GaussianVideo: Efficient Video Representation via Hierarchical Gaussian Splatting

Supplementary Material

1. Applications

Frame Interpolation. One advantange of our Gaussian video representation is the continuous nature. Downstream applications are not our focus, but we designed our method to ensure the learned Gaussian dynamics are semantic and coherent (see Fig. 2 in the main paper). To illustrate this, we demonstrate frame interpolation. Leveraging continuous motion representation for the Gaussians allows us to interpolate frames at arbitrary timesteps, including those not seen during training. Fig. 2 illustrates this process, showing the input left and right frames along with four interpolated frames. These interpolated frames successfully capture the realistic non-rigid motion of the cow, demonstrating the model's ability to maintain natural motion dynamics.

Video Stylization. For video stylization, we adopt the approach from [5]. Specifically, we use an existing image editing model to edit the first frame, then propagate these changes across the entire video with a reconstruction loss, updating only the spherical harmonics coefficients. While there exist some works that perform stylization using Gaussian splatting, we focus on a straightforward approach that does not require extensive extra supervision, such as CLIP [4] losses. Fig. 3 shows an example where a fire image is first used to edit the initial frame, which is then propagated throughout the video in a consistent manner, such as the chair adopting a fire-like texture and the floor to the left of the table turning a vivid red.

Spatial Resampling. The inherently spatial nature of our Gaussian video representation allows for making arbitrary adjustments in the spatial resolution of the learned videos. This is achieved by modifying the scale parameters along with the principal point and focal lengths of the learned camera parameters, using simple scaling operations. In Fig. 4, we demonstrate such a spatial resampling procedure where the frame width is divided by $1.5\times$ and the height is multiplied by $1.5\times$. The GaussianVideo representation effectively maintains high-quality reconstruction, free from artifacts, despite the significant spatial adjustments.

2. Further Implementation Details

We observed that incorporating B-splines provides significant flexibility but requires careful training. To properly learn camera motion, it has to be prioritized in the initial stages of training by using a higher learning rate (0.001) for the camera parameters compared to the mean B-splines (0.0001). This approach leads to properly learning camera motion early in training, while the motion of the Gaussians can be gradually refined throughout training. All other parameters are trained with a learning rate of 0.01. For optimization, we employ the Adan [7] optimizer, following the approach in [8].

For the baseline comparisons presented in the main paper, we use the recommended hyperparameters and architectures as specified in the corresponding literature and code releases.

3. Parametrization of Color and Opacity

In the main paper, we argued for using a lower degrees-of-freedom (DOF) design for color and opacity to encourage the semantic motion of the Gaussians. Here, we provide further details.

Li et al. [3] introduced the idea of modeling time-dependent opacities using radial basis functions (RBF) and time-dependent colors using Multilayer Perceptrons (MLP). However, our experiments revealed that these designs can be overpowering and discourage semantic motions for the learned Gaussians. Specifically, RBF-based opacity modeling often leads to simpler, non-semantic trajectories, where multiple Gaussians tend to work together to assemble the full motion rather than moving independently. Similarly, using MLPs for color representation can result in chaotic trajectories for the Gaussians, where color changes simulate motion instead of the Gaussians themselves following the trajectory.

Consider the example from Figure 4(b) in the main paper. The semantic motion corresponds to moving the ball along the U shape. However, if a reasonable MLP is used for the color, then the video could be learned without any Gaussians moving, by just changing the colors along the path. Similarly, with enough Gaussians, you could replicate the same effect by having a circle of Gaussians at each position of the ball, which pops into and out of existence, if you allow for modeling the opacity with an RBF.

4. Covariance Polynomial Orders

While the positions of the Gaussians often follow complex paths that necessitate the use of B-splines for effective modeling, the covariance of the Gaussians changes less frequently. Considering the standard decomposition $\mathbf{\Sigma} = (\mathbf{RS})(\mathbf{RS})^T$ where $\mathbf{R}$ is a rotation matrix and $\mathbf{S}$ is a scaling matrix, it is crucial to limit the flexibility of the scaling matrix. If the scaling matrix is represented with a

Figure 1. **Frame interpolation results for GaussianVideo using continuous motion representation.**

Figure 2. **Frame interpolation results using a basic optical flow method (using the RAFT [6] optical flow model). Note that the optical-flow based interpolation is much blurrier than the interpolation performed by our GaussianVideo model.**

Figure 3. **Video stylization results using GaussianVideo.** Starting with a *'fire'* style applied to the first frame, the style is propagated across the video by updating only the SH coefficients, ensuring consistent stylization without additional supervision. The visual coherence across frames highlights GaussianVideo's ability to maintain high-quality style transfer while preserving structural details throughout the sequence.

Figure 4. **Spatial resampling results on a learned video.** In this example, the frame height is scaled by 1.5×, and the width is reduced by 1.5×, demonstrating the model's ability to adjust resolution while preserving sharpness and detail. This is achieved by modifying scale parameters, as well as the principal point and focal lengths of the learned camera parameters, allowing for viewport adjustments beyond traditional resampling methods.

high-order polynomial or a B-spline, it can cause scales to fluctuate between 0 and large values unnecessarily, which is almost never necessary when following the semantic motions of objects in a video.

Degree	PSNR ↑	SSIM ↑	LPIPS ↓
Cov Degree 1	37.25	**0.96**	0.022
Cov Degree 3	37.34	**0.96**	**0.018**
Cov Degree 5 (Ours)	**37.38**	**0.96**	0.021
Cov Degree 15	37.11	**0.96**	0.024
Cov B-Spline 10 Knots	37.21	**0.96**	0.021

Table 1. Performance comparison of our approach on the DAVIS dataset using different parameterizations for the scaling and rotation matrices, with all other parameters fixed. Results demonstrate that a degree-5 polynomial achieves the best overall performance, balancing efficiency, and reconstruction quality, while higher degrees or B-splines introduce computational overhead without significant improvements.

In Table 1, we analyze the effect of using different polynomial degrees to parameterize the covariance matrix on the performance of our approach. Additionally, while B-splines provide flexibility, they are computationally more expensive than polynomials, even with our custom CUDA kernel implementation. All results reported in the paper use a degree-5 polynomial for the covariance. Our results indicate that performance remains consistent across different polynomial degrees, as long as they are allowed to vary over time. Notably, degree-5 polynomials yield the best balance of efficiency and performance. Although a B-spline with 10 knots performs reasonably well, it does not surpass most polynomial configurations.

5. Camera Modeling Details

An intrinsic camera matrix consists of focal lengths in the x and y direction, as well as the coordiantes of a principal

point. This is represented by a matrix

$$\mathbf{K} = \begin{bmatrix} \alpha_x & \gamma & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

where α_x, α_y are the focal lengths, and (u_0, v_0) is the principal point. γ is an optional skew parameter that is sometimes used, but in our case, we always assume $\gamma = 0$.

Meanwhile, the extrinsic camera matrix represents a 3D rotation and a 3D position in space for the camera. The rotation is represented as a quaternion, while the position is just represented as a vector. We can represent this by another matrix

$$\mathbf{L} = \begin{bmatrix} \mathbf{W}_{3\times3} & \mathbf{T}_{3\times1} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix}_{4\times4} . \tag{2}$$

Here, $\mathbf{W}$ is the 3×3 rotation matrix corresponding to the quaternion, and $\mathbf{T}$ is the position vector. Given the extrinsic camera matrix, we can calculate the position of the camera in world coordinates with the formula

$$\mathbf{C} = \mathbf{W}^{-1}\mathbf{T} . \tag{3}$$

For the intrinsic camera matrix, we first initialize the focal lengths of the camera as 60% the height and width of the video, respectively, and the principal point as the center of the screen. For the extrinsic parameters, we initialize the position to the origin and initialize to the quaternion representing zero rotation. The position and quaternion are then used as the initial condition for the Neural ODE [1]. Specifically, if $\mathbf{q}$ is the quaternion representing the camera rotation, then we let $\mathbf{z}_0 = [\mathbf{q} : \mathbf{T}^T]$ (concatenation, where we consider $\mathbf{T}$ as a row vector instead), and calculate

$$\mathbf{z}_{dT} = \mathbf{z}_0 + \int_0^t f_\theta(\mathbf{z}(s), s)ds \tag{4}$$

in order to solve for the quaternion and position vectors at time t.

We use a Neural ODE to model the extrinsic camera matrix, as its trajectory is expected to be smooth throughout space, due to the nature of how the video is recorded. The Neural ODE is implemented using an MLP with two hidden layers, ReLU activation, and a hidden dimension of 32. To solve the Neural ODE, we employ the Dopri5 [2] solver with atol and rtol set to 10^{-3}.

While we find that this is able to learn general motion patterns, due to the unsupervised nature of our learning and lack of additional supervisory signals, the learned motion representation is not perfect. Figure 8 in the main text shows a trajectory by the model vs the ground truth trajectory on a video from the DL3DV dataset. It can be seen that the general trajectory does match, even if there are some inconsistencies in the middle.

References

[1] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David Kristjanson Duvenaud. Neural ordinary differential equations. In *The Thirty-second Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2018. 3

[2] J. R. Dormand and P. J. Prince. A family of embedded runge-kutta formulae. *Journal of Computational and Applied Mathematics*, 6:19–26, 1980. 3

[3] Zhan Li, Zhang Chen, Zhong Li, and Yinghao Xu. Spacetime Gaussian feature splatting for real-time dynamic view synthesis. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8508–8520, 2023. 1

[4] 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*, 2021. 1

[5] Yang-Tian Sun, Yi-Hua Huang, Lin Ma, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. Splatter a video: Video gaussian representation for versatile processing. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2024. 1

[6] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European Conference on Computer Vision*, 2020. 2

[7] Xingyu Xie, Pan Zhou, Huan Li, Zhouchen Lin, and Shuicheng Yan. Adan: Adaptive Nesterov momentum algorithm for faster optimizing deep models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46:9508–9520, 2022. 1

[8] Xinjie Zhang, Xingtong Ge, Tongda Xu, Dailan He, Yan Wang, Hongwei Qin, Guo Lu, Jing Geng, and Jun Zhang. GaussianImage: 1000 FPS image representation and compression by 2D Gaussian splatting. In *European Conference on Computer Vision*, 2024. 1