RetimeGS: Continuous-Time Reconstruction of 4D Gaussian Splatting

Supplementary Material

A. Hyperparameter Settings

The implementation adopts the hyperparameter configurations recommended by gsplat library [45]. Opacity regularization and scale regularization are enabled to encourage primitives to maintain low opacity and compact scale, with corresponding weights set to 0.01 and 0.1, respectively. The temporal opacity hyperparameter γ is set to 0.005 to promote a short-tail distribution. The MCMC strategy [12] relocates primitives every 100 iterations using a minimum opacity threshold of 0.01. The flow learning rate is initialized at 0.5 and decays exponentially toward 1×10^{-6} after $12{,}000$ iterations. For all Gaussian properties, we apply an exponential decay to the learning rates after $18{,}000$ iterations to encourage stable convergence at the end of training. Additionally, dynamic stretching is applied every $3{,}000$ iterations to adjust the temporal durations of the primitives. All scenes are trained for a total of $20{,}000$ iterations.

B. More Discussion of Limitations

As noted in Section 5, our method fails when the inter-frame motion is excessively large or when videos are captured at extremely low frame rates, causing off-the-shelf optical flow estimators to become unreliable for establishing coarse correspondences. As shown in Figure 8, in the fast-dancing sequence from the DNA-Rendering dataset [4], we reduce the frame rate to 7.5, effectively halving the original FPS. While our algorithm faithfully reconstructs frames i and $i+1$, where ground-truth supervision is available, the interpolated intermediate frame at $i+0.5$ exhibits noticeable artifacts due to the unreliable motion cues, which incorrectly associate part of the front leg in frame i with the back leg in frame $i{+}1$. A similar issue appears in our Stage-Capture dataset, where reducing the frame rate to approximately 7.33 leads to visible artifacts around the hand region in the intermediate frame at $i+0.33$. In practice, both FPS and physical motion speed contribute to inter-frame motion. Empirically, we find that for 1K videos, our method handles inter-frame motion up to around 50 pixels. A promising direction for future work is to incorporate stronger motion priors or adopt alternative 4D representations that can robustly operate under such conditions, effectively treating the task as a 4D multi-frame start-end interpolation problem.

At discrete input frames, although we independently supervise adjacent sets of dynamic primitives with the ground truth to enforce consistency and smoothly transition their temporal opacity, their inherently disjoint nature can still cause slight flickering artifacts. Addressing this with a unified 4D representation remains future work.

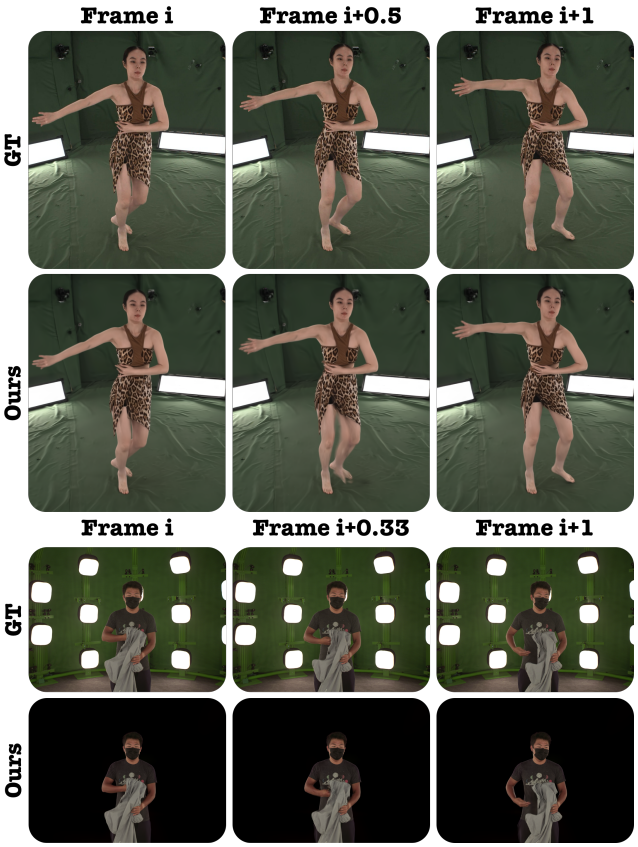

Figure 8. **Failure case under extremely low capture FPS.** Our method struggles to interpolate intermediate frames when the inter-frame motion becomes too large due to low temporal sampling or large motion.

C. More Experiment Results

C.1. Per Scene Breakdown

In Table 3, we provide the per-scene breakdown of the quantitative results on the Stage-Capture Dataset.

C.2. Video-Based Slow-Motion Comparison

The effectiveness of our method is clearly observed from the videos on the project page, where for each pair of input frames we interpolate 29 intermediate frames, slowing down the motion so that artifacts produced by other methods become more apparent and are otherwise difficult to capture in quantitative tables and qualitative static images.

In videos on the project page, all baseline methods, including 4D primitive–based approaches (with or without forward optical flow) [6, 18] and the 2D-to-3D lifted interpolation method [28], produce noticeable ghosting arti-

Table 3. **Per-scene quantitative comparisons on the Stage-Capture Dataset** (foreground region). Higher PSNR/SSIM and lower LPIPS are better.

Method	Scenes									Avg.
	Bear	Doll	Undress	Open Case	Pass Doll	Pickup Doll	Pack Computer	Stretch	Walk	
					PSNR ↑					
Deform-GS [39]	29.91	27.21	29.46	29.99	27.87	28.30	29.96	25.81	27.56	28.45
STGS [44]	22.96	23.49	28.41	29.09	24.07	23.44	28.75	23.42	24.46	25.34
GaussianFlow [6]	24.53	24.48	28.58	28.87	25.11	24.67	28.71	23.88	24.38	25.91
2D Lifting [28, 44]	28.60	26.60	30.23	31.16	27.97	28.37	30.09	27.21	28.82	28.79
Ours	31.27	27.99	30.30	30.54	29.89	30.64	30.17	29.96	29.99	30.08
					SSIM ↑					
Deform-GS [39]	0.877	0.872	0.845	0.897	0.831	0.873	0.900	0.861	0.849	0.867
STGS [44]	0.817	0.820	0.831	0.890	0.780	0.818	0.883	0.811	0.778	0.825
GaussianFlow [6]	0.824	0.826	0.833	0.887	0.789	0.827	0.884	0.806	0.753	0.825
2D Lifting [28, 44]	0.880	0.875	0.875	0.916	0.859	0.891	0.907	0.888	0.878	0.886
Ours	0.905	0.899	0.878	0.914	0.877	0.912	0.912	0.925	0.911	0.904
					LPIPS ↓					
Deform-GS [39]	0.0329	0.0221	0.0347	0.0213	0.0481	0.0302	0.0148	0.0239	0.0171	0.0272
STGS [44]	0.0435	0.0303	0.0368	0.0235	0.0658	0.0449	0.0179	0.0337	0.0247	0.0357
GaussianFlow [6]	0.0411	0.0287	0.0362	0.0231	0.0601	0.0411	0.0176	0.0309	0.0259	0.0339
2D Lifting [28, 44]	0.0335	0.0229	0.0313	0.0190	0.0488	0.0298	0.0155	0.0228	0.0171	0.0267
Ours	0.0300	0.0197	0.0323	0.0190	0.0419	0.0210	0.0157	0.0118	0.0110	0.0225

facts for unseen temporal novel frames, especially in regions with large motion. For the 4D primitive–based methods [6, 18], these artifacts appear in all frames except those with input ground-truth supervision. For the 2D interpolation–to–3D method [28], ghosting occurs in all frames except the input-supervised frames and the directly interpolated middle frames. It is worth noting that in regions with small motion, these methods can indeed recover the correct primitive trajectories, as shown in several prior works. In some cases, because 2D interpolation effectively reduces the apparent inter-frame motion, the ghosting artifacts in unseen frames are also reduced, though not fully eliminated.

For the deformation-based method [39], as discussed in Section 4.2, using a single set of primitives to represent all frames in a dynamic scene enables it to capture a roughly correct global trajectory compared to STGS [18]. Nevertheless, in regions with fast motion, complex textures, or visibility changes, establishing detailed correspondences becomes challenging. As a result, we often observe parts of the scene being mapped to incorrect corresponding regions, leading to erroneous trajectories. Moreover, because the method relies solely on RGB cues to infer correspondences, resolving fine-grained textures becomes difficult, causing many detailed regions to appear blurry or distorted. This limitation can be observed in scenes where the intermediate frames appear visually correct overall but still achieve poor quantitative performance, as shown in Table 3, such as the Open Case scene.

C.3. Video-Based Trajectory Comparison

As discussed in Section 4.3, using a spline trajectory produces smoother results than using a linear trajectory. In addition to the quantitative results shown in Table 2, we include a video comparison on the project page featuring a circular motion, which clearly reveals the piecewise-linear artifacts that appear when the spline design is omitted.

C.4. Additional Analysis of Dynamic Stretching

We discussed the benefits of introducing dynamic stretching in Section 3.3.3: it prevents redundant representations of static objects across multiple primitive sets and allows more primitives to be allocated to dynamic regions. In addition, stretching the duration of static primitives offers another important advantage. Static regions that are covered by fewer camera views, and would therefore receive only sparse training signals, can instead accumulate supervision across multiple timesteps. This effectively increases their training signals, reduces flickering artifacts, and leads to more stable training.

In the main paper, Figure 6 provides a qualitative visualization of the automatically detected static and dynamic components. Here, we additionally present a quantitative analysis of this primitive reduction using an example scene (Figure 10) from the DNA-Rendering Dataset [4]. Train-

Table 4. Quantitative results on the Stage-Capture dataset, including the Ex4DGS baseline.

Method	PSNR↑	SSIM↑	LPIPS↓
Deform-GS [39]	28.45	0.867	0.0272
STGS [44]	25.34	0.825	0.0357
GaussianFlow [6]	25.91	0.825	0.0339
Ex4DGS [15]	25.95	0.811	0.0379
2D Lifting [28, 44]	28.79	0.886	0.0267
Ours	30.08	0.904	0.0225

Table 5. Quantitative evaluation on the Flame Steak and Flame Salmon scenes from the Neural3DV dataset.

Method	PSNR↑	SSIM↑	LPIPS↓
Deform-GS [39]	31.79	0.952	0.081
STGS [44]	32.52	0.959	0.079
GaussianFlow [6]	31.89	0.957	0.082
Ex4DGS [15]	31.06	0.919	0.094
2D Lifting [28, 44]	33.17	0.960	0.080
Ours	33.22	0.959	0.074

ing under a 1M primitive budget with our modified MCMC strategy, we find that approximately 88K primitives (9% of the total) are static Gaussians that span multiple frames. This brings the "temporal sum" of active Gaussians (i.e., the sum of all Gaussian temporal durations) to approximately 2.26M. Therefore, by applying dynamic stretching, our method reduces the effective number of primitives by a factor of $\mathbf{2.26}\times$.

C.5. Video Results for Full Ablation Study

On the project page, we additionally provide slow-motion playback videos for the remaining ablation experiments.

C.6. Additional Comparison with Ex4DGS

Since Ex4DGS [15] also explicitly models temporal opacity and interpolates motion, we include it as another baseline among 4D primitive-based methods. Ex4DGS parameterizes temporal opacity as a constant window with Gaussian fall-offs at the boundaries while interpolating motion parameters across keyframes. Crucially, it leaves this temporal opacity unregularized. Consequently, the Gaussian fall-offs can become excessively narrow and close together, leading to severe overfitting when temporal signals are sparse. We quantitatively outperform Ex4DGS on the Stage-Capture dataset (Table 4) and qualitatively demonstrate that it suffers from ghosting artifacts similar to those in STGS [18] in the next subsection (Figure 9). Note that we evaluate Ex4DGS directly using its public codebase, unlike STGS and Deform-GS [39], which benefit from integration into our framework with the MCMC strategy [12].

C.7. Results on Neural3DV Dataset

Because our method regularizes temporal opacity, its behavior in scenes with rapidly changing opacity, such as fire, warrants discussion. Furthermore, to evaluate its generalization as a general reconstruction method, we test it on non-stage-captured data. Specifically, we compare our approach against the main paper baselines and Ex4DGS [15] introduced in the previous section on the Flame Steak and Flame Salmon sequences from Neural3DV [17], which feature both complex opacity changes and non-stage environments. Following our standard protocol, we subsample the videos to 1/10th of their original frame rate (from 30 FPS to 3 FPS), train using all cameras, and evaluate on all held-out temporal novel frames, per our setting. As shown in Figure 9, although our method is not explicitly designed for rapidly changing opacity, it performs reasonably well on fire scenes and generalizes effectively. Table 5 provides the quantitative results; however, because these scenes are largely static, we believe the qualitative differences to be more revealing than the numerical metrics.

C.8. Discussion on Improved Optical Flow Quality

Our method is robust to small errors in the pseudo-GT and sometimes even improves upon it, likely by leveraging multi-view information. As shown in Figure 10, we observe improvements on fine-grained details, such as the hat tassel and the edges of the dress decorations, which are often blurry or inaccurate in the pseudo-ground truth estimations.

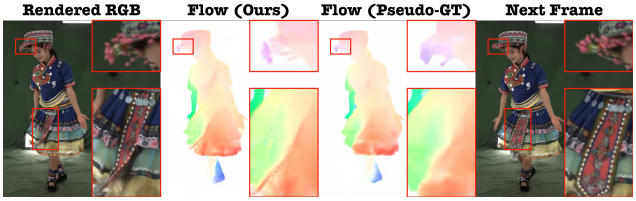

Figure 10. Comparison on rendered and pseudo-GT flow map.

C.9. Ablations on Other Optical Flow Methods

Although our method exhibits robustness to small errors in the pseudo-ground-truth, the quality of the initial pseudo-GT remains critical, as it provides the essential rough correspondences. Without sufficiently accurate initialization, optimization tends to become trapped in poor local minima. To investigate this, we conduct an ablation study by replacing our default flow estimator with the off-the-shelf SEA-RAFT [36] for bidirectional flow supervision. Quantitative results are reported in Table 6.

C.10. Memory and Training Time Footprint

Although only a single rendering pass is required during inference, the triple rendering and flow supervision in our

Figure 9. Qualitative comparison on Neural3DV: Scene with fire (rapid opacity changes) and non-stage-capture setting.

Table 6. Ablation study on flow supervision: comparison of WAFT [35] and SEA-RAFT [36] as optical flow supervision modules.

Method	PSNR↑	SSIM↑	LPIPS↓
WAFT [35]	30.08	0.904	0.0225
SEA-RAFT [36]	29.73	0.898	0.0253

method increase training overhead. We report the average training efficiency and peak GPU memory usage on the DNA-Rendering Dataset [4]. Because our MCMC strategy maintains a fixed total primitive count, we compare our approach against STGS [18] under a shared budget of 1M primitives. We chose this baseline because both methods allow primitives to dynamically appear and disappear over time, unlike deformation-based methods where all primitives persist across all frames (which typically requires fewer total primitives).

Table 7. Comparison of training efficiency and peak GPU memory under a shared budget of 1M primitives.

Method	**Training Time (s)**	**Peak Memory (GB)**
STGS [18]	1406.8	2.47
Ours	3794.3	3.14