EvoGS: 4D Gaussian Splatting as a Learned Dynamical System

Arnold Caleb Asiimwe
Princeton University
asiimwe@cs.princeton.edu

Carl Vondrick
Columbia University
vondrick@cs.columbia.edu

Abstract

*We reinterpret 4D Gaussian Splatting as a continuous-time dynamical system, where scene motion arises from integrating a learned neural dynamical field rather than applying per-frame deformations. This formulation, which we call **EvoGS**, treats the Gaussian representation as an evolving physical system whose state evolves continuously under a learned motion law. This unlocks capabilities absent in deformation-based approaches: (1) sample-efficient learning from sparse temporal supervision by modeling the underlying motion law; (2) temporal extrapolation enabling forward and backward prediction beyond observed time ranges; and (3) compositional dynamics that allow localized dynamics injection for controllable scene synthesis. Experiments on dynamic scene benchmarks show that **EvoGS** achieves better motion coherence and temporal consistency compared to deformation-field baselines while maintaining real-time rendering.*[1]

1. Introduction

Fig. 1 *"Everything flows"*—Heraclitus [23]

Dynamic scene reconstruction has traditionally focused on recovering time-varying geometry and appearance from video. While early progress was driven by dynamic extensions of NeRF [36], these approaches rely on learned deformation fields that warp a canonical scene to each timestep [29, 39, 41, 42]. Although conceptually elegant, deformation-based NeRFs require dense and regular frame sampling, and their deformation fields often collapse when supervision becomes sparse or irregular. They are also computationally costly, as every frame requires evaluating both the canonical radiance field and its deformation.

To improve scalability and stability, subsequent works represent time as an explicit axis in a factorized 4D grid [3, 5, 44], enabling faster, more robust rendering. However, these grid-based models still treat time as a discrete index and therefore cannot reason over missing frames or extrapolate beyond the observed temporal window. Their motion representation is descriptive rather than predictive.

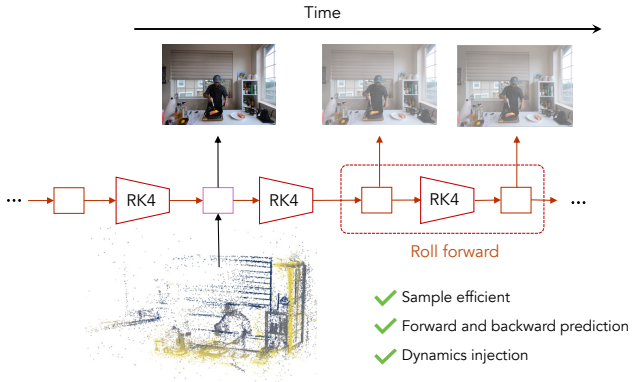

Figure 1. EvoGS learns a continuous-time dynamical system that governs the evolution of Gaussian primitives. A neural velocity field v_θ drives their motion through numerical integration. Unlike discrete deformation-based approaches (Fig. 2), EvoGS reconstructs unseen timesteps by following the learned dynamics, enabling continuous-time extrapolation and controllable motion composition.

Building upon explicit representations, recent advances in 4D Gaussian Splatting [10, 20, 32, 49, 52] model dynamic scenes by updating Gaussian parameters at discrete timestamps. While these approaches differ in how the updates are predicted—ranging from independent per-frame Gaussian clouds [34] to framewise deformation fields [8, 17, 49, 52] via a canonical-to-world mapping (Fig. 2)—they all share the same discrete-time assumption: motion is represented only at the observed frames. As a result, they struggle to maintain coherent trajectories when temporal observations are sparse, irregular, or missing entirely.

Unfortunately, partial observability is the norm outside controlled lab environments. Real-world video streams suffer from missing frames due to camera outages, irregular motion-capture sessions, rolling-shutter artifacts, and dropped frames caused by unreliable networks. In such settings, discrete deformation-based methods often fail to maintain physically meaningful trajectories at unseen timesteps: NeRF variants may freeze or produce ghost arti-

[1] Project page: https://arnold-caleb.github.io/evogs.

facts , while 4D Gaussian methods can smoothly interpolate yet drift away from the correct motion path (Fig. 6).

This reveals a fundamental limitation: ***when time is discretized, models cannot robustly interpolate missing frames or reliably predict future ones.*** Yet both capabilities are crucial. Robust interpolation enables faithful reconstruction under sparse temporal observations, and the ability to predict future motion opens the door to high-stakes applications where anticipating outcomes—such as potential collisions or system failures—can prevent catastrophic events. To address these shortcomings, we propose to reinterpret dynamic scene modeling through the lens of **continuous-time dynamical systems** rather than discrete collections of warped frames. In our formulation, each Gaussian primitive behaves like a particle governed by an underlying velocity field $v_\theta(\mathbf{x}, t)$. Rather than predicting per-frame displacements, the model learns this velocity field directly, and Gaussian parameters evolve through numerical integration (Fig. 1). This allows the scene to be rendered at any continuous moment—including frames that were unobserved during training or timesteps far beyond the original video. We call this framework **EvoGS**.

By treating 4D Gaussian splatting as a learned dynamical system, EvoGS inherits the rendering efficiency of explicit Gaussians while enabling capabilities absent in prior work: Sparse temporal reconstruction (§4.1): EvoGS learns coherent motion from as little as one-third of the total frames. Future and past prediction: Continuous integration supports extrapolation for simulating unseen motion. Compositional motion editing: The learned velocity field enables blending, injecting, or modulating local dynamics (§4.2).

Conceptually, EvoGS echoes ideas from dynamical systems, neural ODEs [6], and filtering-based models [18, 19] and combines prediction from continuous dynamics, correction from observations, and stabilization from temporal consistency priors. This yields coherent scene evolution even under sparse supervision and enables reliable reconstruction and prediction beyond the capabilities of existing deformation-based methods.

2. Related Work

We review three areas that inform our approach. Sec. 2.1 covers continuous-time dynamical formulations that motivate viewing scene evolution through learned velocity fields. Sec. 2.2 surveys dynamic neural scene representations, and Sec. 2.3 discusses recent Gaussian approaches incorporating motion priors or learned dynamics.

2.1. Dynamical Formulations

Modeling time-varying physical systems has a long history in computer graphics and physics-based simulation, from early elastically deformable models [47] to classical fluid solvers [1, 46]. More recent work incorporates differentiable physics and learning-based surrogates, enabling neural networks to approximate or constrain physical dynamics [9, 11, 12, 22, 38, 48].

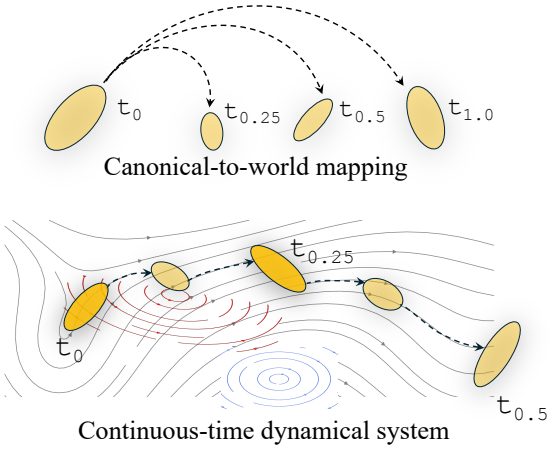

Figure 2. **Top:** Canonical deformation methods assign each timestamp an independent mapping from a shared canonical space to produce a set of per-frame transformations (learn what the scene looks like at each time t). **Bottom:** EvoGS instead learns a continuous velocity field that governs Gaussian evolution through time. Dynamics arise from integrating this field to produce reversible trajectories and coherent motion between arbitrarily spaced timestamps. The swirling field visualization shows how local dynamical structure emerges and how injected motion (blue and red) blends into the learned global flow.

Recent approaches [7, 54] combine differentiable rendering with physics-driven simulation to reconstruct or predict fluid motion directly from video, reflecting a shift toward neural dynamical systems that jointly model perception, geometry, and motion. These ideas align with methods that approximate continuous evolution through learned velocity fields rather than discrete timesteps—most notably neural ODEs [6]. Within the broader context of physics-informed learning, further works demonstrate how learned surrogates can accelerate fluid simulation [24], how differentiable solvers enable gradient-based reconstruction of fluid phenomena from imagery [45], and how PDE-constrained neural networks infer motion from sparse observations [15].

2.2. Dynamic Scene Representations

The introduction of 3D Gaussian Splatting [21] marked a shift from implicit neural fields [37, 39, 40] to explicit, differentiable point-based primitives for radiance field rendering. By representing a scene as a collection of anisotropic Gaussians with learnable position, orientation, opacity, and color, these methods achieve high-fidelity results through differentiable rasterization rather than volumetric integration. Their efficiency and photorealistic quality have established Gaussian splatting as a leading paradigm for explicit neural scene representation.

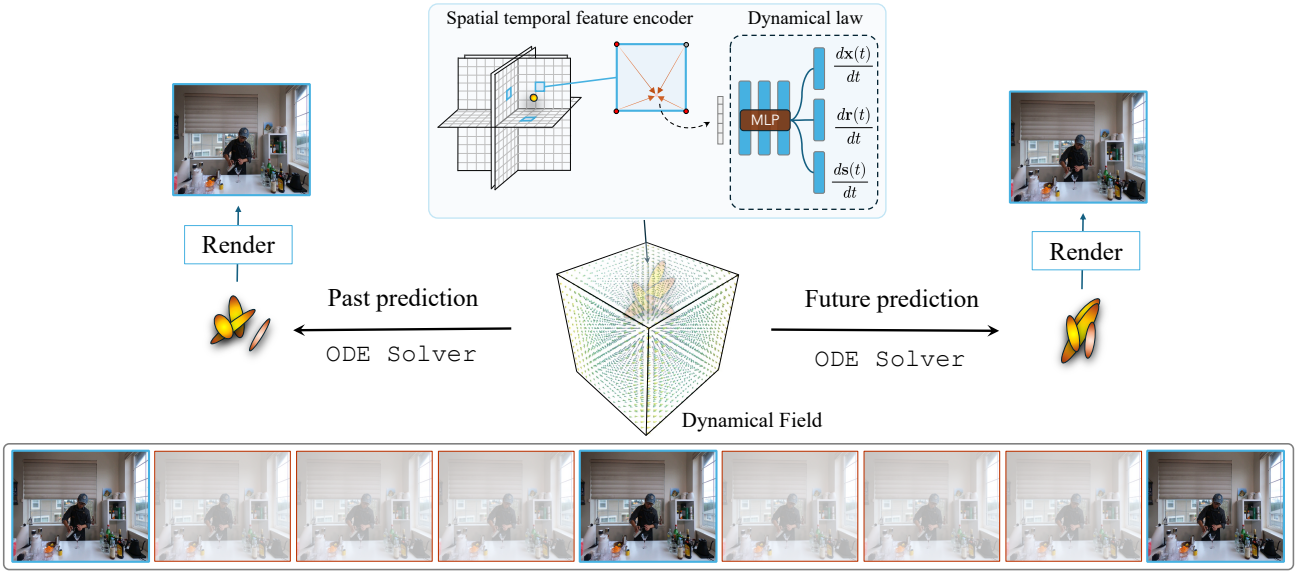

Figure 3. **Overview of EvoGS:** Given input frames (blue) with photometric supervision, each Gaussian is embedded using 4D spatiotemporal features and evolved through a learned continuous-time velocity field. A neural dynamical law predicts time derivatives of Gaussian attributes, and an ODE solver integrates these dynamics forward or backward to produce unseen future and past states (red), which arise purely from continuous-time evolution.

Extending Gaussian splatting to dynamic scenes requires modeling how Gaussian parameters evolve through time while maintaining temporal coherence and rendering efficiency. Most formulations have generalized static Gaussians into the spatiotemporal domain by learning per-frame transformations of a canonical configuration, effectively treating each timestep as an independent deformation of the scene [8, 17, 49, 52]. Subsequent approaches introduced temporally shared Gaussian attributes to improve coherence [56], surfel-based deformation models for finer control of local motion and geometry [35], and disentangled or editable formulations that separate static and dynamic components or apply segmentation-based priors for controllable motion [26, 28].

2.3. Dynamical Gaussian Methods

Recent extensions of Gaussian splatting have introduced explicit motion modeling and learned dynamics, moving beyond frame-wise deformations—several works extend static 3D Gaussians to dynamic settings through temporally coupled transformations, motion-aware attributes, or latent motion factorization [13, 16, 25, 28]. Others focus on motion guidance and continuous motion cues to handle large or blurred motions [27, 57]. Inspired by physical systems, some approaches embed motion laws within Gaussian primitives, treating each as a particle evolved under continuum or flow-based dynamics [51]. Other formulations express temporal variations—such as position or covariance—as compact parametric functions of time, e.g., polynomial or Fourier expansions [31]. Self-supervised variants further learn scene flow for dynamic or unlabeled environments [50].

Despite these advances, existing methods still rely on discrete temporal updates or per-frame optimization, requiring dense supervision and struggling to extrapolate motion beyond observed frames. In contrast, our approach models Gaussian evolution as a continuous-time process governed by a neural velocity field $v_\theta(\mathbf{x}, t)$, enabling controllable motion composition, sparse-frame training, and temporally coherent rollouts.

3. Method

This section introduces EvoGS (Fig. 3), a continuous-time formulation of dynamic Gaussian splatting. We first outline the model design (Sec.3.1), then describe the feature encoder (Sec.3.2), the neural dynamical law (Sec.3.3), and a Kalman-inspired stabilization mechanism (Sec.3.4). We conclude with the rendering process and training objective (Sec.3.5).

3.1. Overview

We treat each Gaussian as a particle evolving under a learned continuous-time dynamical system. Its trajectory is defined by a neural velocity field conditioned on local spatiotemporal features. Following standard practice in Gaussian Splatting [21], all Gaussians are initialized from a point cloud reconstructed via structure-from-motion (SfM). EvoGS (Fig. 3) then consists of: (1) a 4D feature

encoder that produces local embeddings from a factorized space–time representation (e.g., HexPlane [3]), (2) a neural dynamical law predicting instantaneous time derivatives of Gaussian attributes, and (3) a differentiable ODE integrator that advances these states in time.

3.2. Spatiotemporal Feature Encoding

Each Gaussian center $\mathbf{p}_i = (x_i, y_i, z_i)$ at time t is embedded via bilinear interpolation over six space–time factorization planes $\{\mathbf{P}_{xy}, \mathbf{P}_{xz}, \mathbf{P}_{yz}, \mathbf{P}_{xt}, \mathbf{P}_{yt}, \mathbf{P}_{zt}\}$:

$$\mathbf{f}_i(t) = \Phi(\mathbf{p}_i, t),$$

where Φ denotes the differentiable lookup from the 4D grid. These features encode local geometry and motion cues and condition the velocity field used in the dynamical update.

3.3. Neural Dynamical Law

Each Gaussian primitive has a state

$$\mathbf{x}_i(t) = [\mathbf{p}_i(t),\, \mathbf{R}_i(t),\, \mathbf{S}_i(t),\, \mathbf{c}_i(t),\, \alpha_i(t)],$$

where $\mathbf{p}_i$ is its 3D position, $\mathbf{R}_i$ its rotation (parameterized via an exponential-map update), $\mathbf{S}_i$ its anisotropic scale, $\mathbf{c}_i$ its color, and α_i its opacity. The state evolves according to the continuous-time ODE

$$\frac{d\mathbf{x}_i}{dt} = \mathbf{v}_\theta(\mathbf{x}_i(t),\, \mathbf{f}_i(t),\, t), \tag{1}$$

where $\mathbf{v}_\theta$ is a lightweight MLP predicting derivatives of position, rotation, and scale. We integrate this ODE with a differentiable solver (RK4), enabling both forward and backward temporal propagation:

$$\begin{aligned} \mathbf{x}_i(t_1) &= \mathrm{RK4}(\mathbf{x}_i(t_0),\, t_0,\, \Delta t,\, \mathbf{v}_\theta), \\ \mathbf{x}_i(t_0) &= \mathrm{RK4}(\mathbf{x}_i(t_1),\, t_1,\, -\Delta t,\, \mathbf{v}_\theta). \end{aligned} \tag{2}$$

Bidirectional integration yields reversible dynamics and allows the model to propagate motion through missing frames or ambiguously observed regions.

3.4. Gaussian Waypoints for Motion Stabilization

Continuous ODE integration can accumulate drift over long temporal horizons due to numerical error and locally underconstrained motion. In classical filtering, such drift is controlled by alternating prediction and correction steps. While a full Kalman filter is infeasible here—given nonlinear dynamics, millions of latent states, and non-Gaussian rendering losses—we adopt a related idea using *Gaussian waypoints*. During training, a small number of anchor snapshots $\mathcal{A} = \{t_1^{(a)}, t_2^{(a)}, \ldots\}$ store the Gaussian states at fixed times. These anchors act as sparse pseudo-observations of the underlying dynamical system.

For any target frame at time t, we locate the nearest past anchor $t^{(a)}$ and reinitialize the ODE state using the stored Gaussian parameters at $t^{(a)}$, then integrate forward from $t^{(a)}$ to t. That way, the effective integration horizon is reduces so that drift accumulation is reduced and prevents diverging during long rollouts.

Optionally, we penalize deviations between the integrated state and the stored anchor snapshot itself:

$$\mathcal{L}_{\text{anchor}} = \sum_{t^{(a)} \in \mathcal{A}} \|\mathbf{x}(t^{(a)}) - \hat{\mathbf{x}}(t^{(a)})\|_2^2,$$

where $\hat{\mathbf{x}}(t^{(a)})$ is the anchor state and $\mathbf{x}(t^{(a)})$ is the state obtained by integrating from the preceding anchor. This encourages consistency with anchor waypoints while still allowing smooth continuous-time evolution between them. In contrast to classical filters, we do not maintain explicit velocity estimates or covariance; the anchors function solely as sparse, fixed reference states that constrain long-term integration.

3.5. Rendering and Objective

At each target timestamp t_1, the evolved Gaussians $\mathcal{G}(t_1)$ are rendered using differentiable Gaussian splatting [21]. Supervision is provided by a standard photometric reconstruction loss (L1, optionally combined with SSIM/LPIPS).

To encourage stable motion and suppress drift, we include temporal smoothness on the spatiotemporal planes (plane TV and time-smoothing), as well as a velocity-coherence regularizer to encourage nearby Gaussians to move consistently. When anchor waypoints are enabled, we apply a soft anchor-consistency term that pulls integrated states toward stored anchor snapshots.

The full training objective is:

$$\mathcal{L} = \mathcal{L}_{\text{photo}} + \lambda_{\text{coh}}\mathcal{L}_{\text{coh}} + \lambda_{\text{anchor}}\mathcal{L}_{\text{anchor}} + \lambda_{\text{tv}}\mathcal{L}_{\text{tv}}, \tag{3}$$

where $\mathcal{L}_{\text{coh}}$ enforces velocity coherence, $\mathcal{L}_{\text{anchor}}$ applies the optional anchor constraint, and $\mathcal{L}_{\text{tv}}$ smooths the spatiotemporal feature fields.

4. Experiments

We evaluate `EvoGS` on synthetic and real-world datasets, comparing against state-of-the-art dynamic scene reconstruction methods [3, 44, 49]. Section 4.1 describes implementation details, datasets, and experimental settings. Section 4.2 demonstrates external motion injection and controllable dynamics. Section 4.3 provides ablation studies and analysis.

4.1. Experimental Setup

Implementation Details. Our model is implemented in PyTorch and trained on a single NVIDIA L40 GPU. We adopt the optimization settings of [49], with minor adjustments for continuous-time dynamics. To assess temporal robustness, we primarily evaluate in the *sparse-frame*

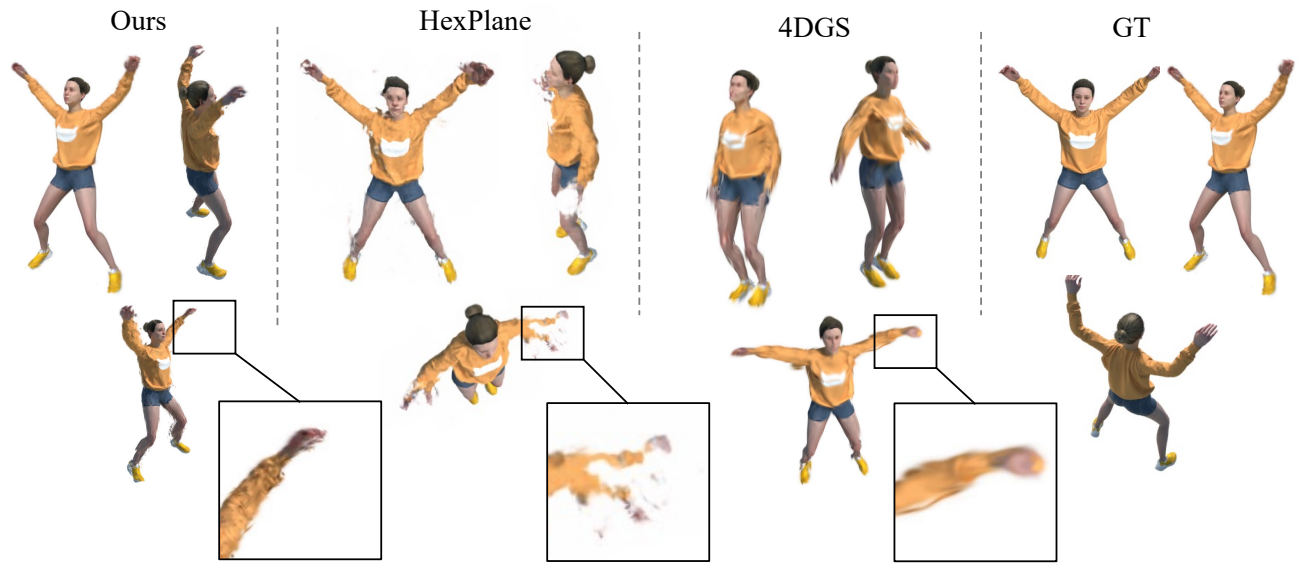

Figure 4. Comparison of EvoGS on reconstruction of unseen dynamic human motion on the Jumping Jacks scene. Compared to HexPlane [3] and 4DGS [49], which breakdown for unseen timesteps (e.g., limbs rupturing or blurring)

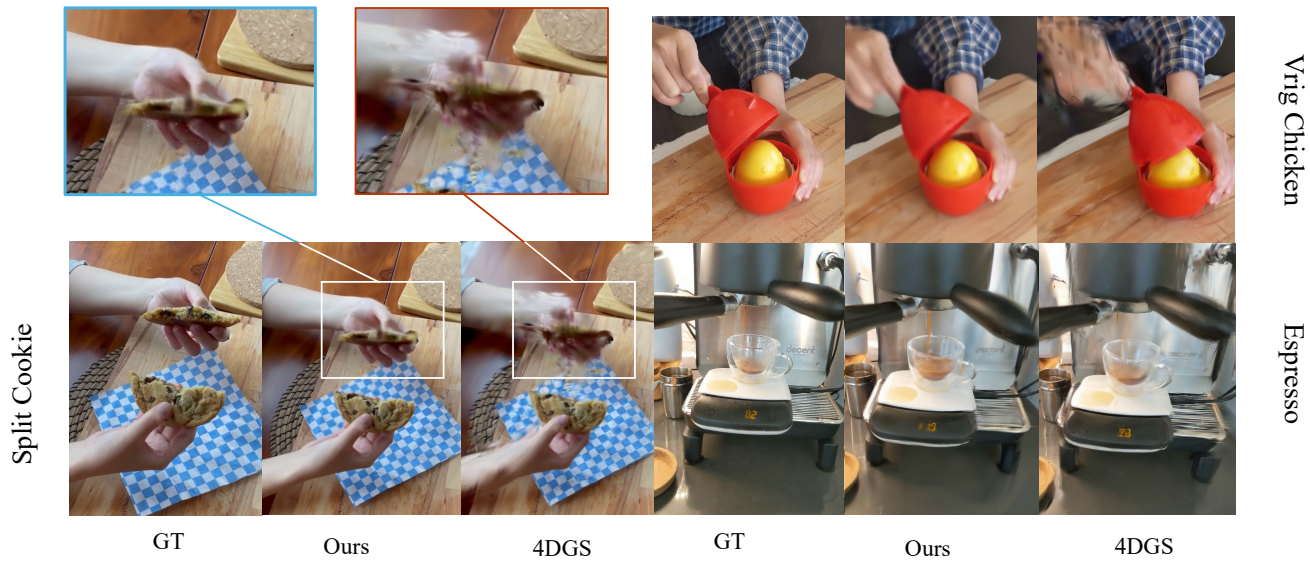

Figure 5. **Extrapolation on real monocular dynamic scenes.** Comparison on Split Cookie, Vrig Chicken, and Espresso sequences, where the model must predict frames beyond the observed time range. We include comparisons to [43, 44, 52] in suppl. for completeness.

regime by dropping frames during training (Figs. 4, 5, 6, 8). The datasets used are described next.

Datasets. For synthetic evaluation, we use the D-NeRF dataset [43], which contains monocular dynamic scenes with 50–200 frames and randomly varying camera trajectories. For real-world evaluation, we use the Neural 3D Video (N3DV) dataset [30], which provides multi-view dynamic captures with calibrated poses and complex nonrigid motion, and the Nerfies dataset [39], consisting of monocular captures with moderate to fast nonrigid motion. All experiments use the provided camera parameters. For each sequence, we uniformly subsample frames for training and evaluation, as detailed below.

Sparse-Frame and Extrapolation Settings. To evaluate temporal generalization, we train using every k-th frame ($k \in \{2, 4, 8, 10\}$) of each sequence. On N3DV (300 frames), this results in only $300/k$ training frames. We report results for strides $k = 2$ and $k = 8$ in Table 1. We also evaluate *future extrapolation* (Fig. 6) by training only on the first 0.75 fraction of frames and predicting all unseen future

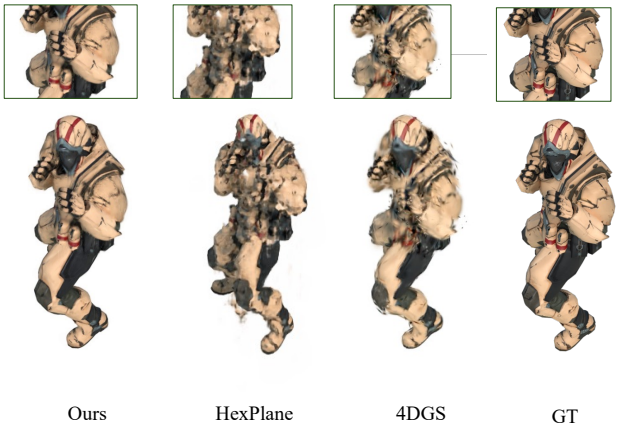

Figure 6. **Interpolation on the Hook scene.** EvoGS maintains coherent geometry and motion, whereas HexPlane freezes the dynamics and 4DGS produces an over-smoothed intermediate frame.

frames. The same protocol can be applied for *backward rollout*, where the learned velocity field is integrated backward to reconstruct earlier frames. The sparse-frame and extrapolation settings are used consistently across N3DV, D-NeRF, and Nerfies datasets.

Waypoint initialization To reduce long-term drift during continuous integration, we introduce three temporal anchors placed at the start, midpoint, and end of each sequence. Each anchor corresponds to a 3D Gaussian state rendered at its timestamp and acts as a soft constraint that keeps the learned trajectories consistent over long time horizons.

Metrics. We evaluate reconstruction quality using standard photometric metrics: PSNR, SSIM, and LPIPS [55]. All results are reported on held-out frames under the same sparse-frame or extrapolation protocols used in training.

4.2. Compositional and Controllable Dynamics

A key advantage of representing scene motion as a continuous-time velocity field is enabling *controllable motion synthesis*. Since dynamics are encoded as a vector field we can directly manipulate, mix, or replace portions of the flow to produce new motion without retraining (Fig. 7).

Velocity field composition and local dynamics injection. Formally, given two velocity fields—a learned field $\mathbf{v}_\theta$ and an external field $\mathbf{v}_{\text{ext}}$ (e.g., a user-defined motion or a field borrowed from another model)—we can form a spatially mixed field, enabling a simple *vector-field algebra*:

$$\mathbf{v}_{\text{mix}}(\mathbf{x}, t) = \lambda(\mathbf{x})\, \mathbf{v}_\theta(\mathbf{x}, t) + \big(1 - \lambda(\mathbf{x})\big)\, \mathbf{v}_{\text{ext}}(\mathbf{x}, t). \quad (4)$$

where $\lambda(\mathbf{x}) \in [0, 1]$ is a spatial mask controlling which region follows which dynamics. This allows selected objects to inherit new motion. Because EvoGS evolves Gaussians through continuous-time integration, $\mathbf{v}_{\text{mix}}$ yields smooth spatiotemporal transitions. Fig. 7 shows a rotational field combined injected into the learned field via

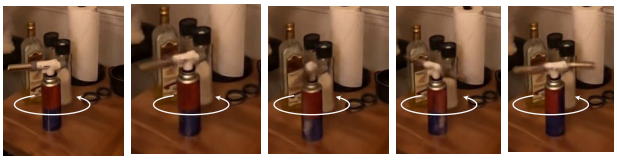

Figure 7. **Compositional Dynamics injection:** By locally blending a rotational velocity field (indicated by white arrow), EvoGS can inject new fields into a scene.

$$\mathbf{v}'(\mathbf{x}, t) = \lambda(\mathbf{x})\, \mathbf{v}_{\text{inj}}(\mathbf{x}, t) + (1 - \lambda(\mathbf{x}))\, \mathbf{v}_\theta(\mathbf{x}, t). \quad (5)$$

where $\mathbf{v}_{\text{inj}}$. Gaussians in the masked region follow v_{inj}, while the rest of the scene continues under v_θ which allows new motion to be created without retraining.

Object incompleteness and the need for recomposition. Injecting a new velocity field into a scene requires a complete 3D representation of the target object. However, the Gaussians associated with an object $\mathcal{G}_{\text{obj}}$ are often incomplete: because training cameras observe the object only from a subset of angles, large portions of its surface are undersampled or entirely missing. When the object is moved or rotated, these unseen regions become exposed and produce severe artifacts.

Geometry completion and reinsertion. We first isolate the target object using a 3D Gaussian segmentation mask $\lambda(\mathbf{x})$ [4]. To reconstruct the missing geometry, we render segmented ground-truth images from the original camera views and use them as input to Zero123 [33] to synthesize novel viewpoints that were never observed. These real and synthesized views supervise a second-pass 3D Gaussian optimization applied only to $\mathcal{G}_{\text{obj}}$, enabling densification and completion of the object's geometry. The refined Gaussians are then reinserted into the full scene, and the injected velocity field $\mathbf{v}_{\text{inj}}$ is applied to them during continuous-time evolution.

4.3. Ablations and Analysis

Integration order. We compare a fourth-order Runge–Kutta solver (RK4) to a first-order Euler integrator in Fig. 9. While Euler integration is numerically cheap, it accumulates drift rapidly and incoherent motion across different gaussians as the system is rolled forward or backward in time. RK4, by contrast, produces stable trajectories and preserves Gaussian structure, however

Figure 8. **Interpolation under sparse-frame training on the N3DV "flame salmon" scene.** K-Planes [44] and HexPlane [3] largely freeze the motion when frames are skipped, while 4DGS [49] preserves appearance but smoothes fast-moving regions (e.g., the hand becomes shortened or smeared).

Table 1. **Sparse-frame training results on the N3DV "coffee martini" scene.** We evaluate reconstruction fidelity when training on every k-th frame (here k=2, 8). We include results on training all frames for completion. ‡Full supervision results are obtained from [49].

	Full supervision			$k=2$			$k=8$		
Model	PSNR↑	D-SSIM↓	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
HexPlane‡ [3]	**31.70**	**0.014**	0.075	26.14	0.830	0.30	24.39	0.782	0.39
KPlanes‡ [44]	31.63	-	-	26.28	0.832	0.29	24.52	0.785	0.39
D-NeRF [43]	29.40	0.028	0.112	23.80	0.801	0.48	23.72	0.721	0.43
Deformable 3DGS [52]	30.52	0.022	0.084	25.40	0.84	0.30	22.12	0.742	0.41
4DGS‡ [49]	31.15	0.016	**0.049**	27.65	0.878	0.27	26.45	0.846	**0.22**
Ours	30.82	0.022	0.085	**28.25**	**0.914**	**0.20**	**26.90**	**0.870**	0.26

Figure 9. Euler integration rapidly accumulates temporal drift. RK4 produces smooth, consistent trajectories for both interpolation and extrapolation.

under extremely long-horizon extrapolation the gaussian structure starts to fall apart (Fig. 10).

Effect of Gaussian waypoints. Removing Gaussian waypoints increases temporal drift because the ODE is integrated from a single fixed reference state and errors compound over long sequences Tab. 2. Waypoints act as sparse re-initialization states: at each target timestamp the system integrates only from the nearest stored anchor, preventing the accumulation of small numerical errors. Without waypoints, we observe increasing trajectory divergence and noticeable spatial jitter like in Fig. 10.

Sparse-frame robustness. With moderate sparsity (e.g., training on every 8th frame), deformation-based and factorized spatiotemporal grids baselines struggle to infer plausible intermediate motion (Fig. 8), while our continuous-time formulation maintains coherent trajectories through the learned velocity field. Under extreme sparsity (e.g., one frame every 20), the dynamics become underconstrained and the advantage over deformation-based models diminishes—both behave similarly when temporal supervision is insufficient. Tab. 1 summarizes performance across sparsity levels.

5. Discussion

We show that reconstruction and prediction can be expressed within the same continuous dynamical space. Instead of optimizing per-frame deformations, the model learns a velocity field that governs scene evolution across both observed and unobserved timestamps. This shared representation reduces temporal discontinuities and enables forward extrapolation and backward rollouts without retraining. Higher-order integration further stabilizes long-range behavior (Fig. 9), suggesting that continuous-time

Figure 10. Without Gaussian waypoints, long forward integration causes rollouts to slowly drift and distort the scene

Table 2. **Ablation study on "flame salmon scene":** Removing waypoints, coherence loss, or the HexPlane encoder degrades long-range prediction. Metrics are reported on frames held out for $t > 0.75$ (supervision on $t \leq 0.25$).

Model	PSNR↑	SSIM↑	LPIPS↓
Ours (w/o λ_{anchor})	24.1	0.846	0.23
Ours (w/o λ_{coh})	25.1	0.872	0.23
Ours (w/o hexplane)	23.3	0.847	0.24
Ours	**25.73**	**0.880**	**0.20**

Figure 11. Failure case: lack of physical reasoning in emergent dynamics. When presented with scenes requiring true physical understanding—such as liquid filling a glass—EvoGS can extrapolate the motion of rigid objects (e.g., the hand and cup) but fails to infer the emergent fluid behavior.

formulations provide a strong inductive bias for modeling dynamic 3D scenes.

Because motion is represented as a vector field, injecting external velocity fields provides a simple and expressive mechanism for editing 4D content. This vector-field algebra (Sec: 4.2) enables localized motion synthesis, mixing, or replacement—all without re-optimizing the entire scene. Such controllable dynamics hint at a broader direction: dynamic scene representations that behave like world models, in which motion rules can be modified, composed, or conditioned on external signals.

Our formulation suggests that continuous-time velocity fields may serve as a useful interface between reconstruction methods and video-generation models. Generative world models [2, 14, 53] typically operate on latent tokens or coarse implicit grids, whereas EvoGS evolves explicit 3D primitives that are directly renderable. Training such dynamical fields at larger scale—or conditioning them on text, audio, or actions—could enable generative 4D scenes with physically plausible, editable dynamics. Dynamic Gaussian splatting may thus form a bridge between reconstruction-centric 3D methods and generative video models.

Limitations and opportunities. Our approach is data-driven and inherits the biases and ambiguities present in the training video. In scenarios requiring genuine causal or physical reasoning, the learned velocity field may fail to generalize. For example, in sequences where a hand begins to pour water into a glass (Fig. 11), EvoGS can extrapolate the hand's motion but cannot infer fluid behavior or anticipate water–glass interaction—phenomena that fall outside the spatiotemporal patterns observed in the training frames. Likewise, under extreme temporal sparsity, the dynamics become underconstrained and gradually regress toward deformation-like behavior.

6. Conclusion

We introduced EvoGS, a dynamic Gaussian framework that models scene evolution through a continuous-time velocity field. Integrating Gaussian parameters over time yields a unified representation for reconstruction, interpolation, extrapolation, and controllable dynamics, without relying on per-frame deformations.

Acknowledgements This research was primarily done while ACA was an undergraduate at Columbia University and completed while ACA was a graduate student at Princeton University. We thank Prof. Felix Heide for insightful conversations during the preparation of the paper, and the I.I. Rabi Scholars program at Columbia for supporting this research.

References

[1] Robert Bridson. *Fluid Simulation for Computer Graphics*. A K Peters, 2008. 2

[2] J. Bruce, J. Schrittwieser, M. Mirza, et al. Genie: Generative interactive environments. *arXiv preprint arXiv:2402.15329*, 2024. 8

[3] Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes. *CVPR*, 2023. 1, 4, 5, 7

[4] Jiazhong Cen, Jiemin Fang, Chen Yang, Lingxi Xie, Xiaopeng Zhang, Wei Shen, and Qi Tian. Segment any 3d gaussians. *arXiv preprint arXiv:2312.00860*, 2023. 6

[5] Anpei Chen, Zexiang Zhang, G. Wang, R. Ding, X. Liu, J. Zhang, and J. Yu. TensoRF: Tensorial radiance fields. In *ECCV*, pages 272–289, 2022. 1

[6] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. 2

[7] Yitong Deng, Hong-Xing Yu, Diyang Zhang, Jiajun Wu, and Bo Zhu. Fluid simulation on neural flow maps. *ACM Transactions on Graphics (TOG)*, 42(6):244:1–244:15, 2023. 2

[8] Bardienus P. Duisterhof, Zhao Mandi, Yunchao Yao, Jia-Wei Liu, Jenny Seidenschwarz, Mike Zheng Shou, Deva Ramanan, Shuran Song, Stan Birchfield, Bowen Wen, and Jeffrey Ichnowski. Deformgs: Scene flow in highly deformable scenes for deformable object manipulation. In *Proceedings of the 16th International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2024. 1, 3

[9] Michael Eckert and Nils Thuerey. Scalarflow: A large-scale volumetric data set of real-world scalar transport flows for computer animation and machine learning. *ACM Transactions on Graphics (TOG)*, 38(4):1–15, 2019. 2

[10] Yutao Feng, Xiang Feng, Yintong Shang, Ying Jiang, Chang Yu, Zeshun Zong, Tianjia Shao, Hongzhi Wu, Kun Zhou, Chenfanfu Jiang, et al. Gaussian splashing: Dynamic fluid synthesis with gaussian splatting. *arXiv preprint arXiv:2401.15318*, 2024. 1

[11] Ernst Franz and Nils Thuerey. Global neural flow: Learning generalizable fluid dynamics from visual data. *ACM Transactions on Graphics (TOG)*, 40(6):1–14, 2021. 2

[12] James Gregson, Ivo Ihrke, and Wolfgang Heidrich. From capture to simulation: Connecting fluid reconstruction and simulation. *ACM Transactions on Graphics (TOG)*, 33(4): 1–11, 2014. 2

[13] Zhiyang Guo, Wengang Zhou, Li Li, Min Wang, and Houqiang Li. Motion-aware 3d gaussian splatting for efficient dynamic scene reconstruction. *arXiv preprint arXiv:2403.11447*, 2024. 3

[14] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018. 8

[15] Mouhammad El Hassan, Ali Mjalled, Philippe Miron, Martin Mönnigmann, and Nikolay Bukharin. Machine learning in fluid dynamics—physics-informed neural networks (pinns) using sparse data. *Fluids*, 10(9):226, 2025. 2

[16] X Hu et al. Motion decoupled 3d gaussian splatting for dynamic object representation with large motion from a monocular camera. *AAAI Conference on Artificial Intelligence (AAAI) 2025*, 2025. 3

[17] 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. *arXiv preprint arXiv:2312.14937*, 2023. 1, 3

[18] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82 (1):35–45, 1960. 2

[19] Rudolph E. Kalman and Richard S. Bucy. New results in linear filtering and prediction theory. *Journal of Basic Engineering*, 83(1):95–108, 1961. 2

[20] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4), 2023. 1

[21] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4):1–14, 2023. 2, 3, 4

[22] Byungsoo Kim, Vinicius C. Azevedo, Markus Gross, and Nils Thuerey. Deep fluids: A generative network for parameterized fluid simulations. *Computer Graphics Forum (Eurographics)*, 38(2):59–70, 2019. 2

[23] G.S. Kirk, J.E. Raven, and M. Schofield. *The Presocratic Philosophers*. Cambridge University Press, 1983. Discussion of Heraclitus' doctrine of flux, commonly paraphrased as "everything flows". 1

[24] Dmitry Kochkov, Amit Maity, Max Zwicker, Nils Thuerey, and Justin Knoll. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(20):e2101784118, 2021. 2

[25] Agelos Kratimenos, Jiahui Lei, and Kostas Daniilidis. Dynmf: Neural motion factorization for real-time dynamic view synthesis with 3d gaussian splatting. In *European Conference on Computer Vision (ECCV) 2024*, 2024. 3

[26] Youngjoong Kwon, Minhyuk Kim, Seungyong Kim, Jeong Joon Park, Jonghyun Choi, and Jaesik Kim. Efficient editable 4d gaussian fields for dynamic scene rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 3

[27] Jungho Lee, Donghyeong Kim, Dogyoon Lee, Suhwan Cho, Minhyeok Lee, Wonjoon Lee, Taeoh Kim, Dongyoon Wee, and Sangyoun Lee. Comogaussian: Continuous motion-aware gaussian splatting from motion-blurred images. *arXiv preprint arXiv:2503.05332*, 2025. 3

[28] Jung-Woo Lee, Jae-Han Kim, and Jaesik Kim. Fully explicit dynamic gaussian splatting for real-time dynamic view synthesis. In *European Conference on Computer Vision (ECCV)*, 2024. 3

[29] Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, et al. Neural 3d video synthesis from multi-view video. In *CVPR*, 2022. 1

[30] Tianye Li, Mira Slavcheva, Michael Zollhöfer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven

Lovegrove, Michael Goesele, Richard Newcombe, and Zhaoyang Lv. Neural 3d video synthesis from multi-view video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5521–5531, 2022. 5

[31] Youtian Lin, Zuozhuo Dai, Siyu Zhu, and Yao Yao. Gaussian-flow: 4d reconstruction with dynamic 3d gaussian particle. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21136–21145, 2024. 3

[32] Youtian Lin, Zuozhuo Dai, Siyu Zhu, and Yao Yao. Gaussian-flow: 4d reconstruction with dynamic 3d gaussian particle. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21136–21145, 2024. 1

[33] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object, 2023. 6

[34] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *3DV*, 2024. 1

[35] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *2024 International Conference on 3D Vision (3DV)*, pages 800–809. IEEE, 2024. 3

[36] 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. *arxiv*, 2020. 1

[37] 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

[38] Makoto Okabe, Yasuyuki Matsushita, and Takeo Igarashi. Fluid volume reconstruction from multi-view video. In *IEEE International Conference on Computer Vision (ICCV)*, 2015. 2

[39] 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 *CVPR*, 2021. 1, 2, 5

[40] 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. In *CVPR*, pages 8151–8161, 2021. 2

[41] 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. *arXiv*, 2021. 1

[42] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *CVPR*, 2021. 1

[43] Adrià Pumarola, Enric Corona, Gerard Pons-Moll, Javier Romero, and Francesc Moreno-Noguer. D-NeRF: Neural radiance fields for dynamic scenes. In *CVPR*, pages 10318–10327, 2021. 5, 7

[44] Sara Fridovich-Keil and Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In *CVPR*, 2023. 1, 4, 5, 7

[45] Connor Schenck and Dieter Fox. Spnets: Differentiable fluid dynamics for deep neural networks. In *arXiv preprint arXiv:1806.06094*, 2018. 2

[46] Jos Stam. Stable fluids. In *Proceedings of SIGGRAPH '99*, pages 121–128, 1999. 2

[47] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. In *Computer Graphics (Proceedings of SIGGRAPH '87)*, pages 205–214, 1987. 2

[48] Stefan Wiewel, Byungsoo Kim, and Nils Thuerey. Latent space physics: Towards learning the temporal evolution of fluid simulations. In *Computer Graphics Forum (Eurographics)*, pages 71–82, 2019. 2

[49] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. arXiv preprint arXiv:2310.08528. 1, 3, 4, 5, 7

[50] Chengyang Xie, Qiang Liu, Xinyue Zhang, Wenhao Xu, Jianfeng He, and Yifan Liu. Splatflow: Self-supervised scene flow estimation with 3d gaussian splatting. *arXiv preprint arXiv:2410.12345*, 2024. 3

[51] Tianyi Xie, Zeshun Zong, Yuxing Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. Physgaussian: Physics-integrated 3d gaussians for generative dynamics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4389–4398, 2024. 3

[52] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. *arXiv preprint arXiv:2309.13101*, 2023. 1, 3, 5, 7

[53] David Junhao Zhang, Roni Paiss, Shiran Zada, Nikhil Karnad, David E. Jacobs, Yael Pritch, Inbar Mosseri, Mike Zheng Shou, Neal Wadhwa, and Nataniel Ruiz. Recapture: Generative video camera controls for user-provided videos using masked video fine-tuning, 2024. 8

[54] H. Zhang, X. Liu, Y. Gao, Y. Wang, and B. Zhu. Fluidnexus: Neural video-based fluid reconstruction and prediction. *arXiv preprint arXiv:2404.01563*, 2024. 2

[55] 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

[56] Xinjie Zhang, Zhening Liu, Yifan Zhang, Xingtong Ge, Dailan He, Tongda Xu, Yan Wang, Zehong Lin, Shuicheng Yan, and Jun Zhang. Mega: Memory-efficient 4d gaussian splatting for dynamic scenes. arXiv preprint arXiv:2410.13613, 2024. 10.48550/arXiv.2410.13613. 3

[57] Ruijie Zhu, Yanzhe Liang, Hanzhi Chang, Jiacheng Deng, Jiahao Lu, Wenfei Yang, Tianzhu Zhang, and Yongdong Zhang. Motiongs: Exploring explicit motion guidance for deformable 3d gaussian splatting. In *Advances in Neural Information Processing Systems (NeurIPS) 2024*, 2024. 3