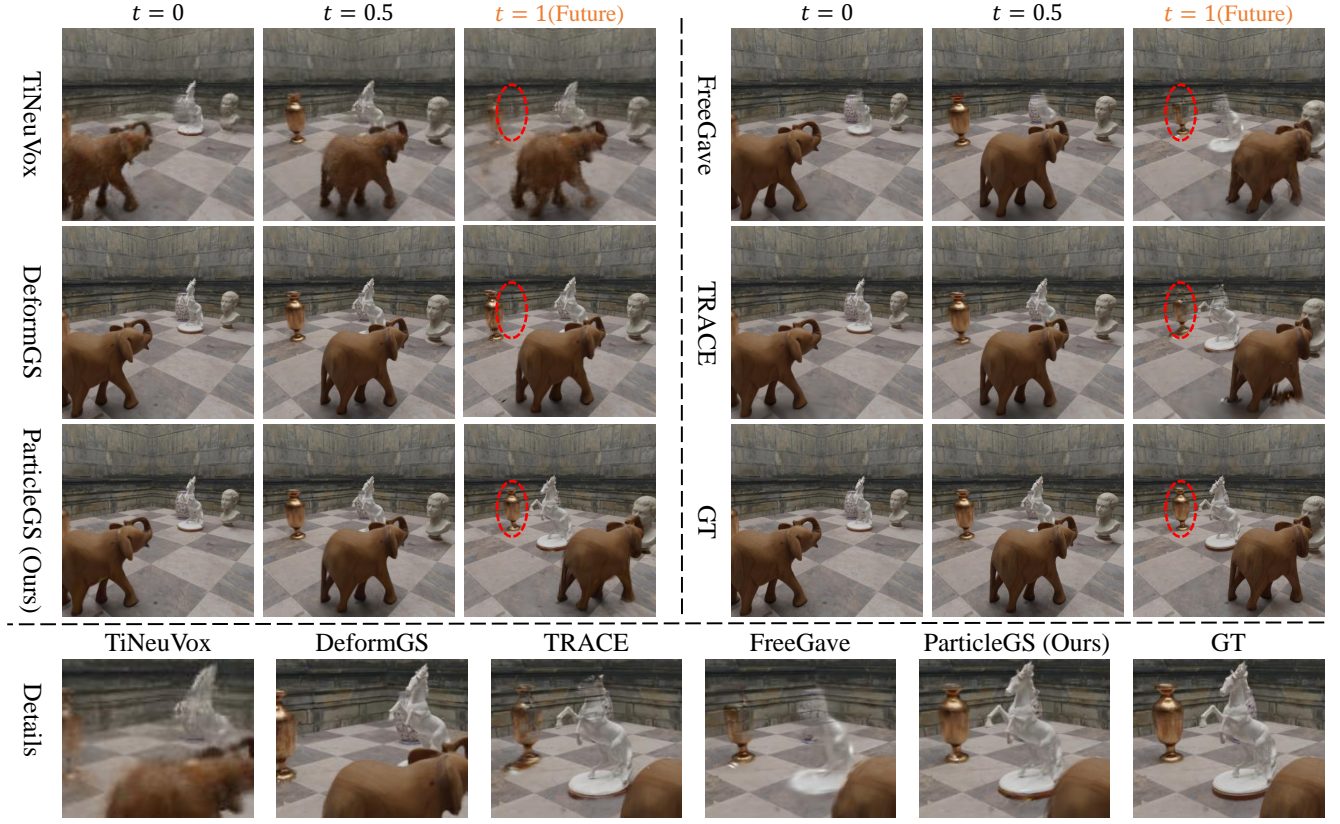


Figure 2. Additional qualitative comparisons on the Dynamic Indoor Scene Dataset.

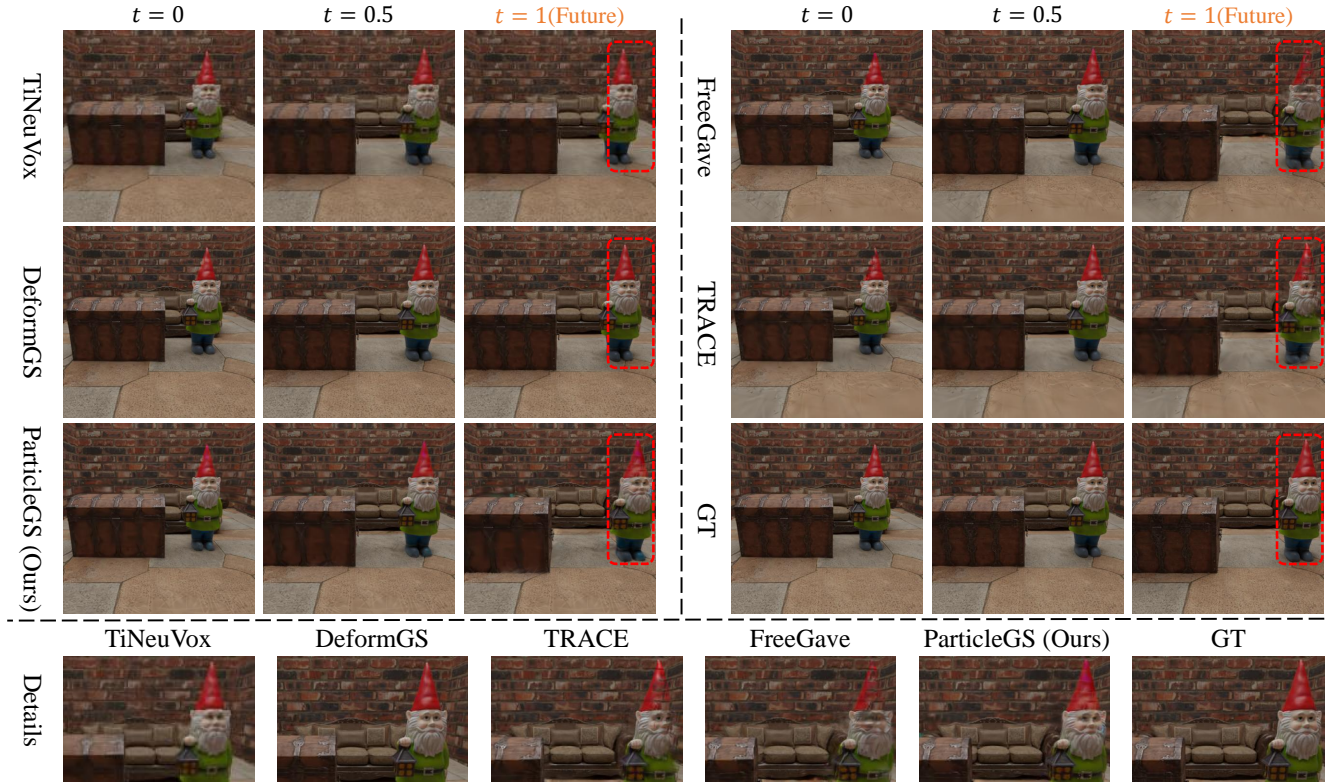


Figure 3. Additional qualitative comparisons on the Dynamic Indoor Scene Dataset.

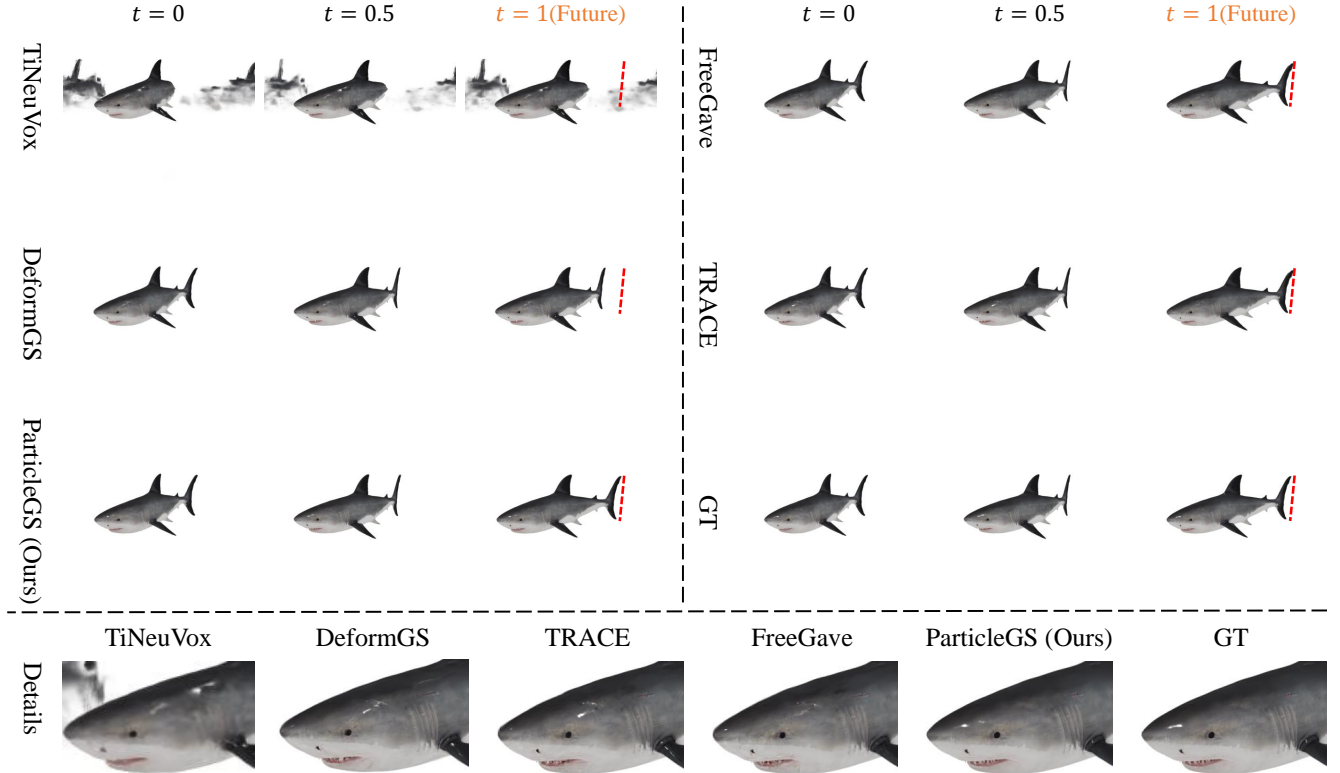


Figure 4. Additional qualitative comparisons on the Dynamic Object Dataset.

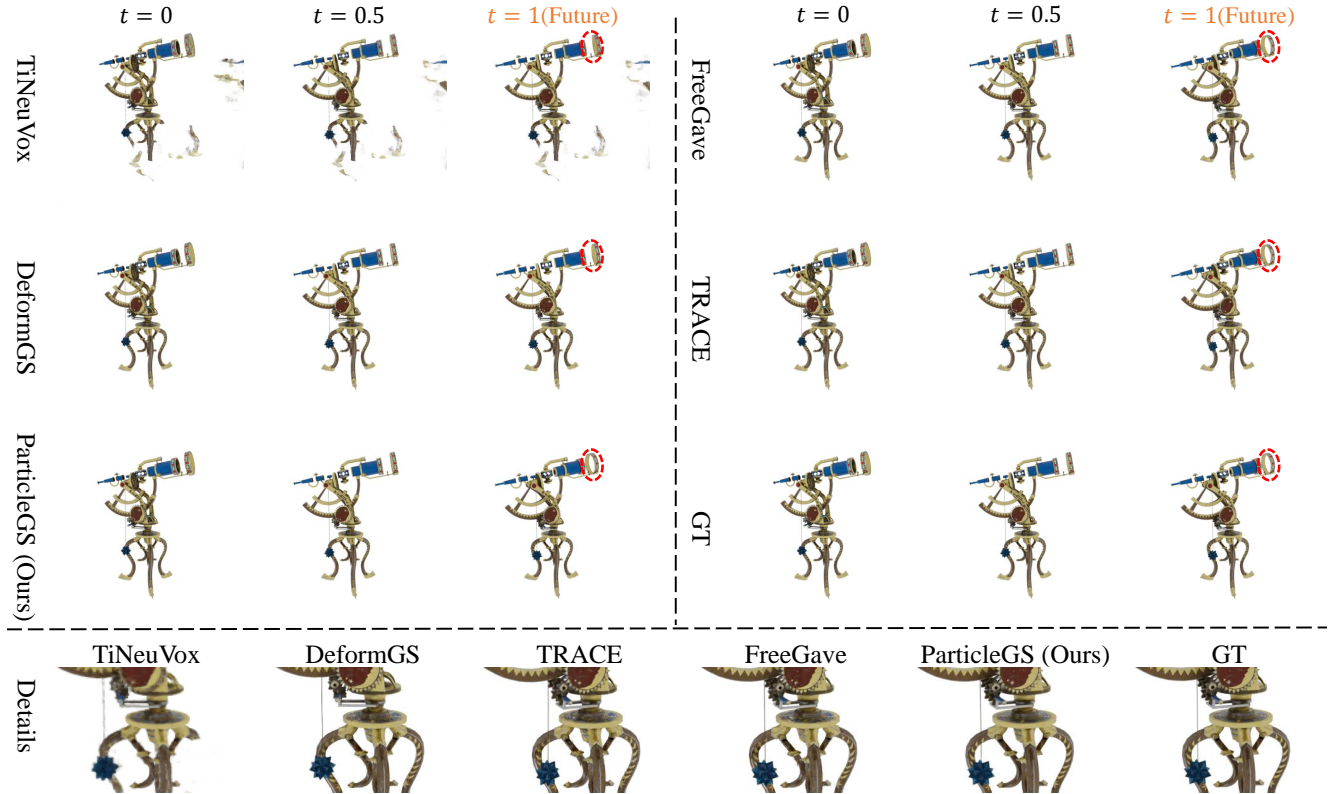


Figure 5. Additional qualitative comparisons on the Dynamic Object Dataset.

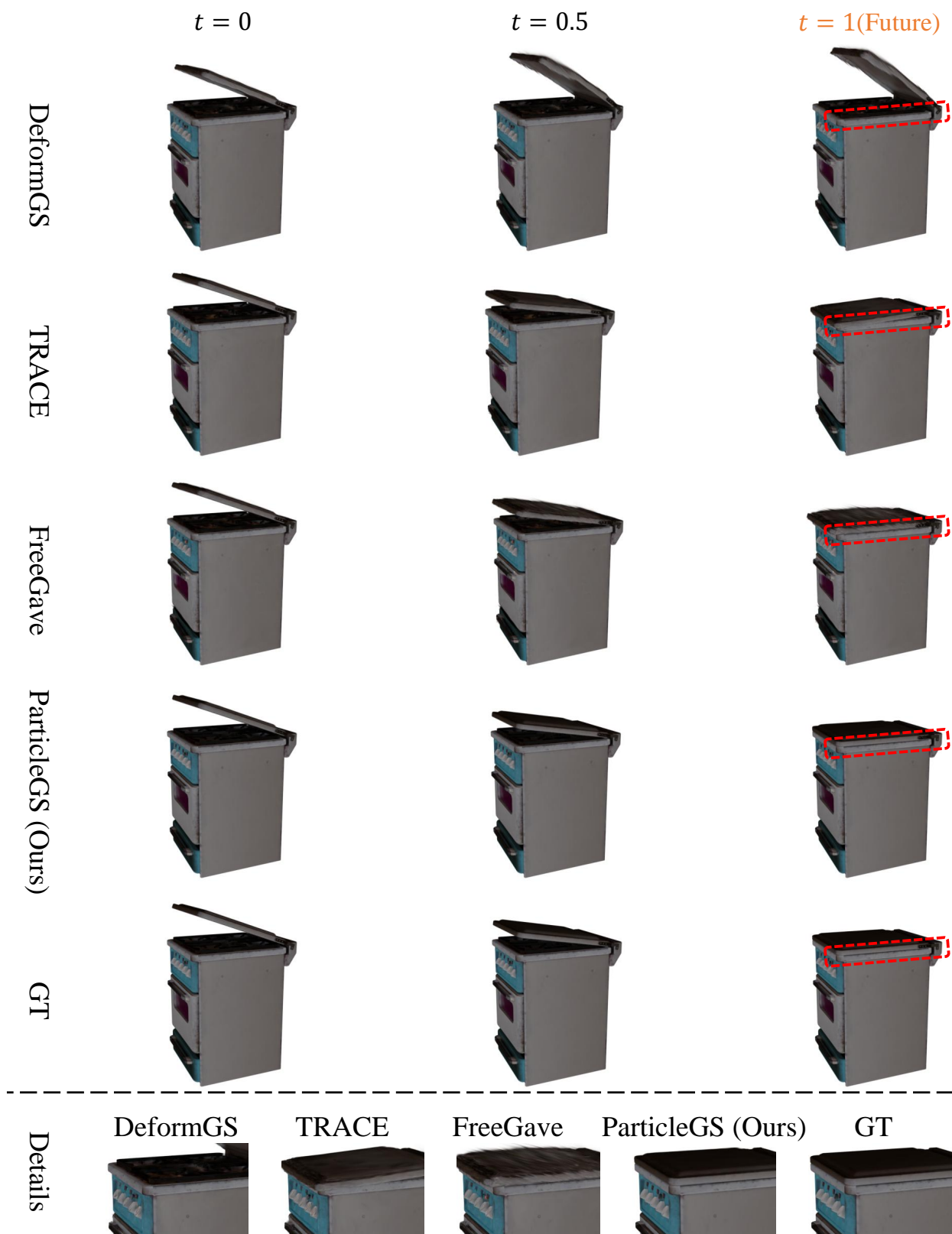


Figure 6. Additional qualitative comparisons on the Dynamic Multipart Dataset.

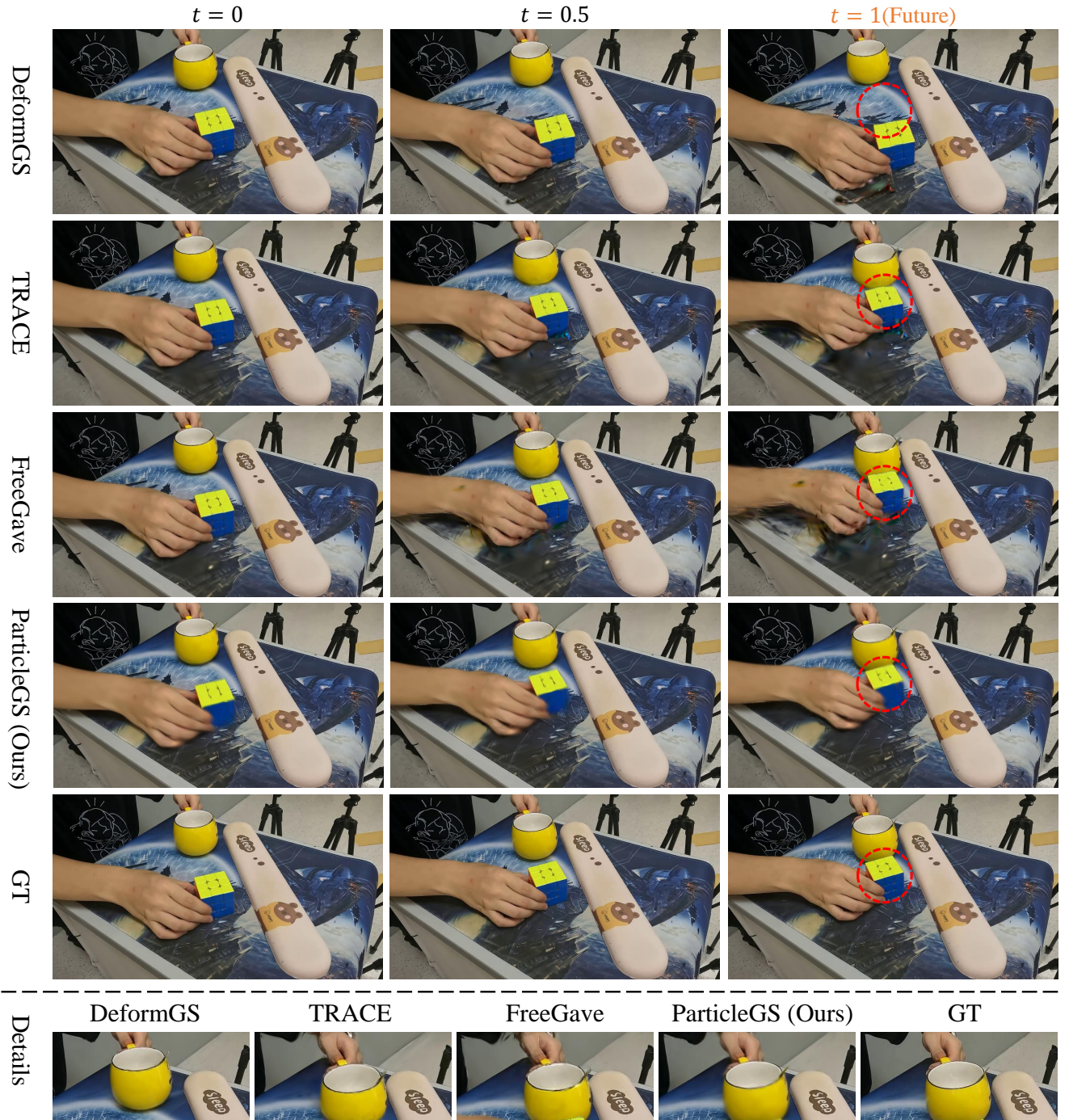


Figure 7. Additional qualitative comparisons on the FreeGave-GoPro Dataset

Table 1. Full quantitative results on the Dynamic Objects, rendered at the source resolution. **Bold** and underline indicate the best and second best performance.

Method	Falling Ball						Bat					
	Reconstruction			Extrapolation			Reconstruction			Extrapolation		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
HexPlane	19.754	0.816	0.215	20.166	0.906	0.130	22.099	0.953	0.080	22.422	0.954	0.061
TiNeuVox	24.323	0.894	0.167	20.474	0.940	0.098	17.611	0.946	0.076	17.139	0.947	0.070
DeformGS	17.736	0.830	0.156	26.384	0.945	0.060	<u>47.034</u>	<u>0.996</u>	0.004	27.109	0.944	0.034
Grid4D	19.119	0.821	0.153	24.926	0.925	0.069	47.443	0.997	0.004	29.551	0.953	0.290
NVFi	34.900	0.969	0.068	27.013	0.966	0.074	24.648	0.970	0.044	25.529	0.973	0.040
GSPrediction	24.549	0.910	0.154	16.790	0.830	0.238	43.290	0.997	0.002	23.495	0.952	0.075
TRACE	<u>41.077</u>	<u>0.995</u>	0.017	38.126	<u>0.993</u>	0.020	39.269	0.995	0.006	28.890	<u>0.982</u>	0.016
FreeGave	42.248	0.996	<u>0.015</u>	37.460	0.994	<u>0.015</u>	39.516	0.995	0.006	<u>30.907</u>	0.980	<u>0.014</u>
ParticleGS	38.275	0.991	0.010	<u>38.067</u>	0.990	0.013	43.232	0.997	<u>0.003</u>	36.348	0.984	0.011

Method	Fan						Telescope					
	Reconstruction			Extrapolation			Reconstruction			Extrapolation		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
HexPlane	18.424	0.842	0.190	19.998	0.887	0.106	21.814	0.917	0.149	23.667	0.931	0.082
TiNeuVox	15.775	0.855	0.189	21.307	0.919	0.081	23.088	0.949	0.084	20.801	0.925	0.071
DeformGS	<u>38.083</u>	0.973	0.026	24.473	0.930	0.047	38.435	0.992	<u>0.005</u>	23.186	0.934	0.037
Grid4D	37.031	0.967	0.030	26.720	0.941	0.044	38.488	0.992	<u>0.006</u>	25.696	0.942	0.033
NVFi	27.748	0.953	0.059	27.347	0.957	0.054	26.889	0.955	0.051	26.861	0.955	0.053
GSPrediction	34.864	<u>0.976</u>	0.020	20.747	0.935	0.054	36.191	0.992	0.004	22.018	0.939	0.043
TRACE	34.038	0.975	0.023	36.779	0.984	0.014	38.839	0.994	0.006	35.725	<u>0.980</u>	0.006
FreeGave	34.158	0.975	0.022	33.987	<u>0.974</u>	<u>0.016</u>	38.737	0.994	0.006	<u>36.407</u>	0.991	<u>0.005</u>
ParticleGS	38.852	0.979	<u>0.021</u>	<u>34.235</u>	<u>0.974</u>	0.023	<u>38.770</u>	<u>0.993</u>	0.004	38.294	0.991	0.004

Method	Shark						Whale					
	Reconstruction			Extrapolation			Reconstruction			Extrapolation		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
HexPlane	22.736	0.956	0.064	24.180	0.962	0.041	24.684	0.976	0.049	24.075	0.964	0.049
TiNeuVox	19.592	0.951	0.058	16.683	0.933	0.096	21.281	0.969	0.052	20.671	0.945	0.063
DeformGS	<u>40.492</u>	0.995	<u>0.008</u>	29.159	0.966	0.022	<u>42.475</u>	0.996	0.004	26.666	0.959	0.027
Grid4D	40.752	0.995	0.007	30.904	0.966	0.021	42.666	0.996	0.004	28.246	0.967	0.020
NVFi	31.374	<u>0.982</u>	0.035	28.649	0.979	0.033	30.778	0.984	0.025	25.529	0.977	0.029
GSPrediction	40.015	0.995	0.010	30.280	0.971	0.032	39.540	0.996	<u>0.005</u>	25.596	0.965	0.046
TRACE	38.637	0.995	0.009	30.547	<u>0.982</u>	0.012	36.177	0.994	0.006	30.068	<u>0.980</u>	<u>0.013</u>
FreeGave	39.378	0.995	0.009	<u>31.639</u>	0.979	<u>0.011</u>	37.788	0.994	0.006	<u>31.395</u>	<u>0.980</u>	0.012
ParticleGS	40.322	0.995	<u>0.008</u>	37.475	0.983	0.010	39.241	<u>0.995</u>	<u>0.005</u>	34.236	0.983	0.012

Table 2. Full quantitative results on Dynamic Indoor Scene dataset, rendered at the source resolution. **Bold** and underline indicate the best and second best performance.

Method	Gnome House						Chessboard					
	Reconstruction			Extrapolation			Reconstruction			Extrapolation		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
HexPlane	18.060	0.419	0.650	23.341	0.601	0.530	18.054	0.515	0.617	22.034	0.680	0.500
TiNeuVox	24.644	0.663	0.385	21.359	0.665	0.357	<u>23.847</u>	<u>0.698</u>	0.392	20.088	0.676	0.408
DeformGS	21.855	0.611	<u>0.341</u>	22.728	0.731	<u>0.196</u>	20.688	0.659	0.443	20.484	0.773	0.280
Grid4D	22.418	0.588	0.360	21.185	0.696	<u>0.258</u>	21.545	0.667	0.437	19.713	0.741	0.331
NVFi	16.857	0.435	0.687	24.087	0.615	0.546	17.515	0.551	0.651	22.571	0.672	0.539
GSPrediction	16.113	0.556	0.632	20.098	0.641	0.371	17.665	0.651	0.470	20.033	0.724	0.317
TRACE	22.968	0.642	0.351	<u>29.325</u>	<u>0.840</u>	0.209	22.969	0.642	<u>0.352</u>	<u>29.326</u>	0.840	<u>0.209</u>
FreeGave	19.178	0.543	0.425	28.962	0.827	0.217	19.905	0.674	0.471	26.339	<u>0.862</u>	0.251
ParticleGS	<u>24.481</u>	<u>0.662</u>	0.307	31.167	0.843	0.160	25.449	0.785	0.287	29.529	0.898	0.153

Method	Factory						Dining Table					
	Reconstruction			Extrapolation			Reconstruction			Extrapolation		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
HexPlane	19.109	0.479	0.637	24.266	0.677	0.470	16.687	0.516	0.601	23.127	0.725	0.395
TiNeuVox	<u>24.857</u>	<u>0.705</u>	0.346	22.467	0.747	0.292	<u>21.476</u>	<u>0.658</u>	0.368	20.529	0.768	0.276
DeformGS	21.703	0.621	0.376	23.471	0.787	<u>0.196</u>	15.832	0.488	0.621	21.244	0.784	0.258
Grid4D	21.822	0.631	0.377	22.119	0.765	0.235	16.407	0.475	0.595	19.516	0.741	0.313
NVFi	18.889	0.503	0.655	25.415	0.690	0.496	15.206	0.485	0.681	22.790	0.711	0.428
GSPrediction	17.388	0.588	0.609	21.723	0.713	0.308	13.451	0.546	0.688	18.214	0.679	0.374
TRACE	22.012	0.581	0.426	27.177	0.817	0.234	23.434	0.705	<u>0.380</u>	<u>32.078</u>	0.900	0.169
FreeGave	19.448	0.498	0.485	<u>28.190</u>	<u>0.827</u>	0.221	20.194	0.643	0.433	32.443	0.900	<u>0.159</u>
ParticleGS	31.627	0.912	0.126	32.679	0.943	0.077	20.461	0.636	0.444	31.037	<u>0.886</u>	0.143

Table 3. Full quantitative results on the Dynamic Multipart dataset, rendered at the source resolution. **Bold** and underline indicate the best and second best performance.

Method	Folding Chair						Satellite					
	Reconstruction			Extrapolation			Reconstruction			Extrapolation		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
DeformGS	<u>41.562</u>	0.994	0.009	18.194	0.847	0.123	35.884	0.988	0.010	29.454	0.980	0.014
FreeGave	39.086	0.994	<u>0.010</u>	26.043	0.951	0.037	<u>37.155</u>	0.991	0.010	34.476	0.989	0.007
TRACE	38.286	<u>0.993</u>	0.011	<u>28.724</u>	<u>0.974</u>	<u>0.023</u>	36.647	<u>0.990</u>	0.010	33.612	<u>0.983</u>	<u>0.009</u>
ParticleGS	41.620	0.994	0.009	31.512	0.977	0.015	38.697	0.991	0.010	34.562	0.981	0.017

Method	Stove						Hyperbolic					
	Reconstruction			Extrapolation			Reconstruction			Extrapolation		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
DeformGS	40.169	0.988	<u>0.027</u>	30.694	0.953	0.030	36.105	0.983	0.031	33.614	0.987	0.015
FreeGave	35.578	<u>0.989</u>	0.028	32.291	<u>0.989</u>	0.021	32.747	0.985	0.034	41.294	0.995	0.011
TRACE	36.201	0.990	0.028	<u>33.869</u>	<u>0.989</u>	<u>0.016</u>	33.942	0.985	0.036	37.628	<u>0.990</u>	<u>0.012</u>
ParticleGS	<u>39.115</u>	<u>0.989</u>	0.024	40.069	0.991	0.013	<u>34.998</u>	<u>0.984</u>	<u>0.032</u>	<u>38.399</u>	<u>0.990</u>	<u>0.012</u>

Table 4. Full quantitative results on the FreeGave-GoPro dataset, rendered at the source resolution. **Bold** and underline indicate the best and second best performance.

Method	Box						Collision					
	Reconstruction			Extrapolation			Reconstruction			Extrapolation		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
DeformGS	19.505	0.810	0.221	25.754	0.900	0.139	<u>20.012</u>	0.840	0.209	18.986	0.810	0.251
TRACE	20.203	0.833	0.195	27.987	0.912	<u>0.131</u>	20.242	0.841	0.209	23.447	0.858	0.193
FreeGave	19.800	0.830	<u>0.197</u>	<u>28.661</u>	<u>0.913</u>	<u>0.131</u>	19.608	<u>0.840</u>	0.212	<u>24.188</u>	<u>0.867</u>	<u>0.187</u>
ParticleGS	<u>20.122</u>	<u>0.831</u>	0.195	29.057	0.919	0.102	19.759	0.841	<u>0.211</u>	24.239	0.901	0.157

Method	Wrist Rest						Hammer					
	Reconstruction			Extrapolation			Reconstruction			Extrapolation		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
DeformGS	19.498	0.829	0.222	19.007	0.808	0.244	20.067	0.835	0.198	23.992	0.896	0.152
TRACE	19.079	<u>0.833</u>	<u>0.209</u>	22.488	0.845	<u>0.204</u>	20.077	0.842	0.195	28.924	0.919	<u>0.137</u>
FreeGave	19.807	0.835	0.208	23.456	0.853	0.197	20.407	<u>0.840</u>	<u>0.194</u>	28.521	<u>0.917</u>	<u>0.137</u>
ParticleGS	<u>19.613</u>	0.835	0.208	<u>23.000</u>	<u>0.846</u>	0.208	<u>20.301</u>	0.842	0.193	<u>28.816</u>	<u>0.917</u>	0.136

Method	Pen1						Pen2					
	Reconstruction			Extrapolation			Reconstruction			Extrapolation		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
DeformGS	21.180	0.851	0.205	20.577	0.865	0.197	20.372	0.824	0.217	21.674	0.870	0.190
TRACE	20.977	<u>0.854</u>	0.205	26.703	<u>0.917</u>	0.149	<u>20.040</u>	<u>0.829</u>	0.214	25.939	0.899	0.167
FreeGave	20.130	0.853	<u>0.206</u>	<u>27.673</u>	0.923	<u>0.143</u>	19.968	0.830	<u>0.215</u>	<u>26.563</u>	<u>0.903</u>	<u>0.162</u>
ParticleGS	<u>21.079</u>	0.856	0.205	27.746	0.923	0.140	19.824	<u>0.829</u>	<u>0.215</u>	27.879	0.910	0.158