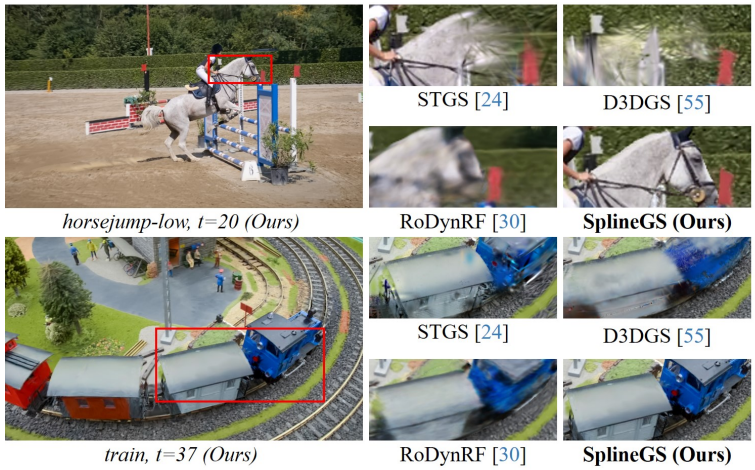
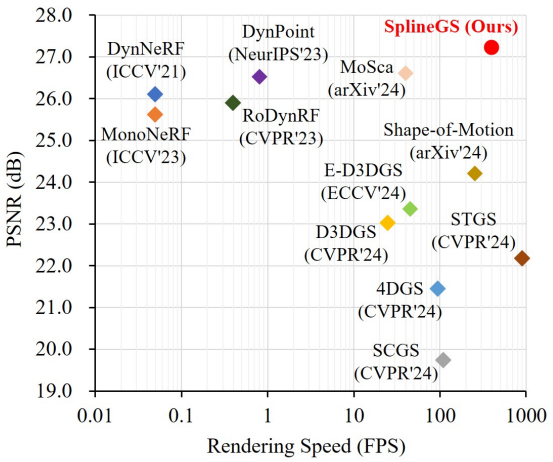
SplineGS: Robust Motion-Adaptive Spline for Real-Time Dynamic 3D Gaussians from Monocular Video

Jongmin Park[1*] Minh-Quan Viet Bui[1*] Juan Luis Gonzalez Bello[1] Jaeho Moon[1]
Jihyong Oh[2†] Munchurl Kim[1†]

[1]Korea Advanced Institute of Science and Technology (KAIST) [2]Chung-Ang University
{jm.park, bvmquan, juanluisgb, jaeho.moon, mkimee}@kaist.ac.kr jihyongoh@cau.ac.kr
https://kaist-viclab.github.io/splinegs-site/



(a) Visual comparison for novel view synthesis on the DAVIS dataset



(b) Performance gain on the NVIDIA dataset

Figure 1. **Our SplineGS achieves state-of-the-art rendering quality with fast rendering speed for novel spatio-temporal view synthesis from monocular videos without relying on pre-computed camera parameters.** (a) We use our predicted camera parameters for [24, 55] since COLMAP [42] is unable to provide reasonable camera parameters for most scenes in the DAVIS dataset [39]. (b) SplineGS achieves the best rendering quality while maintaining a rendering speed of 400 FPS on the NVIDIA dataset [56].

Abstract

*Synthesizing novel views from in-the-wild monocular videos is challenging due to scene dynamics and the lack of multi-view cues. To address this, we propose SplineGS, a COLMAP-free dynamic 3D Gaussian Splatting (3DGS) framework for high-quality reconstruction and fast rendering from monocular videos. At its core is a novel Motion-Adaptive Spline (MAS) method, which represents continuous dynamic 3D Gaussian trajectories using cubic Hermite splines with a small number of control points. For MAS, we introduce a Motion-Adaptive Control points Pruning (MACP) method to model the deformation of each dynamic 3D Gaussian across varying motions, progressively pruning control points while maintaining dynamic modeling integrity. Additionally, we present a joint optimization strategy for camera parameter estimation and 3D Gaussian attributes, leveraging photometric and geometric consistency. This eliminates the need for Structure-from-Motion preprocessing and enhances SplineGS's robustness in real-world conditions. Experiments show that SplineGS significantly outperforms state-of-the-art methods in novel view synthesis quality for dynamic scenes from monocular videos, achieving **thousands** times faster rendering speed.*

1. Introduction

Novel View Synthesis (NVS) is fundamental to 3D vision, supporting applications like virtual reality (VR), augmented reality (AR), and film production. NVS aims to generate

*Co-first authors (equal contribution).
†Co-corresponding authors.

images from any viewpoint in a scene, requiring accurate reconstruction from multiple 2D images. NeRF [34] has advanced the field of NVS by utilizing learned implicit functions to represent static scenes, though it requires considerable training and rendering time. Recently, 3D Gaussian Splatting (3DGS) [20] has revolutionized this process by replacing implicit volumetric rendering with differentiable rasterization of 3D Gaussians, enabling real-time rendering and providing a more explicit scene representation.

For NVS of dynamic scenes, prior works have extended the static scene representations [20, 34] by incorporating deformation models in canonical space using implicit representations [12, 30, 35, 36, 55], grid-based models that decompose the 4D space-time domain into multiple 2D planes [4, 6, 10, 43, 52], and polynomial trajectories [24]. However, these methods face several challenges: implicit representations significantly increase computational overhead and reduce rendering speed [6, 52]; grid-based models struggle to fully capture the dynamic nature of scene structures, hindering their ability to model fine details accurately [24]; and although polynomial trajectories improve efficiency, their fixed degree restricts flexibility for representing complex motions. To the best of our knowledge, none of the dynamic 3DGS-based methods provide experimental evidence to reliably support their novel spatio-temporal view synthesis capabilities for rendering unseen intermediate time indices. Additionally, most existing methods for dynamic NVS rely heavily on external camera parameter estimators like COLMAP [42], which often produce imprecise results for challenging in-the-wild monocular videos.

To address these issues for modeling scene dynamics, we exploit a spline-based model to dynamic 3D Gaussian trajectories, inspired by classic 3D-curve modeling in computer graphics [9]. Splines are widely used in geometric modeling for graphical applications, providing smooth and continuous representations of complex shapes with a minimal number of control points [2, 8]. This efficiency makes them ideal for modeling intricate geometric structures while maintaining both flexibility and precision. In this paper, we propose SplineGS, a framework for high-quality dynamic scene reconstruction and real-time neural rendering from *in-the-wild* monocular videos without relying on external camera estimators like COLMAP [42]. SplineGS introduces Motion-Adaptive Spline (MAS), based on cubic Hermite splines [2, 8], to effectively represent dynamic 3D Gaussian deformations. Our MAS consists of piecewise cubic functions defined by control points that dictate each segment's curvature and direction. Each control point is adjustable and optimized as a learnable parameter, enabling faster and more precise modeling of complex motion trajectories. Additionally, to adaptively model the trajectory of each dynamic 3D Gaussian based on motion complexity during training, we introduce a Motion-Adaptive Control points Pruning (MACP) method that adjusts the number of control points to improve rendering quality and efficiency. Furthermore, we incorporate a camera parameter estimation method jointly optimized with 3D Gaussian attributes and MAS, leveraging photometric and geometric consistency, thus eliminating the need for external estimators. Experiments show that SplineGS *significantly outperforms* state-of-the-art (SOTA) dynamic NVS methods both qualitatively and quantitatively, offering faster rendering speed, as shown in Fig. 1. Our contributions are as follows:

- We propose a novel ***Spline**-based* dynamic 3D **G**aussian **S**platting framework, SplineGS, which is (i) *COLMAP-free*, (ii) *very fast* and (iii) *of high quality* in reconstructing the dynamic scenes from in-the-wild monocular videos;
- A novel **M**otion-**A**daptive **S**pline (MAS) is introduced, which can accurately and effectively represent the continuous trajectory of each dynamic 3D Gaussian;
- A **M**otion-**A**daptive **C**ontrol points **P**runing (MACP) method is presented, which can efficiently adjust the number of control points for each spline function, optimizing both rendering quality and the efficiency of MAS.

2. Related Work

Dynamic NeRF. Recent advancements in video view synthesis have extended the static NeRF model [34] to represent scene dynamics. These include dynamic NeRFs using scene flow-based frameworks [12, 25, 26], deformation estimation in canonical fields [1, 3, 16, 30, 35, 36, 40, 44, 47, 51, 53], and 4D grid-based spatio-temporal radiance fields [4, 6, 10, 43]. Techniques such as NSFF [25], DynNeRF [12], and DynIBaR [26] combine time-independent and time-dependent radiance fields to synthesize novel spatio-temporal perspectives from monocular videos. Despite these advancements, current dynamic NeRFs still fall short of recent dynamic 3D Gaussian Splatting (3DGS) techniques in rendering quality and efficiency.

Dynamic 3DGS. The improvements in rendering quality and speed achieved by 3DGS [20] have inspired further studies [14, 24, 27, 31, 52, 55] to extend the static 3DGS framework to dynamic scenes by enabling the deformation of 3D Gaussian attributes. The pioneering work on dynamic 3DGS [31] introduces time-dependent offsets for the positions and rotations of dynamic 3D Gaussians via an MLP; however, this approach slows down the rendering. D3DGS [55] builds on this concept with an annealing smoothing training mechanism to improve temporal smoothness and rendering quality. 4DGS [52] replaces the MLP network with a grid-based structure to boost efficiency, though this change requires quality trade-offs due to the resolution limits of grid-based methods. In contrast, SC-GS [14] combines an MLP deformation network with sparse spatial deformation, reducing the computational

cost of MLP while maintaining quality. STGS [24] introduces polynomial trajectories for motion modeling, improving speed and quality over implicit representations. CasualFVS [22] warps dynamic content from neighboring frames, and MoSca [23] proposes a graph-based motion modeling approach to handle sparse control deformation. Very recently, a concurrent work [21] proposes modeling dynamic 3D Gaussian trajectories using polynomial interpolation between *fixed* keyframes for NVS from *multi-view* videos *with COLMAP [42] assistance*. Unlike [21], our SplineGS *adaptively* optimizes the deformation of dynamic 3D Gaussians, accounting for varying motion degrees and types in in-the-wild *monocular* videos, without requiring preprocessed camera parameters (*COLMAP-free* approach).

Neural Rendering without SfM Preprocessing. Accurate camera parameters, including extrinsics and intrinsics, are essential for neural rendering approaches to capture fine details [17, 28]. However, in real-world settings, camera parameters derived from Structure-from-Motion (SfM) algorithms such as COLMAP [42] often exhibit pixel-level inaccuracies [29, 41], compromising the structural details of rendered scenes [30]. To address this, several NeRF methods [28, 32, 37, 50] jointly optimize NeRF architectures and camera parameters. Recently, a local-to-global training approach [11] is introduced to optimize both camera parameters and 3D Gaussians. However, these methods are limited to static scenes. For dynamic scene reconstruction, RoDynRF [30] and MoSca [23] use motion masks to gather multi-view cues from static regions, allowing robust rendering without pre-computed camera parameters. Our SplineGS is also COLMAP-free, significantly outperforming RoDynRF [30] and MoSca [23] in rendering quality and efficiency, enabled by our novel spline-based architecture.

3. Preliminary: 3D Gaussian Splatting

3DGS [20] represents the radiance field of a scene using anisotropic 3D Gaussians, each of which is formulated as

$$G(\boldsymbol{x}) = e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})} \tag{1}$$

where $\boldsymbol{x} \in \mathbb{R}^3$ denotes a 3D position, and $\boldsymbol{\mu} \in \mathbb{R}^3$ and $\boldsymbol{\Sigma} \in \mathbb{R}^{3\times3}$ represent the mean (center) and the covariance matrix of the 3D Gaussian, respectively. To ensure that $\boldsymbol{\Sigma}$ is positive semi-definite and contains physical meaning, it is decomposed into a diagonal scaling matrix $\boldsymbol{S} \in \mathbb{R}^{3\times3}$ of a scale vector $\boldsymbol{s} \in \mathbb{R}^3$ and a rotation matrix $\boldsymbol{R} \in \mathbb{R}^{3\times3}$ as $\boldsymbol{\Sigma} = \boldsymbol{RSS}^\top\boldsymbol{R}^\top$, where $\boldsymbol{R}$ is parameterized by a learnable unit-length quaternion $\boldsymbol{q} \in \mathbb{R}^4$. In addition, each 3D Gaussian is parameterized by an opacity $\sigma \in \mathbb{R}$ and a color $\boldsymbol{c} \in \mathbb{R}^3$. To render the color of each pixel, the color and the opacity of each 3D Gaussian are computed using Eq. 1, and the rendered color $\boldsymbol{C}$ is computed by the alpha-blending of the $\mathcal{N}$ ordered 3D Gaussians overlapping the pixel as

$$\boldsymbol{C} = \textstyle\sum_{i\in\mathcal{N}} \boldsymbol{c}_i\alpha_i \prod_{j=1}^{i-1}(1-\alpha_j), \tag{2}$$

where $\boldsymbol{c}_i$ is the color of the i^{th} 3D Gaussian and α_i is a density of the i^{th} 3D Gaussian which is given by evaluating the 2D covariance $\boldsymbol{\Sigma}' \in \mathbb{R}^{2\times2}$. Here, $\boldsymbol{\Sigma}'$ is formulated as $\boldsymbol{\Sigma}' = \boldsymbol{JW\Sigma W}^\top\boldsymbol{J}^\top$, where $\boldsymbol{J}$ is the Jacobian of the affine approximation of the projective transformation and $\boldsymbol{W}$ is a viewing transformation matrix. Similar to prior works [23, 27], we extend 3DGS [20] to a union of static 3D Gaussians $\{G_i^{\text{st}} | i = 1, 2, ..., n^{\text{st}}\}$ and dynamic 3D Gaussians $\{G_i^{\text{dy}} | i = 1, 2, ..., n^{\text{dy}}\}$ to represent static backgrounds and moving objects, respectively, in dynamic scenes.

4. Proposed Method: SplineGS

4.1. Overview of SplineGS

We describe the overall architecture of SplineGS in Fig. 2. Given a monocular video $\{\boldsymbol{I}_t | t = 0, 1, ..., N_f - 1\}$ where N_f is the total number of frames, SplineGS is designed to synthesize high-quality novel spatio-temporal views with fast rendering speed, and to estimate the camera parameters, including the extrinsics $[\hat{\boldsymbol{R}}_t|\hat{\boldsymbol{T}}_t] \in \mathbb{R}^{3\times4}$ for each time t, and the shared intrinsic $\hat{\boldsymbol{K}} \in \mathbb{R}^{3\times3}$. As shown in Fig. 2, to stabilize joint optimization of the 3D Gaussian attributes and camera parameter estimation, we adopt a two-stage optimization process consisting of a warm-up stage and a main training stage for our SplineGS architecture. In the warm-up stage, we coarsely optimize the camera parameters $[\hat{\boldsymbol{R}}_t|\hat{\boldsymbol{T}}_t]$ and $\hat{\boldsymbol{K}}$ using photometric and geometric consistency. In the main training stage, we initialize the 3D Gaussians based on the estimated camera poses and jointly optimize the 3D Gaussian attributes with the camera parameters. Specifically, for each dynamic 3D Gaussian, we propose a novel spline-based deformation modeling method, called Motion-Adaptive Spline (MAS), which utilizes a cubic Hermite spline function [2, 8] to accurately model the continuous trajectory with a small number of control points. For MAS, we introduce a Motion-Adaptive Control points Pruning (MACP) method that effectively adjusts the number of control points for each dynamic 3D Gaussian, considering the object's motion types and degrees.

In the following sections, we detail the process of our MAS method in Sec. 4.2, followed by the camera parameter estimation process in Sec. 4.3. Finally, we describe the overall optimization process in Sec. 4.4.

4.2. Motion-Adaptive Spline for 3D Gaussians

To represent the continuous trajectory of *each* dynamic 3D Gaussian for moving objects over time, we propose MAS which modifies the mean parameter to a set of learnable control points. For this, we use the cubic Hermite spline function [2, 8] to model the mean of each *dynamic* 3D Gaussian at time t as

$$\boldsymbol{\mu}(t) = S(t, \mathbf{P}), \tag{3}$$

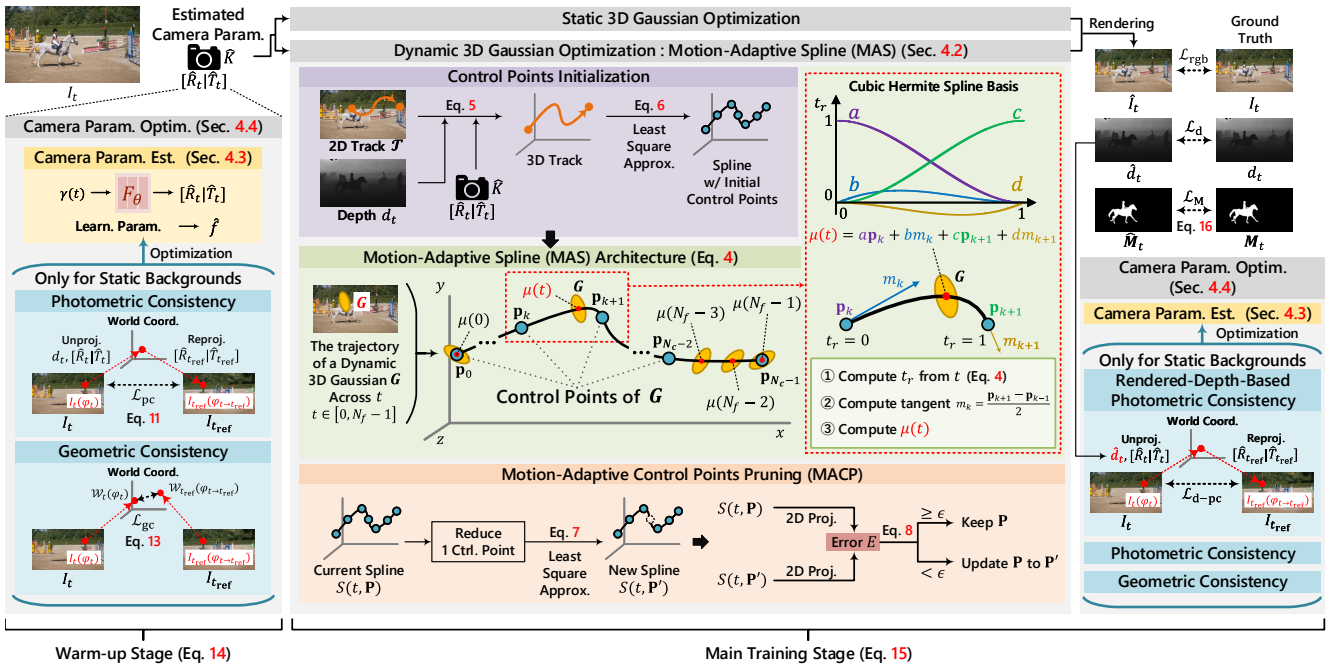


Figure 2. **Overview of SplineGS.** Our SplineGS leverages spline-based functions to model the deformation of dynamic 3D Gaussians with a novel Motion-Adaptive Spline (MAS) architecture. It is composed of sets of learnable control points based on a cubic Hermite spline function [2, 8] to accurately model the trajectory of each dynamic 3D Gaussian and to achieve faster rendering speed. To avoid any preprocessing of camera parameters, i.e., COLMAP-free, we adopt a two-stage optimization: warm-up and main training stages.

where $\mathbf{P} = \{\mathbf{p}_k | \mathbf{p}_k \in \mathbb{R}^3, k \in 0, \ldots, N_c - 1\}$ is a set of N_c learnable control points, serving as an additional attribute for each dynamic 3D Gaussian, and $S(t, \mathbf{P})$ is defined as

$$\begin{aligned} S(t, \mathbf{P}) = (2t_r^3 - 3t_r^2 + 1)\mathbf{p}_{\lfloor t_s \rfloor} + (t_r^3 - 2t_r^2 + t_r)\mathbf{m}_{\lfloor t_s \rfloor} \\ +(-2t_r^3 + 3t_r^2)\mathbf{p}_{\lfloor t_s \rfloor + 1} + (t_r^3 - t_r^2)\mathbf{m}_{\lfloor t_s \rfloor + 1}, \quad (4) \end{aligned}$$

$$t_r = t_s - \lfloor t_s \rfloor, \quad t_s = t_n(N_c - 1), \quad t_n = t/(N_f - 1),$$

where $\mathbf{m}_k$ is the approximated tangent of the control point $\mathbf{p}_k$, formulated as $\mathbf{m}_k = (\mathbf{p}_{k+1} - \mathbf{p}_{k-1})/2$. Note that $S(t, \mathbf{P})$ indicates the piecewise cubic function to represent the whole continuous trajectory of each dynamic 3D Gaussian, and N_c can be different from N_f. The optimal N_c can be estimated by Motion-Adaptive Control points Pruning (MACP) in the following section.

Control Points Initialization. To stably optimize $S(t, \mathbf{P})$, it is essential to initialize the control points with appropriate geometric considerations that ensure temporal consistency. For this, we leverage the long-range 2D tracking [19] and the metric depth [38] priors. Let $\boldsymbol{\mathcal{T}} = \{\boldsymbol{\varphi}_t^{\text{tr}} | \boldsymbol{\varphi}_t^{\text{tr}} \in \mathbb{R}^2,\ t = 0, \ldots, N_f - 1\}$ be a 2D track, $\pi_{\boldsymbol{K}}(\cdot)$ be a projection function from the camera space to the image space with the camera intrinsic $\boldsymbol{K}$, and $\boldsymbol{\varphi}_t^{\text{tr}}$ be a pixel coordinate corresponding to the 2D track $\boldsymbol{\mathcal{T}}$ at time t. We unproject each 2D track $\boldsymbol{\mathcal{T}}$ to the world space to compute the 3D track aided by the frame's metric depth $\boldsymbol{d}_t$ and camera extrinsic $[\hat{\boldsymbol{R}}_t | \hat{\boldsymbol{T}}_t]$ at time t. The unprojection function $\mathcal{W}_t(\cdot)$ from the image space to the world space is formulated as

$$\mathcal{W}_t(\boldsymbol{\varphi}_t^{\text{tr}}) = \hat{\boldsymbol{R}}_t^\top \pi_{\hat{\boldsymbol{K}}}^{-1}\big(\boldsymbol{\varphi}_t^{\text{tr}}, \boldsymbol{d}_t(\boldsymbol{\varphi}_t^{\text{tr}})\big) - \hat{\boldsymbol{R}}_t^\top \hat{\boldsymbol{T}}_t. \quad (5)$$

It should be noted that we estimate $[\hat{\boldsymbol{R}}_t | \hat{\boldsymbol{T}}_t]$ and $\hat{\boldsymbol{K}}$ from a sequence of frames only, without using any given ground truth values, as mentioned in Sec. 4.3. Then, we initialize the per-Gaussian control points set $\mathbf{P}$, by a least-square (LS) approximation, such that the spline-described curve fits the initial curve by the tracker, given by

$$\min_{\mathbf{P}} \textstyle\sum_{t=0}^{N_f - 1} \big\| \mathcal{W}_t(\boldsymbol{\varphi}_t^{\text{tr}}) - S(t, \mathbf{P}) \big\|_2^2. \quad (6)$$

Motion-Adaptive Control Points Pruning (MACP). An excessive number of control points may cause over-fitting and decrease the processing speed of our MAS module, leading to poorer rendering qualities with reduced rendering speed (see Table 3). Furthermore, as each scene exhibits various types and degrees of motion for moving objects, the number of control points required for each dynamic 3D Gaussian trajectory should be adaptively adjusted to accommodate the scene dynamics. To achieve this, we propose the MACP method for MAS, which can generate sparser control points while ensuring no dynamic modeling degradation. MACP is designed on top of 3D Gaussian densification [20], but focuses on optimizing the number of control points in MAS. Our MACP computes a new spline function $S(t, \mathbf{P}')$ after every 3D Gaussian densification step, where $\mathbf{P}' = \{\mathbf{p}'_l | \mathbf{p}'_l \in \mathbb{R}^3,\ l = 0, \ldots, N_c - 2\}$ is a set of $N_c - 1$ control points, which contains one-fewer control points than the current set $\mathbf{P}$ of control points. We compute the LS approximation to find the reduced set $\mathbf{P}'$ of control points as

$$\min_{\mathbf{P}'} \sum_{t=0}^{N_f-1} \left\| S(t,\mathbf{P}) - S(t,\mathbf{P}') \right\|_2^2. \tag{7}$$

Then, the updated optimal set $\mathbf{P}$ of control points is assigned with the values of $\mathbf{P}'$ if the error E between $S(t,\mathbf{P})$ and $S(t,\mathbf{P}')$ is smaller than a threshold ϵ, as given by

$$\mathbf{P} = \begin{cases} \mathbf{P}', & \text{if } E < \epsilon \\ \mathbf{P}, & \text{otherwise} \end{cases}, \text{ where}$$
$$E = \frac{1}{N_f} \sum_{t=0}^{N_f-1} \|\pi_{\hat{\boldsymbol{K}}}(\hat{\boldsymbol{R}}_t S(t,\mathbf{P}) + \hat{\boldsymbol{T}}_t) - \pi_{\hat{\boldsymbol{K}}}(\hat{\boldsymbol{R}}_t S(t,\mathbf{P}') + \hat{\boldsymbol{T}}_t)\|_2^2. \tag{8}$$

By following MACP, each dynamic 3D Gaussian is allowed to have a different number of control points. Therefore, the MACP can guide our MAS to have a minimal number of control points when modeling simple motion, while having more control points for more complex motions (see Fig. 8).

4.3. Camera Parameter Estimation

The traditional SfM methods, such as COLMAP [42], fail to reliably estimate camera parameters in dynamic scenes from in-the-wild monocular videos [30]. For this reason, we propose to estimate camera parameters for joint optimization with the 3D Gaussian attributes. For each frame at time t, we predict a rotation matrix $\hat{\boldsymbol{R}}_t \in \mathbb{R}^{3\times3}$ and a translation vector $\hat{\boldsymbol{T}}_t \in \mathbb{R}^3$, representing the extrinsic parameters of a monocular camera, using a shallow MLP F_θ as

$$[\hat{\boldsymbol{R}}_t|\hat{\boldsymbol{T}}_t] = F_\theta(\gamma(t)), \tag{9}$$

where $\gamma(\cdot)$ is a positional encoding [34]. We also predict the focal length $\hat{f}$ of the camera intrinsic $\hat{\boldsymbol{K}}$ as a learnable parameter that is shared across all frames in the monocular video. To accurately optimize the camera parameters, we enforce two types of consistency—photometric and geometric—for the static background between random reference cameras $[\hat{\boldsymbol{R}}_{t_{\text{ref}}}|\hat{\boldsymbol{T}}_{t_{\text{ref}}}]$ and the target camera $[\hat{\boldsymbol{R}}_t|\hat{\boldsymbol{T}}_t]$, encouraging alignment both visually and structurally.

Photometric Consistency. Given the pre-computed metric depth [38], the camera intrinsics and extrinsics under optimization will converge as long as the projected reference frame's color $\boldsymbol{I}_{t_{\text{ref}}}(\boldsymbol{\varphi}_{t\to t_{\text{ref}}})$ is aligned to the target frame's color $\boldsymbol{I}_t(\boldsymbol{\varphi}_t)$. $\boldsymbol{\varphi}_{t\to t_{\text{ref}}}$ is the reference pixel coordinate corresponding to the target frame's pixel coordinate $\boldsymbol{\varphi}_t$ as

$$\boldsymbol{\varphi}_{t\to t_{\text{ref}}} = \pi_{\hat{\boldsymbol{K}}}(\hat{\boldsymbol{R}}_{t_{\text{ref}}}(\hat{\boldsymbol{R}}_t^\top \pi_{\hat{\boldsymbol{K}}}^{-1}(\boldsymbol{\varphi}_t, \boldsymbol{d}_t(\boldsymbol{\varphi}_t)) - \hat{\boldsymbol{R}}_t^\top \hat{\boldsymbol{T}}_t) + \hat{\boldsymbol{T}}_{t_{\text{ref}}}). \tag{10}$$

We refer to such projection alignment as photometric consistency, which is encouraged by the loss $\mathcal{L}_{\text{pc}}$ given by

$$\mathcal{L}_{\text{pc}} = \sum_{\boldsymbol{\varphi}_t} \left\| \boldsymbol{M}_{t,t_{\text{ref}}}(\boldsymbol{\varphi}_t) \odot (\boldsymbol{I}_t(\boldsymbol{\varphi}_t) - \boldsymbol{I}_{t_{\text{ref}}}(\boldsymbol{\varphi}_{t\to t_{\text{ref}}})) \right\|_2^2, \tag{11}$$

where $\odot$ is the Hadamard product [13], and $\boldsymbol{M}_{t,t_{\text{ref}}}$ is a union motion mask that excludes dynamic objects in $\boldsymbol{I}_t$ and $\boldsymbol{I}_{t_{\text{ref}}}$ for removing the inconsistencies due to dynamic regions, which is given by

$$\boldsymbol{M}_{t,t_{\text{ref}}}(\boldsymbol{\varphi}_t) = \boldsymbol{M}_t(\boldsymbol{\varphi}_t)\boldsymbol{M}_{t_{\text{ref}}}(\boldsymbol{\varphi}_{t\to t_{\text{ref}}}), \tag{12}$$

where $\boldsymbol{M}_t$ and $\boldsymbol{M}_{t_{\text{ref}}}$ are pre-computed motion masks [54] of $\boldsymbol{I}_t$ and $\boldsymbol{I}_{t_{\text{ref}}}$, respectively. Note that motion mask's value is 0 for static regions, and 1 otherwise.

Geometric Consistency. Along with the photometric consistency, we compute the geometric consistency of unprojected pixels in 3D space to make our optimization more stable. The geometric consistency loss is formulated as

$$\mathcal{L}_{\text{gc}} = \sum_{\boldsymbol{\varphi}_t} \left\| \boldsymbol{M}_{t,t_{\text{ref}}}(\boldsymbol{\varphi}_t) \odot (\mathcal{W}_t(\boldsymbol{\varphi}_t) - \mathcal{W}_{t_{\text{ref}}}(\boldsymbol{\varphi}_{t\to t_{\text{ref}}})) \right\|_2^2, \tag{13}$$

where $\mathcal{W}_t(\boldsymbol{\varphi}_t)$ and $\mathcal{W}_{t_{\text{ref}}}(\boldsymbol{\varphi}_{t\to t_{\text{ref}}})$ are the corresponding 3D locations of $\boldsymbol{\varphi}_t$ and $\boldsymbol{\varphi}_{t\to t_{\text{ref}}}$, respectively (see Eq. 5).

4.4. Optimization

To stabilize joint training of the MAS and the camera parameter estimation, we adopt a two-stage optimization process consisting of the warm-up stage and the main training stage. In the warm-up stage, we optimize only the camera parameters using the photometric and geometric consistency. The total loss in the warm-up stage is given by

$$\mathcal{L}_{\text{total}}^{\text{warm}} = \lambda_{\text{pc}}\mathcal{L}_{\text{pc}} + \lambda_{\text{gc}}\mathcal{L}_{\text{gc}}. \tag{14}$$

After the warm-up stage, we obtain the coarsely predicted camera intrinsic $\hat{\boldsymbol{K}}$ and the set of extrinsics $[\hat{\boldsymbol{R}}|\hat{\boldsymbol{T}}]$ for all frames, which are then used to initialize the set $\mathbf{P}$ of control points for each dynamic 3D Gaussian, as described in Sec. 4.2. In the main training stage, we jointly optimize the static and dynamic 3D Gaussians along with the camera parameter estimation based on the total loss function as

$$\begin{aligned} \mathcal{L}_{\text{total}}^{\text{main}} = {} & \lambda_{\text{rgb}}\mathcal{L}_{\text{rgb}} + \lambda_{\text{d}}\mathcal{L}_{\text{d}} + \lambda_{\text{M}}\mathcal{L}_{\text{M}} \\ & + \lambda_{\text{pc}}\mathcal{L}_{\text{pc}} + \lambda_{\text{d-pc}}\mathcal{L}_{\text{d-pc}} + \lambda_{\text{gc}}\mathcal{L}_{\text{gc}}, \end{aligned} \tag{15}$$

where $\mathcal{L}_{\text{rgb}}$ and $\mathcal{L}_{\text{d}}$ are the L1 losses between the rendered frame and GT frame, and between the rendered depth and GT depth, respectively. In the main training stage, we compute an additional photometric consistency loss $\mathcal{L}_{\text{d-pc}}$ that utilizes the rendered depth $\hat{\boldsymbol{d}}$ of the 3D Gaussians instead of the metric depth [38] prior $\boldsymbol{d}$ as $\hat{\boldsymbol{d}}$ allows the estimated 3D Gaussian geometry to guide the joint optimization of the camera parameters and 3D Gaussian attributes.

In addition to the camera parameter estimation and imagery reconstruction losses, we adopt a binary dice loss [45] $\mathcal{L}_{\text{M}}$ between the pre-computed motion mask $\boldsymbol{M}_t$ [54] and the rendered motion mask $\hat{\boldsymbol{M}}_t$ that can be derived from the dynamic 3D Gaussians. The binary dice loss initially proposed in [45] for highly imbalanced segmentation of medical imagery helps encouraging better separation between our dynamic and static 3D Gaussians as described by

$$\mathcal{L}_{\text{M}} = 1 - \frac{2\sum_{\boldsymbol{\varphi}_t} \boldsymbol{M}_t(\boldsymbol{\varphi}_t)\hat{\boldsymbol{M}}_t(\boldsymbol{\varphi}_t) + \varepsilon}{\sum_{\boldsymbol{\varphi}_t} \left(\boldsymbol{M}_t(\boldsymbol{\varphi}_t) + \hat{\boldsymbol{M}}_t(\boldsymbol{\varphi}_t)\right) + \varepsilon}, \tag{16}$$

where ε is a smooth term to avoid numerical issues. $\hat{\boldsymbol{M}}_t(\boldsymbol{\varphi}_t)$ is computed by the alpha-blending of the 3D Gaussians overlapping $\boldsymbol{\varphi}_t$ (similar to Eq. 2) as

	Methods	Jumping	Skating	Truck	Umbrella	Balloon1	Balloon2	Playground	Average	FPS↑	Tr. (h)↓
SfM preprocessing	DynNeRF (ICCV'21) [12]	24.68 / 0.090	32.66 / 0.035	28.56 / 0.082	23.26 / 0.137	22.36 / 0.104	27.06 / 0.049	24.15 / 0.080	26.10 / 0.082	0.05	>24
	MonoNeRF (ICCV'23) [46]	24.26 / 0.091	32.06 / 0.044	27.56 / 0.115	23.62 / 0.180	21.89 / 0.129	27.36 / 0.052	22.61 / 0.130	25.62 / 0.106	0.05	N/A
	STGS (CVPR'24) [24]	20.82 / 0.187	24.80 / 0.109	25.01 / 0.103	21.88 / 0.195	20.36 / 0.196	23.12 / 0.124	19.23 / 0.151	22.17 / 0.152	**900**	**0.2**
	SCGS (CVPR'24) [14]	15.68 / 0.920	14.88 / 0.908	23.81 / 0.140	21.84 / 0.160	20.17 / 0.179	21.07 / 0.149	20.71 / 0.115	19.74 / 0.367	110	1.1
	D3DGS (CVPR'24) [55]	22.02 / 0.266	24.06 / 0.227	23.04 / 0.247	22.67 / 0.192	21.22 / 0.202	25.86 / 0.118	22.30 / 0.111	23.02 / 0.195	25	1.0
	4DGS (CVPR'24) [52]	22.37 / 0.178	26.72 / 0.084	25.93 / 0.097	22.36 / 0.178	21.89 / 0.153	24.85 / 0.081	21.36 / 0.089	23.64 / 0.123	95	0.25
	SP-GS (ICML'24) [48]	22.13 / 0.468	29.21 / 0.236	27.38 / 0.190	24.88 / 0.323	24.36 / 0.180	**29.65** / 0.097	22.29 / 0.234	25.70 / 0.247	180	6.0
	Casual-FVS (ECCV'24) [22]	23.45 / 0.100	29.98 / 0.045	25.22 / 0.090	23.24 / 0.096	23.76 / 0.079	24.15 / 0.081	22.19 / 0.074	24.57 / 0.081	48	0.25
	Kinematic Fields (ECCV'24) [15]	23.74 / 0.102	31.89 / 0.035	28.34 / 0.074	25.46 / 0.098	23.72 / 0.079	26.49 / 0.055	24.64 / 0.053	26.33 / 0.071	0.3	7.0
	E-D3DGS (ECCV'24) [5]	22.39 / 0.180	24.62 / 0.137	26.85 / 0.081	23.60 / 0.145	20.87 / 0.161	24.39 / 0.073	20.79 / 0.124	23.36 / 0.129	45	2.6
	CTNeRF (Pattern Recognition'24) [33]	24.35 / 0.094	33.51 / 0.034	28.27 / 0.084	23.48 / 0.129	22.19 / 0.111	26.86 / 0.048	24.28 / 0.077	26.13 / 0.082	0.1	>24
	D-NPC (Eurographics'25) [18]	24.51 / 0.116	30.22 / 0.061	28.92 / 0.084	24.15 / 0.136	22.26 / 0.161	25.84 / 0.107	23.59 / 0.099	25.64 / 0.109	70	0.27
	Shape-of-Motion (arXiv'24) [49]	22.74 / 0.128	29.12 / 0.048	24.63 / 0.126	23.79 / 0.089	22.74 / 0.120	25.83 / 0.072	20.61 / 0.130	24.21 / 0.102	255	1.1
	Ex4DGS (NeurIPS'24) [21]	18.93 / 0.321	21.92 / 0.233	19.04 / 0.308	19.03 / 0.340	14.69 / 0.503	16.29 / 0.457	14.16 / 0.437	17.72 / 0.371	84	0.65
	DynPoint (NeurIPS'23) [58]	24.69 / 0.097	31.34 / 0.045	**29.30** / 0.061	24.59 / 0.086	22.77 / 0.099	27.63 / 0.049	**25.37** / **0.039**	26.53 / 0.068	0.8	0.5
	RoDynRF (CVPR'23) [30] w/	**25.66** / 0.071	28.68 / 0.040	29.13 / 0.063	24.26 / 0.089	22.37 / 0.103	26.19 / 0.054	24.96 / 0.048	25.89 / 0.067	0.45	>24
	MoSca (arXiv'24) [23] w/	25.21 / 0.083	32.77 / 0.033	28.22 / 0.090	24.41 / 0.092	23.26 / 0.092	28.90 / 0.042	23.05 / 0.060	26.55 / 0.070	40	0.8
	SplineGS (Ours) w/	25.62 / **0.064**	**34.10** / 0.034	27.83 / **0.048**	24.60 / 0.076	22.67 / 0.120	29.04 / **0.030**	24.47 / 0.046	26.90 / 0.060	400	0.5
SfM-Free	RoDynRF (CVPR'23) [30]	24.27 / 0.100	28.71 / 0.046	28.85 / 0.066	23.25 / 0.104	21.81 / 0.122	25.58 / 0.064	25.20 / 0.052	25.38 / 0.079	0.45	>24
	MoSca (arXiv'24) [23]	25.43 / 0.080	32.62 / 0.033	28.29 / 0.086	24.40 / 0.091	23.27 / 0.091	29.01 / 0.042	23.23 / 0.058	26.61 / 0.069	40	1.0
	SplineGS (Ours)	25.50 / 0.068	33.72 / **0.031**	28.66 / 0.056	**25.61** / **0.071**	**24.43** / **0.068**	28.37 / 0.032	24.19 / 0.047	**27.21** / **0.053**	400	1.5

Table 1. **Novel view synthesis evaluation on the NVIDIA dataset (PSNR↑ / LPIPS↓).** **Red** and Blue denote the best and second-best performances, respectively. For Casual-FVS [22], we directly use the results from their paper, as the official code is unavailable. 'w/' denotes the use of camera poses obtained by COLMAP [42] for each SfM-free method. 'Tr.' denotes the training time. For MonoNeRF [46], the training time is omitted as it is not a per-scene optimized model. We use an RTX 3090 Ti GPU for each method.

D3DGS [55] STGS [24] RoDynRF [30] DynNeRF [12] SplineGS (Ours) Ground Truth

Figure 3. **Visual comparisons for novel view synthesis on the NVIDIA dataset.**

$$\hat{\boldsymbol{M}}_t(\boldsymbol{\varphi}_t) = \sum_{i\in\mathcal{N}} m_i \alpha_i \prod_{j=1}^{i-1}(1-\alpha_j), \qquad (17)$$

where $m_i = 0$ if the i^{th} 3D Gaussian is the static 3D Gaussian and $m_i = 1$ otherwise.

In conjunction, all terms in $\mathcal{L}_{\text{total}}^{\text{main}}$ guide our SplineGS to effectively and efficiently model dynamic 3D scenes from pure monocular videos, achieving more structural details and better temporal consistency than previous works [12, 14, 21–24, 30, 46, 52, 55], without relying on camera parameters obtained from external estimators [42].

5. Experiments

Implementation Details. We refer readers to the *Supplementary material* for the details of our implementation.

Datasets. We evaluate both the quantitative and qualitative performance of novel view and time synthesis on the widely used NVIDIA dataset [56] which features challenging monocular videos. Additionally, we assess novel view synthesis performance on in-the-wild monocular videos from the DAVIS dataset [39] which contains an average of 69.7 frames per video sequence.

5.1. Comparison with State-of-the-Art Methods

Novel View Synthesis. Table 1 presents a quantitative comparison of NVS between our SplineGS and existing SfM-based [5, 12, 14, 15, 18, 21, 22, 24, 33, 46, 48, 49, 52, 55, 58] and SfM-free [23, 30] methods on the NVIDIA dataset [56]. For this comparison, we follow the evaluation configuration in [30]. The results demonstrate that our SplineGS significantly outperforms SOTA methods in both the PSNR and LPIPS [57] metrics. Notably, SplineGS achieves **890**× and **8,000**× faster rendering speed compared to RoDynRF [30] and DynNeRF [12], respectively. Ex4DGS [21] and STGS [24], which are designed for multi-view settings, face challenges with inconsistent geometry alignment over time when trained on monocular videos. Fig. 3 shows the qualitative comparison between our SplineGS with the existing methods in [12, 24, 30, 55]. As highlighted by the red boxes, our method yields not only higher rendering quality but also dynamic objects more aligned and closer to the ground truth. In Fig. 4, we show our superior NVS results for in-the-wild monocular videos from the DAVIS dataset [39] compared to the existing methods in [24, 30, 55].

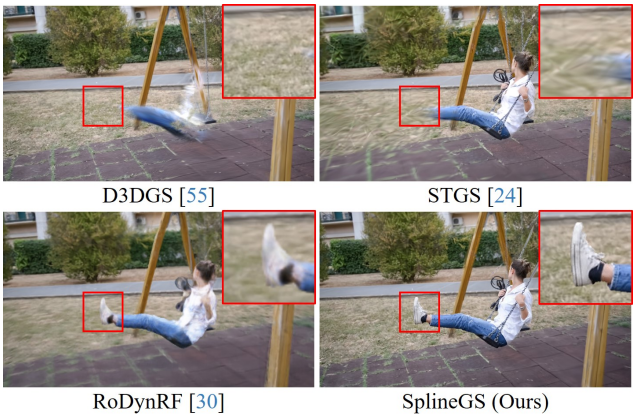


Figure 4. **Visual comparisons for novel view synthesis on the DAVIS dataset.**

Compared to [30] that is also COLMAP-free, SplineGS yields considerably more detailed novel views (red boxes) as shown in Fig. 4. For the other methods [24, 55], we observe that COLMAP [42] fails to recover camera parameters and initial point clouds on the DAVIS dataset [39], as also claimed in [30]. On the other hand, our COLMAP-free SplineGS reconstructs accurate camera parameters that are the ones actually used to train [24, 55], which are shown in Fig. 4 for comparison. More results are in the *Supplementary material*.

Novel View and Time Synthesis. To evaluate the capability of SplineGS to model continuous trajectories of moving objects in a scene, we compare the novel view and time synthesis results of SplineGS with those of NeRF-based [12, 30] and 3DGS-based [24, 52, 55] methods. For this evaluation, we follow the dataset sampling strategy in [25], which samples 24 timestamps from the NVIDIA dataset [56]. In addition, to simulate a larger motion, we exclude frames with odd time indices in the training sets. To ensure all test timestamps are not seen during training, and thus, to create a more *challenging* novel view and time synthesis validation, we exclude frames with even time indices in the test sets. Table 2 and Fig. 5 show the quantitative and qualitative comparisons for this challenging experiment. We observe that the NeRF-based methods, RoDynRF [30] and DynNeRF [12], generate inconsistent artifacts and blurriness for unseen times. Furthermore, the 3DGS-based methods [24, 52, 55] yield even more significant degradation when predicting unseen time indices. In contrast, SplineGS, built upon 3DGS [20] and equipped with our novel spline-based deformation, provides SOTA novel view rendering at unseen intermediate timestamps. Thanks to our MAS, SplineGS naturally and precisely captures the continuous trajectories of moving objects over time, enhancing the temporal consistency of rendered scenes that can be checked in tOF scores [7] of Table 2.

	Methods	PSNR↑	LPIPS↓	tOF↓
SfM preprocessing	DynNeRF (ICCV'21) [12]	23.36	0.219	0.921
	4DGS (CVPR'24) [52]	17.07	0.459	6.314
	D3DGS (CVPR'24) [55]	19.63	0.343	3.225
	STGS (CVPR'24) [24]	15.72	0.474	2.105
SfM-Free	RoDynRF (CVPR'23) [30]	21.58	0.221	2.138
	SplineGS (Ours)	**25.92**	**0.098**	**0.703**

Table 2. **Novel view and time synthesis evaluation on the NVIDIA dataset.**

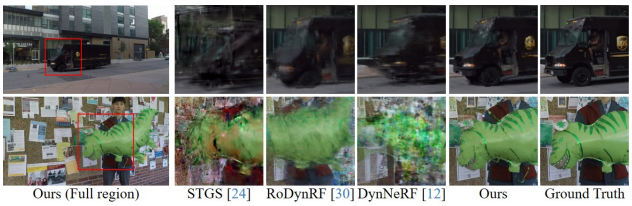


Figure 5. **Visual comparisons for novel view and time synthesis on the NVIDIA dataset.**

To further analyze the SplineGS's ability to model continuous trajectories of dynamic 3D Gaussians, we visualize the projected 2D motion tracking of dynamic objects in pixel space, comparing it with D3DGS [55] and STGS [24], as shown in Fig. 6. We observe that D3DGS [55] and STGS [24] cannot provide reliable motion tracking for moving objects, underscoring their limitations in modeling continuous trajectories of dynamic 3D Gaussians. In contrast, SplineGS provides accurate motion tracking, demonstrating the effectiveness of MAS for deforming dynamic 3D Gaussians. Further analysis is in the *Supplementary material*.

Figure 6. **Visual comparisons for motion tracking.** We visualize 2D pixel tracks to analyze motions of dynamic 3D Gaussians.

5.2. Ablation Study

Motion-Adaptive Spline (MAS). To demonstrate the effectiveness of MAS, we replace the MAS model with various deformation models, including an MLP, a grid-based model, polynomial functions of third degree (denoted as 'Poly (3^{rd})') and tenth degree (denoted as 'Poly (10^{th})'), and a Bézier curve [9]. For the MLP, grid-based model, and polynomial functions, we apply to them the structures similar to those in prior works, including D3DGS [55], 4DGS [52], and STGS [24], respectively. Additionally, we implement the Bézier curve [9], a commonly used method for curve modeling in computer graphics. Table 3-(a) presents quantitative comparisons of each 3D Gaussian trajectory model, focusing on rendering quality (PSNR, LPIPS) and deformation latency per Gaussian, denoted as g_{def}. This latency re-

Figure 7. **Visual comparisons for MACP ablation study.**

flects the computational time required to estimate the deformation of a single dynamic 3D Gaussian. As shown in Table 3-(a), our MAS model achieves superior rendering quality compared to all other deformation models. Consistent with analyses in previous works [24, 52, 55], we observe that the MLP and grid-based architectures require substantial computational costs for rendering. Among these methods, 'Poly (3^{rd})', as implemented in [24], demonstrates the best latency. However, fixed-degree polynomial functions have limited flexibility across varying motion complexities which adversely impacts rendering performance. To explore this further, we experiment with 'Poly (10^{th})' to assess changes in modeling capability. This adjustment, however, leads to noisier optimization and reduced efficiency, as variables under high exponents in 'Poly (10^{th})' lead to numerical instability. The Bézier curve [9] offers the second-best rendering quality, but its latency remains higher than our MAS due to its recursive nature of computation.

Motion-Adaptive Control Points Pruning (MACP). To assess the effectiveness of our MACP technique for MAS, we compare our full model with MACP against other versions of our model with two fixed numbers of control points $N_c = 4$ and $N_c = N_f$. As shown in Table 3-(c) and Fig. 7, our SplineGS with MACP achieves a good trade-off between rendering quality and g_{def} compared to the ablated models with fixed N_c. Using $N_c = 4$ for every dynamic 3D Gaussian limits the motion modeling capacity of MAS, resulting in significantly lower metrics and visible artifacts in the dynamic regions. Moreover, an excessive $N_c = N_f$ decreases the rendering speed of our MAS module and still falls short of the quality achieved by our full model with MACP, potentially due to motion overfitting. Fig. 8 shows the distribution of N_c values after optimization with MACP across scenes of varying motion complexities. In Fig. 8-(a), we visualize 'N_c Heatmap' that contains the pixel-wise averaged N_c values of the dynamic 3D Gaussians needed to render the 2D pixels (red higher, blue lower number of control points). As shown, the simpler motions in these scenes, such as those of the human bodies, can be modeled by the dynamic 3D Gaussians with smaller averages of N_c values, whereas the objects with complex and extensive motions (e.g., balloon) require higher averaged N_c values. Fig. 8-(b) presents the corresponding histogram of N_c values for the scenes' dynamic 3D Gaussians. For sequences with simple motion, such as 'Skating', the trajectories of most

(a) Motion-Adaptive Spline

	PSNR↑	LPIPS↓	g_{def} (ns)↓
MLP	23.51	0.125	149.41
Grid	25.48	0.090	98.89
Poly (3^{rd})	25.14	0.111	**1.80**
Poly (10^{th})	24.38	0.120	7.71
Bézier	27.19	0.060	8.78
Ours	**27.21**	**0.053**	5.63

(b) Loss function

	PSNR↑	LPIPS↓
w/o $\mathcal{L}_{\text{pc}}$	17.49	0.853
w/o $\mathcal{L}_{\text{gc}}$	26.33	0.067
w/o $\mathcal{L}_{\text{d-pc}}$	26.18	0.066
w/o $\mathcal{L}_{\text{M}}$	26.34	0.088
Ours	**27.21**	**0.053**

(c) Motion-Adaptive Control points Pruning

	PSNR↑	LPIPS↓	g_{def} (ns)↓
w/o MACP ($N_c = 4$)	26.62	0.065	**5.34**
w/o MACP ($N_c = N_f$)	27.08	0.054	6.11
Ours	**27.21**	**0.053**	5.63

Table 3. **Ablation studies.** We ablate our framework and report the average results on the NVIDIA dataset with the same setting as Novel View Synthesis experiment in Sec. 5.1.

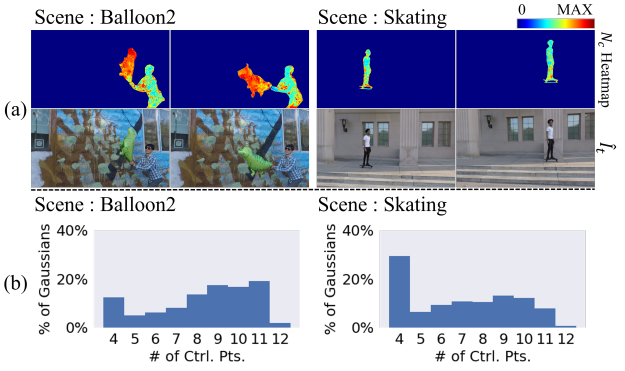


Figure 8. **Analysis of MACP's Efficacy.** (a) N_c Heatmaps as the averaged N_c values of dynamic 3D Gaussians and their corresponding rendered frames $\hat{I}_t$ for 'Balloon2' and 'Skating' scenes. (b) Histograms of the number of control points (N_c) in percentages (%) of dynamic 3D Gaussians in two scenes.

dynamic 3D Gaussians can be represented using a minimal N_c, thanks to our MACP. While 'Balloon2' has more evenly distributed N_c due to more complex and diverse motion.

Loss Functions. Table 3-(b) shows the effectiveness of each loss for our overall SplineGS architecture. As noted, no consistent camera parameters can be learned without $\mathcal{L}_{\text{pc}}$, drastically impacting the rendering quality of the dynamic 3D Gaussians. Moreover, our $\mathcal{L}_{\text{gc}}$, $\mathcal{L}_{\text{d-pc}}$ and $\mathcal{L}_{\text{M}}$ can considerably impact the overall rendering quality.

6. Conclusion

We present SplineGS, a COLMAP-free dynamic 3DGS framework designed for novel spatio-temporal view synthesis from monocular videos. Leveraging our innovative Motion-Adaptive Spline (MAS) for dynamic motion modeling, SplineGS efficiently renders high-quality novel views from complex in-the-wild videos. The effectiveness of our approach is validated through extensive quantitative and qualitative comparisons, significantly outperforming the existing SOTA methods with very fast rendering speed.

Acknowledgement

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korean Government [Ministry of Science and ICT (Information and Communications Technology)] (Project Number: RS-2022-00144444, Project Title: Deep Learning Based Visual Representational Learning and Rendering of Static and Dynamic Scenes, 100%).

References

[1] Fast dynamic radiance fields with time-aware neural voxels. In *SIGGRAPH Asia 2022 Conference Papers*, 2022. 2

[2] J Harold Ahlberg, Edwin Norman Nilson, and Joseph Leonard Walsh. *The Theory of Splines and Their Applications: Mathematics in Science and Engineering: A Series of Monographs and Textbooks, Vol. 38*. Elsevier, 2016. 2, 3, 4

[3] ShahRukh Athar, Zexiang Xu, Kalyan Sunkavalli, Eli Shechtman, and Zhixin Shu. Rignerf: Fully controllable neural 3d portraits. In *CVPR*, 2022. 2

[4] Benjamin Attal, Jia-Bin Huang, Christian Richardt, Michael Zollhoefer, Johannes Kopf, Matthew O'Toole, and Changil Kim. Hyperreel: High-fidelity 6-dof video with ray-conditioned sampling. In *CVPR*, 2023. 2

[5] Jeongmin Bae, Seoha Kim, Youngsik Yun, Hahyun Lee, Gun Bang, and Youngjung Uh. Per-gaussian embedding-based deformation for deformable 3d gaussian splatting. In *European Conference on Computer Vision*, pages 321–335. Springer, 2024. 6

[6] Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes. In *CVPR*, 2023. 2

[7] Mengyu Chu, You Xie, Jonas Mayer, Laura Leal-Taixé, and Nils Thuerey. Learning temporal coherence via self-supervision for gan-based video generation. *ACM Transactions on Graphics (TOG)*, 2020. 7

[8] C De Boor. A practical guide to splines. *Springer-Verlag google schola*, 1978. 2, 3, 4

[9] Gerald Farin. *Curves and surfaces for CAGD: a practical guide*. Elsevier, 2001. 2, 7, 8

[10] Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In *CVPR*, 2023. 2

[11] Yang Fu, Sifei Liu, Amey Kulkarni, Jan Kautz, Alexei A. Efros, and Xiaolong Wang. Colmap-free 3d gaussian splatting. In *CVPR*, 2024. 3

[12] Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang. Dynamic view synthesis from dynamic monocular video. In *ICCV*, 2021. 2, 6, 7

[13] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012. 5

[14] Yi-Hua Huang, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. Sc-gs: Sparse-controlled gaussian splatting for editable dynamic scenes. *CVPR*, 2024. 2, 6

[15] Woobin Im, Geonho Cha, Sebin Lee, Jumin Lee, Juhyeong Seon, Dongyoon Wee, and Sung-Eui Yoon. Regularizing dynamic radiance fields with kinematic fields. In *European Conference on Computer Vision*, pages 312–328. Springer, 2024. 6

[16] Wei Jiang, Kwang Moo Yi, Golnoosh Samei, Oncel Tuzel, and Anurag Ranjan. Neuman: Neural human radiance field from a single video. In *ECCV*, 2022. 2

[17] Yifan Jiang, Peter Hedman, Ben Mildenhall, Dejia Xu, Jonathan T Barron, Zhangyang Wang, and Tianfan Xue. Alignerf: High-fidelity neural radiance fields via alignment-aware training. In *CVPR*, 2023. 3

[18] Moritz Kappel, Florian Hahlbohm, Timon Scholz, Susana Castillo, Christian Theobalt, Martin Eisemann, Vladislav Golyanik, and Marcus Magnor. D-npc: Dynamic neural point clouds for non-rigid view synthesis from monocular video. *arXiv preprint arXiv:2406.10078*, 2024. 6

[19] Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker: It is better to track together. *arXiv*, 2023. 4

[20] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 2023. 2, 3, 4, 7

[21] Junoh Lee, Chang-Yeon Won, Hyunjun Jung, Inhwan Bae, and Hae-Gon Jeon. Fully explicit dynamic gaussian splatting. In *NeurIPS*, 2024. 3, 6

[22] Yao-Chih Lee, Zhoutong Zhang, Kevin Blackburn-Matzen, Simon Niklaus, Jianming Zhang, Jia-Bin Huang, and Feng Liu. Fast view synthesis of casual videos with soup-of-planes. In *ECCV*, 2024. 3, 6

[23] Jiahui Lei, Yijia Weng, Adam Harley, Leonidas Guibas, and Kostas Daniilidis. Mosca: Dynamic gaussian fusion from casual videos via 4d motion scaffolds. *arXiv*, 2024. 3, 6

[24] Zhan Li, Zhang Chen, Zhong Li, and Yi Xu. Spacetime gaussian feature splatting for real-time dynamic view synthesis. In *CVPR*. 1, 2, 3, 6, 7, 8

[25] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *CVPR*, 2021. 2, 7

[26] Zhengqi Li, Qianqian Wang, Forrester Cole, Richard Tucker, and Noah Snavely. Dynibar: Neural dynamic image-based rendering. In *CVPR*, 2023. 2

[27] Yiqing Liang, Numair Khan, Zhengqin Li, Thu Nguyen-Phuoc, Douglas Lanman, James Tompkin, and Lei Xiao. Gaufre: Gaussian deformation fields for real-time dynamic novel view synthesis. *arXiv*, 2023. 2, 3

[28] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. Barf: Bundle-adjusting neural radiance fields. In *ICCV*, 2021. 3

[29] Philipp Lindenberger, Paul-Edouard Sarlin, Viktor Larsson, and Marc Pollefeys. Pixel-perfect structure-from-motion with featuremetric refinement. In *ICCV*, 2021. 3

[30] Yu-Lun Liu, Chen Gao, Andreas Meuleman, Hung-Yu Tseng, Ayush Saraf, Changil Kim, Yung-Yu Chuang, Johannes Kopf, and Jia-Bin Huang. Robust dynamic radiance fields. In *CVPR*, 2023. 2, 3, 5, 6, 7

[31] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *3DV*, 2024. 2

[32] Quan Meng, Anpei Chen, Haimin Luo, Minye Wu, Hao Su, Lan Xu, Xuming He, and Jingyi Yu. Gnerf: Gan-based neural radiance field without posed camera. In *ICCV*, 2021. 3

[33] Xingyu Miao, Yang Bai, Haoran Duan, Fan Wan, Yawen Huang, Yang Long, and Yefeng Zheng. Ctnerf: Cross-time transformer for dynamic neural radiance field from monocular video. *Pattern Recognition*, 156:110729, 2024. 6

[34] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 2, 5

[35] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *ICCV*, 2021. 2

[36] 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.*, 2021. 2

[37] Keunhong Park, Philipp Henzler, Ben Mildenhall, Jonathan T. Barron, and Ricardo Martin-Brualla. Camp: Camera preconditioning for neural radiance fields. *ACM Trans. Graph.*, 2023. 3

[38] Luigi Piccinelli, Yung-Hsu Yang, Christos Sakaridis, Mattia Segu, Siyuan Li, Luc Van Gool, and Fisher Yu. Unidepth: Universal monocular metric depth estimation. In *CVPR*, 2024. 4, 5

[39] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alex Sorkine-Hornung, and Luc Van Gool. The 2017 davis challenge on video object segmentation. *arXiv*, 2018. 1, 6, 7

[40] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *CVPR*, 2021. 2

[41] Vincent Raoult, Sarah Reid-Anderson, Andreas Ferri, and Jane E Williamson. How reliable is structure from motion (sfm) over time and between observers? a case study using coral reef bommies. *Remote Sensing*, 2017. 3

[42] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016. 1, 2, 3, 5, 6, 7

[43] Ruizhi Shao, Zerong Zheng, Hanzhang Tu, Boning Liu, Hongwen Zhang, and Yebin Liu. Tensor4d: Efficient neural 4d decomposition for high-fidelity dynamic reconstruction and rendering. In *CVPR*, 2023. 2

[44] Liangchen Song, Anpei Chen, Zhong Li, Zhang Chen, Lele Chen, Junsong Yuan, Yi Xu, and Andreas Geiger. Nerfplayer: A streamable dynamic scene representation with decomposed neural radiance fields. *IEEE Transactions on Visualization and Computer Graphics*, 2023. 2

[45] Carole H. Sudre, Wenqi Li, Tom Vercauteren, Sebastien Ourselin, and M. Jorge Cardoso. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, 2017. 5

[46] Fengrui Tian, Shaoyi Du, and Yueqi Duan. MonoNeRF: Learning a generalizable dynamic radiance field from monocular videos. In *ICCV*, 2023. 6

[47] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In *ICCV*, 2021. 2

[48] Diwen Wan, Ruijie Lu, and Gang Zeng. Superpoint gaussian splatting for real-time high-fidelity dynamic scene reconstruction. *arXiv preprint arXiv:2406.03697*, 2024. 6

[49] 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. 6

[50] Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. Nerf-: Neural radiance fields without known camera parameters. *CoRR*, 2021. 3

[51] Chung-Yi Weng, Brian Curless, Pratul P. Srinivasan, Jonathan T. Barron, and Ira Kemelmacher-Shlizerman. HumanNeRF: Free-viewpoint rendering of moving people from monocular video. In *CVPR*, 2022. 2

[52] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. In *CVPR*. 2, 6, 7, 8

[53] Gengshan Yang, Minh Vo, Neverova Natalia, Deva Ramanan, Vedaldi Andrea, and Joo Hanbyul. Banmo: Building animatable 3d neural models from many casual videos. In *CVPR*, 2022. 2

[54] Jinyu Yang, Mingqi Gao, Zhe Li, Shang Gao, Fangjing Wang, and Feng Zheng. Track anything: Segment anything meets videos. *arXiv*, 2023. 5

[55] 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. 1, 2, 6, 7, 8

[56] 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*, 2020. 1, 6, 7

[57] 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. 6

[58] Kaichen Zhou, Jia-Xing Zhong, Sangyun Shin, Kai Lu, Yiyuan Yang, Andrew Markham, and Niki Trigoni. Dynpoint: Dynamic neural point for view synthesis. *Advances in Neural Information Processing Systems*, 36:69532–69545, 2023. 6