

FreeGave: 3D Physics Learning from Dynamic Videos by Gaussian Velocity

Jinxi Li, Ziyang Song, Siyuan Zhou, Bo Yang*

vLAR Group, The Hong Kong Polytechnic University

{jinxi.li, ziyang.song, siyuan.zhou}@connect.polyu.hk, bo.yang@polyu.edu.hk



Figure 1. Given video frames of a real-world dynamic scene, our FreeGave can learn the underlying physics and accurately predict the future motion of a purple pen going through the hole of a tape, while existing baselines cannot.

Abstract

*In this paper, we aim to model 3D scene geometry, appearance, and the underlying physics purely from multi-view videos. By applying various governing PDEs as PINN losses or incorporating physics simulation into neural networks, existing works often fail to learn complex physical motions at boundaries or require object priors such as masks or types. In this paper, we propose **FreeGave** to learn physics of complex dynamic 3D scenes without needing any object priors. The key to our approach is to introduce a physics code followed by a carefully designed divergence-free module for estimating a per-Gaussian velocity field, without relying on the inefficient PINN losses. Extensive experiments on three public datasets and a newly collected challenging real-world dataset demonstrate the superior performance of our method for future frame extrapolation and motion segmentation. Most notably, our investigation into the learned physics codes reveals that they truly learn meaningful 3D physical motion patterns in the absence of any human labels in training. Our code and data are available at <https://github.com/vLAR-group/FreeGave>*

1. Introduction

Obtaining an accurate physical model for complex dynamic 3D environments is a key enabler for many emerging applications, including robot planning, embodied interaction, and dynamic 3D/4D generation. Represented by NeRF [47], 3DGS [28] and their variants [5, 29, 33, 51, 68, 75], recent advancements in 3D learning have demonstrated an unprecedented level of fidelity in modeling dynamic 3D

scenes, achieving excellent performance in novel view interpolation and/or 3D shape reconstruction. However, the majority of these approaches lack the core ability of future prediction, essentially because they fail to effectively learn physical properties of 3D space.

To learn physical properties, one line of works introduces various governing PDEs, commonly known as PINN [53], into the learning process of dynamic 3D scenes. While achieving encouraging results in modeling 3D physics such as velocity and viscosity [13, 30, 31, 66], the integration of PINN losses often compromises the accuracy of learned physics in boundary regions, leading to blurred predictions for future frames. Another line of works explicitly incorporates various physics models into neural networks [6, 24, 67, 83], showing impressive results in learning physical properties and further transferring to simulation. Nevertheless, these methods are usually limited to specific types of objects or motions, as the encoded physics models are often less generic than PINN constraints, necessitating additional boundary conditions such as object masks.

In this work, we aim to present a simple and general method for learning physics of complex dynamic 3D scenes using only multi-view videos, without needing any PINN losses or extra priors such as object types or masks, which are often needed by existing methods. Among various physical properties of a dynamic 3D scene, we follow the recent work NVFi [30] and focus on learning 3D velocity as it directly governs 3D dynamics and future predictions.

Given video frames containing an unknown number of objects/parts exhibiting diverse motions, at a single glance, it is extremely challenging to learn velocities of all (un-

known) objects or parts in 3D space, due to the lack of physics and semantics in raw pixels. Nevertheless, the whole dynamic 3D scene can be simply regarded as a collection of many 3D rigid particles with their own sizes, orientations, and motion patterns. The dynamics of each particle are actually governed by its hidden physical parameters, such as mass and forces. Thus, our challenge of learning complex 3D scene dynamics can be naturally framed as learning dynamics of these 3D particles in general.

With this motivation, we propose an elegant pipeline to simultaneously learn 3D geometry, appearance, and velocity of dynamic scenes. Treating each 3D Gaussian kernel as a rigid particle, we extend the successful 3DGS [28] to learn an additional physics code for each 3D Gaussian. With such a code, the 3D velocity of each Gaussian is then learned from visual pixels without relying on PINN losses. Particularly, our pipeline comprises three major components: 1) a **canonical 3D representation module** which adopts a vanilla 3DGS to learn the dynamic 3D scene geometry and appearance at a canonical timestamp; 2) a **neural divergence-free Gaussian velocity module** to estimate per-Gaussian velocity based on the introduced latent physics code; and 3) a **deformation-aided optimization module** to drive the velocity to be physically meaningful.

The core to our pipeline lies in the latter two modules. Basically, given video frames of a dynamic 3D scene, our neural divergence-free Gaussian velocity module together with the optimization strategy explicitly drives the per-Gaussian physics code to describe the corresponding motion patterns in latent space. This ensures that the estimated velocities adhere to fundamental physics rules, ultimately enabling precise predictions of future frames. Figure 1 shows qualitative results of a challenging dynamic scene. Our method is named **FreeGave** and our contributions are:

- We introduce a general framework for learning physics of dynamic 3D scenes solely from RGB videos, eliminating the need for any object priors, such as types and masks.
- We propose to learn per-Gaussian velocities from latent physics codes, guided by a carefully designed divergence-free ingredient instead of inefficient PINN losses.
- We showcase superior results in future frame extrapolation on three existing datasets, and a newly collected real-world dataset featuring highly challenging dynamics.

2. Related Works

3D Representation Learning: Classic methods for representing 3D geometry mainly include point clouds [14], meshes [26], voxels [73], and primitives [84]. However, the fidelity of 3D shapes is often limited by the spatial resolution inherent in these explicit representations. To address this limitation, a line of neural 3D representations has been introduced recently, including un/signed distance

fields [12, 49, 65], occupancy fields [10, 45], and radiance fields [47]. Leveraging coordinate-based MLPs, these representations have demonstrated remarkable performance in novel view rendering and 3D shape recovery [60, 63, 71]. Nonetheless, the integration of coordinate-based representations leads to slow extraction of 2D views and 3D surfaces. To resolve this issue, 3DGS [28] is introduced to represent a 3D shape as a set of 3D Gaussian kernels with various properties, achieving real-time rendering speeds as well as exceptional quality in novel view synthesis [4, 8, 16]. In our pipeline, we adopt 3DGS as our 3D shape and appearance representation, treating each Gaussian kernel as a rigid particle for learning its physics code and velocity.

Dynamic 3D Reconstruction: To reconstruct dynamic 3D objects and scenes, existing works are primarily built on established static 3D representations such as SDF [49], NeRF [47], and 3DGS [28]. Given dynamic visual input over time, current methods [7, 15, 18, 19, 23, 38, 41, 50, 59, 61, 62, 74, 77, 81] commonly add the time dimension into a static 3D representation to learn the underlying motion for rigid or deformable targets. While obtaining impressive accuracy in novel view rendering, especially when utilizing 3DGS as the 3D representation [2, 27, 34, 36, 42, 46, 52, 56, 68, 76], these methods can only synthesize high-quality views within the observed time, lacking the ability to predict physically meaningful future frames. This limitation arises because these methods fail to encode physics priors into the learning process, but just fit the visual observations during training. In our pipeline, we explicitly learn a physics code for each Gaussian via our Gaussian velocity module and the divergence-free parameterization.

Video Prediction: Some studies focus on directly predicting future video frames [17, 20]. Thanks to the powerful diffusion model and Transformer-based architectures, many of these methods [9, 21, 22, 39, 48, 64, 69, 80] have demonstrated superior performance in video generation. However, they are prone to predicting physically implausible motions, primarily because their networks do not explicitly incorporate physics priors in training, as discussed in [25, 40].

3D Physics Learning: Inspired by recent physics-informed neural networks (PINN) [53], a series of subsequent methods [3, 23, 54, 66] learn various physical properties of 3D objects and scenes by converting different types of PDEs into loss functions as soft constraints. While achieving encouraging results in physics learning and future prediction, they often struggle to model complex physics in boundary regions. Another class of methods directly incorporates physics systems [24, 37, 44, 72, 82, 83] such as graphs and springs into the network to learn the dynamics of specific targets including fluids and elastic objects. In our pipeline, we directly encode the generic divergence-free constraint into the learning process, thereby avoiding the use of PINN losses to train neural networks.

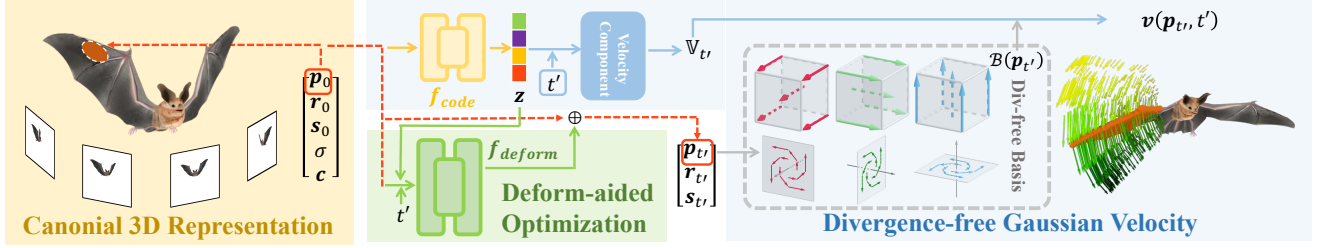


Figure 2. The leftmost block shows the canonical 3D Gaussians module, the middle bottom block shows the deformation-aided optimization module and the right block shows our divergence-free Gaussian velocity module.

3. FreeGave

Given dynamic multi-view RGB videos with known camera poses and intrinsics over time, all frames are denoted as $\{\{I_0^1 \cdots I_0^n \cdots I_0^N\} \cdots \{I_T^1 \cdots I_T^n \cdots I_T^N\}\}$, where N represents the total number of cameras and T the greatest timestamp in training. Our pipeline is designed to simultaneously learn 3D geometry, appearance, and velocity information of the captured dynamic scene. As shown in Figure 2, the first component - canonical 3D representation module - aims to learn a set of 3D Gaussian kernels to represent the 3D scene geometry and appearance in a canonical space, which is elaborated in Section 3.1. The second component - neural divergence-free Gaussian velocity module - attempts to learn the 3D velocity of each Gaussian kernel, which is detailed in Section 3.2, while the third module - deformation-aided optimization - is discussed in Section 3.3.

3.1. Canonical 3D Representation Module

Following the vanilla 3DGS [28], this module aims to learn a set of 3D Gaussian kernels P_0 to represent the canonical scene geometry and appearance at time $t = 0$. Each kernel is parameterized by a 3D position p_0 , scaling s_0 , covariance matrix converted from quaternion r_0 , opacity σ , and color c converted from spherical harmonics (SH). Following prior works of dynamic Gaussian [68, 75], we assume the opacity σ and color c of each Gaussian are not changed over time, but constantly associated and transported with the kernel.

We exactly follow 3DGS to train canonical kernels. Having the N images at time $t = 0$, we first initialize all canonical kernels P_0 randomly or based on sparse point clouds created by SfM [55], and then project kernels into camera space, rendering 2D images and optimizing Gaussian parameters via ℓ_1 and ℓ_{ssim} losses as used in 3DGS:

$$\underbrace{\{\cdots (p_0, r_0, s_0, \sigma, c) \cdots\}}_{P_0} \xrightarrow[\ell_1 + \ell_{ssim}]{\text{render}} \{I_0^1 \cdots I_0^n \cdots I_0^N\} \quad (1)$$

3.2. Divergence-free Gaussian Velocity Module

This module aims to model the velocity of each Gaussian kernel. A naïve solution is to implement a simple net-

work similar to the prior work NVFi [30] to take each Gaussian as input and directly regress the corresponding 3D velocity, followed by commonly-used PINN losses to drive the estimated velocity to be physically meaningful, *i.e.*, divergence-free. However, there are two key limitations. First, it is extremely hard to learn distinct 3D velocities for neighboring Gaussians exhibiting radically different motions, as a neural network just models a continuous and smooth function. Second, PINN losses are inefficient in learning physics priors as they involve dense sampling in both spatial and temporal dimensions during training. To this end, we tackle the problem of Gaussian velocity learning by introducing the following new technique ingredients.

Physics Code: For each canonical Gaussian, we propose to learn a vector z , which is regarded as a latent physics code to generally describe the underlying physical information possibly including mass, forces, *etc.*. Basically, this code aims to explain the motion type of a Gaussian kernel over a whole period of time. Therefore, the code z is shared across all timestamps. In particular, we feed the position of each canonical Gaussian into an MLP-based network f_{code} , directly predicting its physics code z :

$$z = f_{code}(p_0), \quad z \in \mathcal{R}^L \quad (2)$$

Technically, we may assign a learnable code to each canonical Gaussian independently. Nevertheless, such a strategy, albeit flexible, involves too many parameters to learn and we empirically find it is inferior as verified in our ablation study. Now, all canonical Gaussians P_0 are as follows:

$$P_0 \leftarrow \{\cdots (p_0, r_0, s_0, \sigma, c, z) \cdots\} \quad (3)$$

Divergence-free Velocity: Having the physics code of a Gaussian kernel, our goal is to estimate its velocity field over the whole time period, without relying on the inefficient PINN losses. However, the estimated velocity field for each Gaussian must satisfy the basic divergence-free property. Otherwise, the extrapolated future frames will not be physically meaningful.

To this end, we treat each canonical Gaussian p_0 as a rigid particle with its size and orientation, whose future motion is governed by a divergence-free velocity field $v(p_t, t)$, where p_t indicates the Gaussian position at time t . To learn the divergence-free velocity field $v(p_t, t)$ for a rigid parti-

cle, we propose to disentangle $\mathbf{v}(\mathbf{p}_t, t)$ into six basic velocity components: $\mathbb{V}_t \leftarrow [v_t^x, v_t^y, v_t^z, w_t^x, w_t^y, w_t^z]$ and a basis matrix $\mathcal{B}(\mathbf{p}_t)$, where $v_t^x/v_t^y/v_t^z$ represent the linear velocities along $x/y/z$ axes respectively, and $w_t^x/w_t^y/w_t^z$ represent the angular velocities of the rigid particle around $z/y/x$ axes respectively. Mathematically, the velocity field $\mathbf{v}(\mathbf{p}_t, t)$ is decomposed as follows:

$$\mathbf{v}(\mathbf{p}_t, t) = \mathbb{V}_t \cdot \mathcal{B}(\mathbf{p}_t), \quad \mathbb{V}_t \in \mathcal{R}^{1 \times 6} \quad \mathcal{B}(\mathbf{p}_t) \in \mathcal{R}^{6 \times 3} \quad (4)$$

where $\mathcal{B}(\mathbf{p}_t)$ is a set of basis vectors defined as follows:

$$\mathcal{B}(\mathbf{p}_t) = \begin{bmatrix} 1 & 0 & 0 & -p_t^y & p_t^z & 0 \\ 0 & 1 & 0 & p_t^x & 0 & -p_t^z \\ 0 & 0 & 1 & 0 & -p_t^x & p_t^y \end{bmatrix}^T \quad (5)$$

where $p_t^x/p_t^y/p_t^z$ represent the values of $\mathbf{p}_t$ at $x/y/z$ axes. The left three columns correspond to the linear velocities $v_t^x/v_t^y/v_t^z$, whereas the right three columns correspond to the angular velocities $w_t^x/w_t^y/w_t^z$. Note that, to guarantee the divergence-free property of $\mathbf{v}(\mathbf{p}_t, t)$, we will design $\mathbb{V}_t$ to be irrelevant to $\mathbf{p}_t$.

For any specific rigid particle at time t , we can easily collect its canonical position $\mathbf{p}_0$ and physics code $\mathbf{z} = f_{code}(\mathbf{p}_0)$. To learn $\mathbb{V}_t$, a naïve solution is to feed $\mathbf{z}$ and t into an MLP-based network, directly decoding a 6-dimensional vector. Nevertheless, we empirically find that this is too flexible and results in inferior performance as shown in our ablation study. To tackle this issue, we opt for a simple design illustrated in Figure 3.

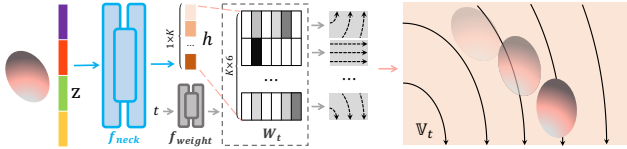


Figure 3. An illustration of the network architecture to learn basic velocity component $\mathbb{V}_t$.

In particular, we first feed the physics code $\mathbf{z}$ into several MLP layers f_{neck} , obtaining a bottleneck vector $\mathbf{h} \in \mathcal{R}^K$, where K is a relatively small hyperparameter. In parallel, we feed the timestamp t into another several MLP layers f_{weight} , obtaining a $K \times 6$ vector. This is followed by a reshape operation and we get a matrix $\mathbf{W}_t \in \mathcal{R}^{K \times 6}$. The basic velocity component $\mathbb{V}_t$ is obtained by a matrix multiplication as follows:

$$\mathbb{V}_t = \mathbf{h} \cdot \mathbf{W}_t = f_{neck}(\mathbf{z}) \cdot f_{weight}(t) \quad (6)$$

Basically, our insight behind this simple design is that: the physics code $\mathbf{z}$ tends to be decoded into K types of motion patterns by f_{neck} , whereas f_{weight} tends to generate a set of weights to select the decoded motion patterns at a particular timestamp t . Empirically, such a simple architecture achieves superior results and we leave more sophisticated designs for future exploration. All implementation details of the physics code, the velocity module, and the proof of divergence-free property are in Appendix A.

3.3. Deformation-aided Optimization

Regarding each Gaussian kernel, we need to effectively optimize its physics code $\mathbf{z}$ and basic velocity components $\mathbb{V}_t$, such that the corresponding per-Gaussian velocity field $\mathbf{v}(\mathbf{p}_t, t)$ is physically meaningful. However, this is particularly challenging because there are only multi-view images at sparse timestamps as supervision without any other priors. We empirically find that directly training all networks is hard to converge as shown in our ablation study.

In this work, we introduce an auxiliary deformation field to aid the optimization process of our pipeline. In particular, we feed each canonical Gaussian kernel $\mathbf{p}_0$, timestamp t , and its physics code $\mathbf{z}$ into an MLP-based network f_{deform} , predicting the corresponding position displacement $\delta\mathbf{p}$, and the change of quaternion $\delta\mathbf{r}$ and scaling $\delta\mathbf{s}$ from timestamp 0 to t . Naturally, all Gaussians $\mathbf{P}_t$ at time t can be computed by transporting all canonical Gaussians as follows. The deformation operations $\circ$ and $\odot$ follow [75].

$$(\delta\mathbf{p}, \delta\mathbf{r}, \delta\mathbf{s}) = f_{deform}(\mathbf{p}_0, t, \mathbf{z}), \quad \mathbf{z} = f_{code}(\mathbf{p}_0) \quad (7)$$

$$\mathbf{p}_t = \mathbf{p}_0 + \delta\mathbf{p}, \quad \mathbf{r}_t = \mathbf{r}_0 \circ \delta\mathbf{r}, \quad \mathbf{s}_t = \mathbf{s}_0 \odot \delta\mathbf{s}$$

$$\mathbf{P}_t \leftarrow \{\cdots (\mathbf{p}_t, \mathbf{r}_t, \mathbf{s}_t, \sigma, \mathbf{c}, \mathbf{z}) \cdots\}$$

With our design of physics code $\mathbf{z}$, basic velocity components $\mathbb{V}_t$, and the deformation field f_{deform} , we now discuss how to link and train them together, ultimately driving the learned per-Gaussian velocity field to be truly learned.

First, given training images $\{\{I_0^1 \cdot I_0^N\} \cdot \{I_T^1 \cdot I_T^N\}\}$ and the canonical Gaussians $\mathbf{P}_0$, we sample a timestamp t from training set and another timestamp t' , where $\Delta t = t - t'$ is predefined to be small enough. We can obtain Gaussians $\mathbf{P}_{t'}$ via the auxiliary deformation field f_{deform} (Eq 7):

$$\mathbf{P}_{t'} \leftarrow \{\cdots (\mathbf{p}_{t'}, \mathbf{r}_{t'}, \mathbf{s}_{t'}, \sigma, \mathbf{c}, \mathbf{z}) \cdots\} \quad (8)$$

Second, we feed the set of Gaussians $\mathbf{P}_{t'}$ at time t' and the sampled interval Δt into our interleaved mid-point Algorithm 1, predicting the set of Gaussians $\mathbf{P}_t$ at time t transported by our per-Gaussian velocity field.

Algorithm 1 Interleaved mid-point method.

Input: (1) Gaussians at t' : $\{\cdots (\mathbf{p}_{t'}, \mathbf{r}_{t'}, \mathbf{s}_{t'}, \sigma, \mathbf{c}, \mathbf{z}) \cdots\}$, and (2) Δt
Output: Gaussians at t , i.e., at $(t' + \Delta t)$: $\{\cdots (\mathbf{p}_t, \mathbf{r}_t, \mathbf{s}_t, \sigma, \mathbf{c}, \mathbf{z}) \cdots\}$

- $\mathbf{v}(\mathbf{p}_{t'}, t) \leftarrow f_{neck}(\mathbf{z}) \cdot f_{weight}(t') \cdot \mathcal{B}(\mathbf{p}_{t'})$, (ref to Eqs 4/5/6)
- $\mathbf{p}_{mid} \leftarrow \mathbf{p}_{t'} + \frac{\Delta t}{2} \mathbf{v}(\mathbf{p}_{t'}, t)$
- $\mathbf{v}_{mid} \leftarrow f_{neck}(\mathbf{z}) \cdot f_{weight}(t' + \frac{\Delta t}{2}) \cdot \mathcal{B}(\mathbf{p}_{mid})$
- $\mathbf{p}_t \leftarrow \mathbf{p}_{t'} + \Delta t \mathbf{v}_{mid}$
- Convert quaternion to rotation: $\mathbf{R}_{t'} \leftarrow \mathbf{r}_{t'}$
- $\mathbf{R}_t \leftarrow (\mathbf{I} + \Delta t \frac{\partial \mathbf{v}_{mid}}{\partial \mathbf{p}_{mid}}) \mathbf{R}_{t'}$, (following the approximation in [70])
- Convert rotation to quaternion: $\mathbf{r}_t \leftarrow \mathbf{R}_t$
- Assign $\mathbf{s}_{t'}$ to $\mathbf{s}_t$: $\mathbf{s}_t \leftarrow \mathbf{s}_{t'}$, (each Gaussian is a rigid particle with its size fixed over time, ensuring the divergence-free property)

Return: All gaussians at t : $\{\cdots (\mathbf{p}_t, \mathbf{r}_t, \mathbf{s}_t, \sigma, \mathbf{c}, \mathbf{z}) \cdots\}$

Third, having the set of Gaussians $\mathbf{P}_t$ transported from t' by our per-Gaussian velocity field, we project them to 2D space, rendering RGBs supervised by the total N training views $\{I_t^1 \cdot I_t^N\}$ at time t via the standard 3DGS losses:

$$\{\mathbf{P}_0, f_{code}, f_{neck}, f_{weight}, f_{deform}\} \leftarrow (\ell_1 + \ell_{ssim}) \quad (9)$$

Implementation and training details are in Appendix B & C.

4. Experiments

Datasets: Our method focuses on learning physical information from dynamic videos for accurate future frame predictions, rather than fitting training views for interpolation. In this regard, the closest work to us is NVFi [30], and we mainly evaluate our method on its two synthetic dynamic datasets: **1) Dynamic Object Dataset** which consists of 6 dynamic objects undergoing different rigid or deformable motions; and **2) Dynamic Indoor Scene Dataset** which comprises 4 complex indoor scenes with 3 to 6 objects exhibiting challenging motions. For the sake of motion diversity, we also evaluate another synthetic **3) ParticleNeRF Dataset** [1], which includes various motions from robot manipulation to harmonic oscillation. All datasets have test splits for *novel view interpolation* and *future frame extrapolation*. More details of datasets are in Appendix D.

While NVFi [30] is evaluated on a real-world dataset, it just has two simple scenes selected from NVIDIA Dynamic Scene Dataset [79] which is not primarily collected for physics learning. Since this field of study is still in its infancy and there is lack of real-world datasets in high-quality, we opt to collect a new real-world dataset particularly for benchmarking 3D physics learning, which is the first of its kind to the best of our knowledge and will be made public.

4) FreeGave-GoPro Dataset: We capture 6 dynamic scenes with 20 GoPro cameras (GoPro Hero 10 Black). All cameras are rigged on tripods and placed around an operation platform, as illustrated in Figure 4. The 6 dynamic scenes consist of particularly challenging physical motions such as a moving object is tightly beside a static object, or a thin object is moving through a tiny hole of another object.

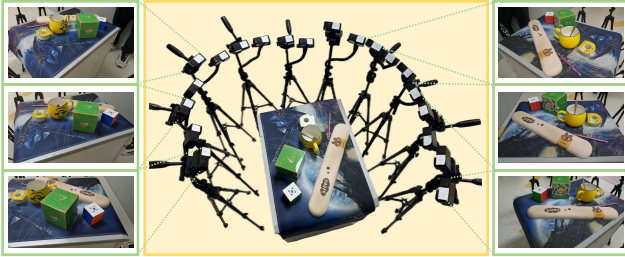


Figure 4. The camera settings for our dataset collection.

We use voice to synchronize all videos and use COLMAP [55] on the first frames of all 20 views to estimate camera poses. Since all cameras are fixed under a stabilizing mode, the camera poses do not change over time. For each dynamic scene, we select 89 frames from each view, and resize images to be a resolution of 960×540 . We reserve the first 67 frames at 17 picked viewing angles as the training split, *i.e.*, 1139 frames, while leaving the

67 frames at the remaining 3 viewing angles for evaluating *novel view interpolation* within the training time period, *i.e.*, 201 frames. We keep the last 22 frames at all 20 viewing angles for evaluating *future frame extrapolation*, *i.e.*, 440 frames in total. More details are in Appendix D.

Baselines: We choose the following baselines: **1) NVFi** [30]. It is the closest work to us, but relies on PINN losses to learn physics priors whereas we directly build a divergence-free velocity field for each rigid particle. **2) T-NeRF** [51], **3) D-NeRF** [51], **4) NSFF** [32], and **5) TiNeuVox** [15]. All these baselines adopt NeRF framework as the backbone, thus being short in 3D reconstruction compared to 3DGS. As our method adopts 3DGS, for a fair comparison, we also include the following two baselines. **6) DefGS** [75]. This recent method Deformable 3D Gaussians is excellent to model dynamic 3D scenes for novel view interpolation, but lacks the ability for future prediction. In this regard, we build the most comparable baseline to us **7) DefGS_{nvfi}** by combining DefGS with the velocity field proposed by NVFi. It shows strong ability in novel view synthesis (inherited from DefGS) and extrapolation ability (inherited from NVFi framework). To be fair, we combine the original deformation field of DefGS with exactly the same velocity field of NVFi. The same deformation-aided optimization as our FreeGave is used to train DefGS_{nvfi}.

Metrics: The standard metrics **PSNR**, **SSIM**, and **LPIPS** are reported for RGB view synthesis in two tasks: novel view interpolation and future frame extrapolation.

4.1. Evaluation for Future Frame Extrapolation

We evaluate our method on all four datasets for both novel view interpolation and future frame extrapolation. All baselines are evaluated on the Dynamic Object Dataset and Dynamic Indoor Scene Dataset. We select 4 representative baselines to compare on our new FreeGave-GoPro Dataset, including **TiNeuVox**, **NVFi**, **DefGS** and **DefGS_{nvfi}**.

Results & Analysis: From Tables 1&2, it can be seen that: 1) Our FreeGave achieves the best performance on all datasets, clearly surpassing all baselines by a large margin on the challenging task of future frame extrapolation, showing that our method can indeed learn the underlying physical information, thus predicting accurate future motions. 2) Our method also achieves better scores on the task of novel view interpolation, notably outperforming the strong baselines DefGS and DefGS_{nvfi}. We hypothesize that the effective incorporation of physics priors allows our method to fully utilize all Gaussian kernels for representing 3D geometry and appearance. As shown in Figure 6, our method makes promising predictions, especially in dynamic multi-object scenes, such as the pen-through-tape scene, where only our method succeeds while others all fail. More quantitative and qualitative results are in Appendix K to O.

Table 1. Quantitative results of all methods for both novel view interpolation and future frame extrapolation on Dynamic Object Dataset and Dynamic Indoor Scene Dataset, and future frame extrapolation on ParticleNeRF Dataset. Bold numbers indicate the best performance.

	Dynamic Object Dataset						Dynamic Indoor Scene Dataset						ParticleNeRF Dataset		
	Interpolation			Extrapolation			Interpolation			Extrapolation			Extrapolation		
	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$
T-NeRF [51]	13.163	0.709	0.353	13.818	0.739	0.324	24.944	0.742	0.336	22.242	0.700	0.363	-	-	-
D-NeRF [51]	14.158	0.697	0.352	14.660	0.737	0.312	25.380	0.766	0.300	20.791	0.692	0.349	-	-	-
T-NeRF _{PINN}	15.286	0.794	0.293	16.189	0.835	0.230	16.250	0.441	0.638	17.290	0.477	0.618	-	-	-
HexPlane _{PINN}	27.042	0.958	0.057	21.419	0.946	0.067	25.215	0.763	0.389	23.091	0.742	0.401	-	-	-
NSFF [32]	-	-	-	-	-	-	29.365	0.829	0.278	24.163	0.795	0.289	-	-	-
TiNeuVox [15]	27.988	0.960	0.063	19.612	0.940	0.073	29.982	0.864	0.213	21.029	0.770	0.281	20.407	0.888	0.102
NVFi [30]	29.027	0.970	0.039	27.594	0.972	0.036	30.675	0.877	0.211	29.745	0.876	0.204	18.173	0.867	0.119
DefGS [75]	<u>37.865</u>	<u>0.994</u>	<u>0.007</u>	19.849	0.949	0.045	29.926	<u>0.916</u>	<u>0.130</u>	21.380	0.819	0.188	19.205	0.900	0.083
DefGS _{nvfi}	37.316	0.994	0.008	28.749	0.984	0.013	30.170	0.915	0.133	31.096	0.945	0.077	22.730	0.931	0.050
FreeGave (Ours)	39.393	0.995	0.005	31.987	0.990	0.007	32.287	0.930	0.092	35.019	0.966	0.051	26.657	0.956	0.030

Table 2. Quantitative results for both novel view interpolation and future frame extrapolation on FreeGave-GoPro Dataset.

	FreeGave-GoPro Dataset					
	Interpolation			Extrapolation		
	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$
TiNeuVox [15]	19.026	0.740	0.319	20.600	0.760	0.292
NVFi [30]	18.947	0.739	0.322	22.747	0.756	0.393
DefGS [75]	<u>28.408</u>	0.920	0.091	21.344	0.858	0.170
DefGS _{nvfi}	28.217	0.920	<u>0.092</u>	<u>26.431</u>	<u>0.903</u>	<u>0.124</u>
FreeGave (Ours)	28.451	0.920	0.091	28.094	0.914	0.112

4.2. Analysis of Physics Code and Motion Patterns

Our divergence-free Gaussian velocity module is designed to learn the velocity field for each Gaussian particle. With our design of bottleneck vector decomposition, our model can naturally group Gaussian kernels according to the learned motion patterns. To validate this nice property of bottleneck vector, we conduct the following experiments.

For each Gaussian, we query its learned velocity by uniformly sampling 10 timestamps to form a velocity trajectory $\{v_0, \dots, v_{10}\}$. Then, we apply K-means [43] to group the learned bottleneck vectors $h = f_{neck}(z)$ into C groups. In each group, we sample a number of Gaussian kernels and the queried velocity trajectories, followed by rendering to 2D space. We color the velocity trajectory based on the velocity direction. As shown in Figure 5, we can see that Gaussian kernels in the same group share the same velocity trajectory, while Gaussian kernels in different groups have totally different motion trajectories.

We further quantitatively evaluate the objectness after grouping our learned physics codes on Dynamic Indoor Scene dataset. In particular, we follow [78] to render the group masks as 2D object segmentation masks for all 30 views over 60 timestamps on all 4 scenes, *i.e.*, 7200 images in total. We compare our performance with D-NeRF, NVFi, DefGS and DefGS_{nvfi}. We follow NVFi to get the segmentation details of D-NeRF and NVFi. For the 3DGS based baselines, we also adopt OGC [57, 58] to segment Gaussians based on scene flows calculated from their learned deformation fields. Implementation details are in Appendix I. In addition, we include a powerful image-based

Table 3. Quantitative results of scene decomposition on the Synthetic Indoor Scene Dataset.

	AP $\uparrow$	PQ $\uparrow$	F1 $\uparrow$	Pre $\uparrow$	Rec $\uparrow$	mIoU $\uparrow$
Mask2Former [11]	65.37	73.14	78.29	94.83	68.88	64.42
D-NeRF [51]	57.26	46.15	59.02	56.55	62.94	46.58
NVFi [30]	<u>91.21</u>	<u>78.74</u>	<u>93.75</u>	<u>93.76</u>	<u>93.74</u>	<u>67.64</u>
DefGS [75]	51.73	57.60	66.43	63.21	70.07	54.46
DefGS _{nvfi}	55.26	62.75	69.83	69.39	72.91	56.82
FreeGave (Ours)	99.75	96.27	99.99	99.99	99.93	80.52

object segmentation method, Mask2Former [11] pre-trained on COCO [35] dataset, as a fully-supervised baseline.

As shown in Table 3 and Figure 6, we can see that our method achieves nearly perfect object segmentation results on all metrics, clearly outperforming all baselines. This shows that our learned physics codes truly represent object physical motion patterns which can be easily used to identify individual objects accordingly without needing any human labels to train a separate segmentation network.

4.3. Ablation Study

Our framework comprises 3DGS as the scene representation backbone and the neural divergence-free Gaussian velocity module with a deformation-aided optimization module. We conduct the following groups of ablation experiments to validate the choices of our method. All ablation experiments are conducted on the Dynamic Objects dataset.

(1) **Replacing f_{code} by a learnable code z in Eq 2:** Instead of feeding the position of each canonical Gaussian to f_{code} to learn its physics code z , we assign a learnable $z \in \mathcal{R}^L$ for each Gaussian respectively.

(2) **Removing divergence-free parametrization:** Instead of decomposing the velocity field by divergence-free parametrization in Eqs 4/5/6, we directly define the velocity field as $v(p_t, t) = f_{neck}(z) \cdot f_{motion}(p_t, t)$, where $f_{motion} : \mathcal{R}^4 \mapsto \mathcal{R}^K$ is an MLP-based network. Notably, now f_{motion} is based on p_t , so it is not divergence-free.

(3) **Removing bottleneck vector decomposition for velocity components in Eq 6:** Instead of decomposing velocity components network $\mathbb{V}_t$ by bottleneck vector in Eq 6, we directly define $\mathbb{V}_t$ as a single MLP: $\mathbb{V}_t = \text{MLP}(z, t)$.

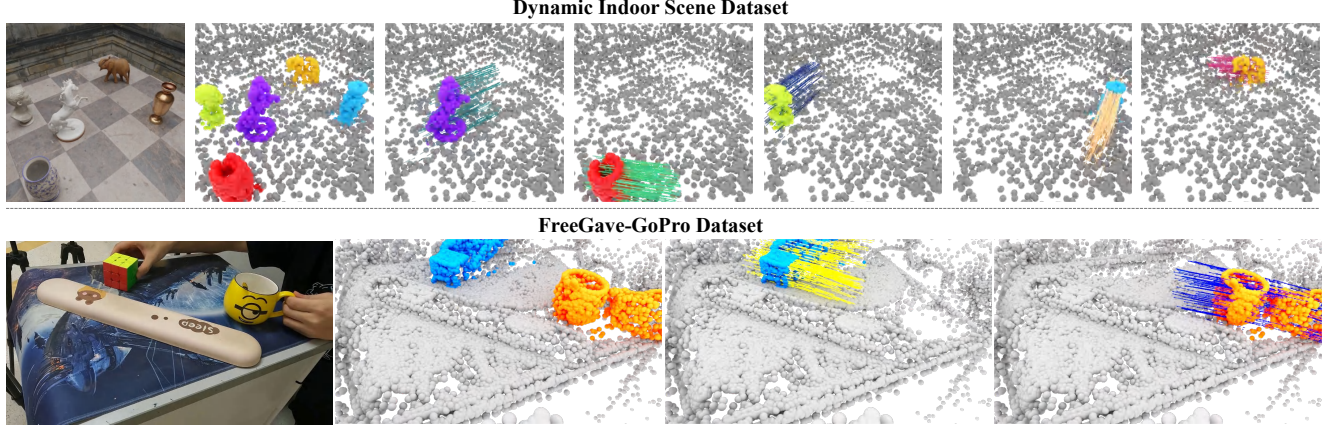


Figure 5. Qualitative results of decomposed motion patterns (colored objects) and their corresponding velocity fields (colored trajectories).

Table 4. Quantitative results of ablation studies on Dynamic Object Dataset.

	Code z	f_{deform}	$v(p_t, t)$	K	Extrapolation		
					PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$
(1)	learnable	full	full	16	25.961	0.975	0.025
(2)	field	full	w/o $B(p_t)$	16	29.400	0.986	0.010
(3)	field	full	w/o decomp	16	29.432	0.985	<u>0.009</u>
(4)	field	full	full	8	30.972	<u>0.989</u>	<u>0.009</u>
(4)	field	full	full	32	31.438	0.990	0.007
(5)	field	$\times$	full	16	17.927	0.922	0.088
(6)	field	w/o z	full	16	31.217	0.988	<u>0.009</u>
(7)	field	w/o δs	full	16	31.704	0.990	0.007
FreeGave	field	full	full	16	31.987	0.990	0.007

(4) **Different choices of motion patterns K :** We compare 3 choices for the number of motion patterns in f_{neck} / f_{weight} : $\{8, 16, 32\}$, while $K=16$ in our main experiments.

(5) **Removing the auxiliary deformation field f_{deform} :** To demonstrate the necessity of our deformation-aided optimization scheme, we additionally train our velocity field without the deformation field. Particularly, we use Algorithm 1 to directly transport canonical Gaussians from $t = 0$ to target time t . Since this requires autoregressive integration of all Gaussians by several steps, the memory cost accumulates rapidly, so we choose $\Delta t = \frac{t}{2}$ to control the memory cost same as our main setting.

(6) **Removing the physics code from deformation field f_{deform} :** Instead of feeding the physics code into the deformation field, we compare with the deformation field same as DefGS [75], i.e., $(\delta p, \delta r, \delta s) = f_{deform}(p_0, t)$.

(7) **Removing the scaling deformation from deformation field:** We remove the scaling deformation from the output of f_{deform} , forcing the motion of each Gaussian to be volume-preserving all time, i.e., $(\delta p, \delta r) = f_{deform}(p_0, t, z)$. Gaussian scales are not updated all time.

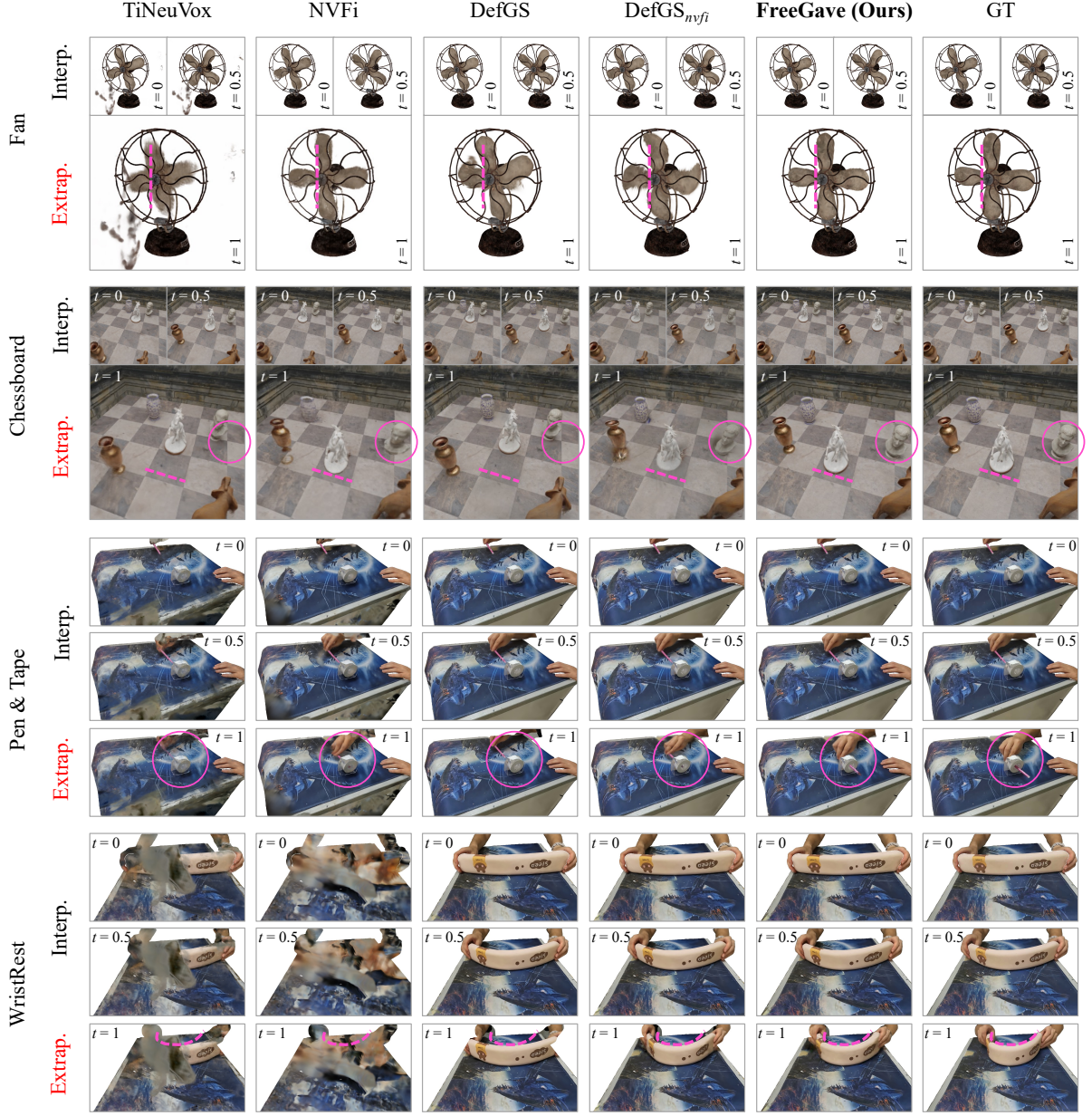
Results & Analysis: From Table 4, we can see that: 1) Removing deformation field is the most influential. Since autoregressive integration is needed in this case, in order to fit the memory cost, we have to use a large integration time interval, as large as half of the target time. This in-

curs instability and noise in training, making the network totally unable to converge. The noise also leads to large gradients for all Gaussian kernels, resulting in unreasonable densification and ultimately making the network totally collapse on Dynamic Indoor Scene Dataset. 2) Replacing the physics code network by assigning a learnable physics code to each Gaussian kernel shows the next greatest impact on performance, demonstrating that the local continuity property of an MLP is helpful for network converging. 3) Either removing divergence-free parametrization or removing motion pattern decomposition diminishes the performance, due to the drastically increase in the degree of freedom and complexity of velocity field. 4) Feeding physics codes as input to the deformation field can enhance the performance. 5) Too few motion patterns decomposition hurts the performance, while over-parametrization is not sensitive. Complete ablation results are in Appendix J.

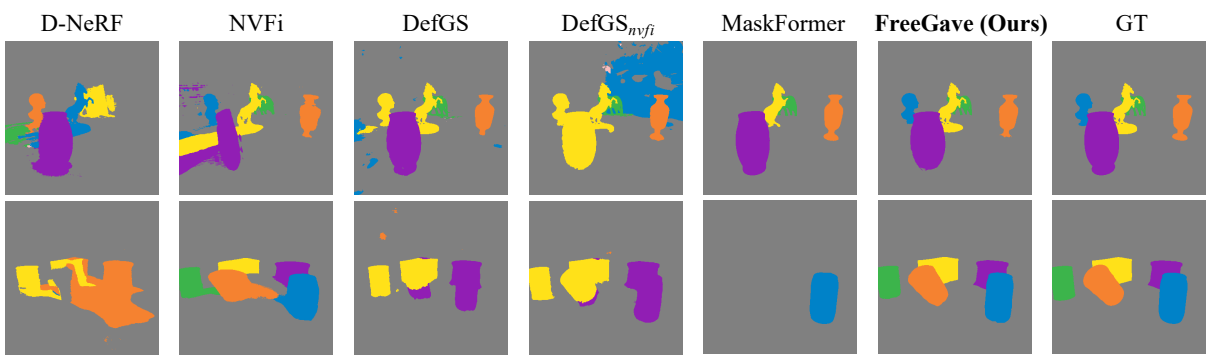
5. Conclusion

Our method extends 3D Gaussian splatting to simultaneously learn 3D scene geometry, appearance, and the critical physics information just from multi-view videos, without needing any object priors. By introducing a physics code and a divergence-free velocity field for each Gaussian kernel, our method can effectively learn precise physical motion patterns without needing the inefficient PINN losses. Extensive experiments on three public datasets and our newly captured challenging real-world dataset demonstrate extraordinary performance of our method for future frame prediction and object motion decomposition. It would be interesting to apply our method in 4D generation where 3D physics learning is crucial to generate plausible motions.

Acknowledgment: This work was supported in part by Research Grants Council of Hong Kong under Grants 25207822 & 15225522.



Future Frame Extrapolation



2D Masks for Motion Segmentation

Figure 6. Qualitative results of our method and baselines for future frame extrapolation and unsupervised motion segmentation.

References

- [1] Jad Abou-Chakra, Feras Dayoub, and Niko Sünderhauf. Particlenet: Particle based encoding for online neural radiance fields. *WACV*, 2024. 5, 2
- [2] Jeongmin Bae, Seoha Kim, Youngsik Yun, Hahyun Lee, Gun Bang, and Youngjung Uh. Per-Gaussian Embedding-Based Deformation for Deformable 3D Gaussian Splatting. *ECCV*, 2024. 2
- [3] Daniele Baieri, Stefano Esposito, Filippo Maggioli, and Emanuele Rodolà. Fluid Dynamics Network: Topology-Agnostic 4D Reconstruction via Fluid Dynamics Priors. *arXiv:2303.09871*, 2023. 2
- [4] Yanqi Bao, Tianyu Ding, Jing Huo, Yaoli Liu, Yuxin Li, Wenbin Li, Yang Gao, and Jiebo Luo. 3D Gaussian Splatting: Survey, Technologies, Challenges, and Opportunities. *arXiv:2407.17418*, 2024. 2
- [5] Jonathan T Barron, Keunhong Park, Steven M Seitz, and Ricardo Martin-brualla. Nerfies: Deformable Neural Radiance Fields. *ICCV*, 2021. 1
- [6] Junhao Cai, Yuji Yang, Weihao Yuan, Yisheng He, Zilong Dong, Liefeng Bo, Hui Cheng, and Qifeng Chen. Gaussian-Informed Continuum for Physical Property Identification and Simulation. *NeurIPS*, 2024. 1
- [7] Ang Cao and Justin Johnson. HexPlane: A Fast Representation for Dynamic Scenes. *CVPR*, 2023. 2
- [8] Guikun Chen and Wenguan Wang. A Survey on 3D Gaussian Splatting. *arXiv:2401.03890*, 2024. 2
- [9] Xinyuan Chen, Yaohui Wang, Lingjun Zhang, Shaobin Zhuang, Xin Ma, Jiashuo Yu, Yali Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. SEINE: Short-to-Long Video Diffusion Model for Generative Transition and Prediction. *ICLR*, 2024. 2
- [10] Zhiqin Chen and Hao Zhang. Learning Implicit Fields for Generative Shape Modeling. *CVPR*, 2019. 2
- [11] Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention Mask Transformer for Universal Image Segmentation. *CVPR*, 2022. 6, 4
- [12] Julian Chibane, Aymen Mir, and Gerard Pons-Moll. Neural Unsigned Distance Fields for Implicit Function Learning. *NeurIPS*, 2020. 2
- [13] Mengyu Chu, Lingjie Liu, Quan Zheng, Erik Franz, Hans Peter Seidel, Christian Theobalt, and Rhaleb Zayer. Physics informed neural fields for smoke reconstruction with sparse data. *TOG*, 2022. 1
- [14] Haoqiang Fan, Hao Su, and Leonidas Guibas. A Point Set Generation Network for 3D Object Reconstruction from a Single Image. *CVPR*, 2017. 2
- [15] Jiemin Fang, Xinggang Wang, and Matthias Nießner. Fast Dynamic Radiance Fields with Time-Aware Neural Voxels. *SIGGRAPH Asia*, 2022. 2, 5, 6, 7, 8
- [16] Ben Fei, Jingyi Xu, Rui Zhang, Qingyuan Zhou, Weidong Yang, and Ying He. 3D Gaussian Splatting as New Era: A Survey. *TVCG*, 2024. 2
- [17] Katerina Fragkiadaki, Pulkit Agrawal, Sergey Levine, and Jitendra Malik. Learning visual predictive models of physics for playing billiards. *ICLR*, 2016. 2
- [18] Sara Fridovich-Keil, Giacomo Meanti, Frederik Warburg, Benjamin Recht, and Angjoo Kanazawa. K-Planes: Explicit Radiance Fields in Space, Time, and Appearance. *CVPR*, 2023. 2
- [19] Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang. Dynamic View Synthesis from Dynamic Monocular Video. *ICCV*, 2021. 2
- [20] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. *ICML*, 2019. 2
- [21] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. Video Diffusion Models. *arXiv:2204.03458*, 2022. 2
- [22] Tobias Höppe, Arash Mehrjou, Stefan Bauer, Didrik Nielsen, and Andrea Dittadi. Diffusion Models for Video Prediction and Infilling. *TMLR*, 2022. 2
- [23] Woobin Im, Geonho Cha, Sebin Lee, Jumin Lee, Juhyeong Seon, Dongyoon Wee, and Sung-Eui Yoon. Regularizing Dynamic Radiance Fields with Kinematic Fields. *ECCV*, 2024. 2
- [24] Alvaro Sanchez-gonzalez Jonathan, Godwin Tobias, Pfaff Rex, Ying Jure, and Peter W Battaglia. Learning to Simulate Complex Physics with Graph Networks. *ICML*, 2020. 1, 2
- [25] Bingyi Kang, Yang Yue, Rui Lu, Zhijie Lin, Yang Zhao, Kaixin Wang, Gao Huang, and Jiashi Feng. How Far is Video Generation from World Model: A Physical Law Perspective. *arXiv:2411.02385*, 2024. 2
- [26] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3D Mesh Renderer. *CVPR*, 2018. 2
- [27] Kai Katsumata, Duc Minh Vo, and Hideki Nakayama. A Compact Dynamic 3D Gaussian Representation for Real-Time Dynamic View Synthesis. *ECCV*, 2024. 2
- [28] Bernhard Kerbl, Université Côte, Georgios Kopanas, Université Côte, Thomas Leimkühler, Max-planck-institut Informatik, and G R Aug. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *SIGGRAPH*, 2023. 1, 2, 3
- [29] Agelos Kratimenos, Jiahui Lei, and Kostas Daniilidis. DynMF: Neural Motion Factorization for Real-time Dynamic View Synthesis with 3D Gaussian Splatting. *ECCV*, 2024. 1
- [30] Jinxi Li, Ziyang Song, and Bo Yang. NVFi: Neural Velocity Fields for 3D Physics Learning from Dynamic Videos. *NeurIPS*, 2023. 1, 3, 5, 6, 2, 4, 7, 8
- [31] Xuan Li, Yi-Ling Qiao, Peter Yichen Chen, Krishna Murthy Jatavallabhula, Ming Lin, Chenfanfu Jiang, and Chuang Gan. PAC-NeRF: Physics Augmented Continuum Neural Radiance Fields for Geometry-Agnostic System Identification. *ICLR*, 2023. 1
- [32] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural Scene Flow Fields for Space-Time View Synthesis of Dynamic Scenes. *CVPR*, 2021. 5, 6, 7
- [33] Zhengqi Li, Qianqian Wang, Forrester Cole, Richard Tucker, and Noah Snavely. DynIBaR: Neural Dynamic Image-Based Rendering. *CVPR*, 2023. 1
- [34] Zhan Li, Zhang Chen, Zhong Li, and Yi Xu. Spacetime Gaussian Feature Splatting for Real-Time Dynamic View Synthesis. *CVPR*, 2024. 2

- [35] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft COCO: Common Objects in Context. *ECCV*, 2014. 6
- [36] Youtian Lin, ZuoZhuo Dai, Siyu Zhu, and Yao Yao. Gaussian-Flow: 4D Reconstruction with Dynamic 3D Gaussian Particle. *CVPR*, 2024. 2
- [37] Fangfu Liu, Hanyang Wang, Shunyu Yao, and Shengjun Zhang. Physics3D: Learning Physical Properties of 3D Gaussians via Video Diffusion. *arXiv:2406.04338*, 2024. 2
- [38] Jia-Wei Liu, Yan-Pei Cao, Weijia Mao, Wenqiao Zhang, David Junhao Zhang, Jussi Keppo, Ying Shan, Xiaohu Qie, and Mike Zheng Shou. DeVRF: Fast Deformable Voxel Radiance Fields for Dynamic Scenes. *NeurIPS*, 2022. 2
- [39] Shaowei Liu, Zhongzheng Ren, Saurabh Gupta, and Shenglong Wang. PhysGen: Rigid-Body Physics-Grounded Image-to-Video Generation. *ECCV*, 2024. 2
- [40] Yixin Liu, Kai Zhang, Yuan Li, Zhiling Yan, Chujie Gao, Ruoxi Chen, Zhengqing Yuan, Yue Huang, Hanchi Sun, Jianfeng Gao, Lifang He, and Lichao Sun. Sora: A Review on Background, Technology, Limitations, and Opportunities of Large Vision Models. *arXiv:2402.17177*, 2024. 2
- [41] 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. *CVPR*, 2023. 2
- [42] Zhicheng Lu, Xiang Guo, Le Hui, Min Yang, Xiao Tang, Feng Zhu, and Yuchao Dai. 3D Geometry-aware Deformable Gaussian Splatting for Dynamic View Synthesis. *CVPR*, 2024. 2
- [43] J. Macqueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, Berkeley, Calif., 1967. University of California Press. 6
- [44] Arjun Mani, Ishaan Preetam Chandratreya, Elliot Creager, Carl Vondrick, and Richard Zemel. SURFSUP: Learning Fluid Simulation for Novel Surfaces. *ICCV*, 2023. 2
- [45] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy Networks: Learning 3D Reconstruction in Function Space. *CVPR*, 2019. 2
- [46] Marko Mihajlovic, Sergey Prokudin, Siyu Tang, Robert Maier, Federica Bogo, Tony Tung, and Edmond Boyer. SplatFields: Neural Gaussian Splats for Sparse 3D and 4D Reconstruction. *ECCV*, 2024. 2
- [47] 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. *ECCV*, 2020. 1, 2
- [48] Haomiao Ni, Changhao Shi, Kai Li, Sharon X. Huang, and Martin Renqiang Min. Conditional Image-to-Video Generation with Latent Flow Diffusion Models. *CVPR*, 2023. 2
- [49] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. *CVPR*, 2019. 2
- [50] 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. *SIGGRAPH Asia*, 2021. 2
- [51] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural Radiance Fields for Dynamic Scenes. *CVPR*, 2021. 1, 5, 6, 4, 7
- [52] Zhiyin Qian, Shaoifei Wang, Marko Mihajlovic, Andreas Geiger, and Siyu Tang. 3DGS-Avatar: Animatable Avatars via Deformable 3D Gaussian Splatting. *CVPR*, 2024. 2
- [53] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 2019. 1, 2
- [54] Maziar Raissi, Alireza Yazdani, and George Em Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 2020. 2
- [55] Johannes L. Schonberger and Jan-Michael Frahm. Structure-from-Motion Revisited. *CVPR*, 2016. 3, 5
- [56] Richard Shaw, Jifei Song, Arthur Moreau, Michal Nazarczuk, Sibi Catley-Chandar, Helisa Dharmo, and Eduardo Perez-Pellitero. SWAGS: Sampling Windows Adaptively for Dynamic 3D Gaussian Splatting. *ECCV*, 2024. 2
- [57] Ziyang Song and Bo Yang. OGC: Unsupervised 3D Object Segmentation from Rigid Dynamics of Point Clouds. *NeurIPS*, 2022. 6, 3
- [58] Ziyang Song and Bo Yang. Unsupervised 3D Object Segmentation of Point Clouds by Geometry Consistency. *TPAMI*, 2024. 6
- [59] Ziyang Song, Jinxi Li, and Bo Yang. OSN: Infinite Representations of Dynamic 3D Scenes from Monocular Videos. *ICML*, 2024. 2
- [60] Ayush Tewari, Justus Thies, Ben Mildenhall, Pratul Srinivasan, Edgar Tretschk, Yifan Wang, Christoph Lassner, Vincent Sitzmann, Ricardo Martin-Brualla, Stephen Lombardi, Tomas Simon, Christian Theobalt, Matthias Niessner, Jonathan T. Barron, Gordon Wetzstein, Michael Zollhoefer, and Vladislav Golyanik. Advances in Neural Rendering. *SIGGRAPH*, 2021. 2
- [61] Fengrui Tian, Shaoyi Du, and Yueqi Duan. MonoNeRF: Learning a Generalizable Dynamic Radiance Field from Monocular Videos. *ICCV*, 2023. 2
- [62] 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. *ICCV*, 2021. 2
- [63] Alex Trevithick and Bo Yang. GRF: Learning a General Radiance Field for 3D Representation and Rendering. *ICCV*, 2021. 2
- [64] Vikram Voleti, Alexia Jolicœur-Martineau, and Christopher Pal. MCVD: Masked Conditional Video Diffusion for Prediction, Generation, and Interpolation. *NeurIPS*, 2022. 2
- [65] Bing Wang, Zhengdi Yu, Bo Yang, Jie Qin, Toby Breckon, Ling Shao, Niki Trigoni, and Andrew Markham.

- RangeUDF: Semantic Surface Reconstruction from 3D Point Clouds. *arXiv:2204.09138*, 2022. [2](#)
- [66] Yiming Wang, Siyu Tang, and Mengyu Chu. Physics-Informed Learning of Characteristic Trajectories for Smoke Reconstruction. *SIGGRAPH*, 2024. [1](#), [2](#)
- [67] William F. Whitney, Tatiana Lopez-Guevara, Tobias Pfaff, Yulia Rubanova, Thomas Kipf, Kimberly Stachenfeld, and Kelsey R. Allen. Learning 3D Particle-based Simulators from RGB-D Videos. *ICLR*, 2024. [1](#)
- [68] 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. *CVPR*, 2024. [1](#), [2](#), [3](#)
- [69] Ziyi Wu, Nikita Dvornik, Klaus Greff, Thomas Kipf, and Animesh Garg. SlotFormer: Unsupervised Visual Dynamics Simulation with Object-Centric Models. *ICLR*, 2023. [2](#)
- [70] Tianyi Xie, Zeshun Zong, Yuxing Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. Physgaussian: Physics-integrated 3d gaussians for generative dynamics. *CVPR*, 2024. [4](#)
- [71] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural Fields in Visual Computing and Beyond. *Computer Graphics Forum*, 41, 2022. [2](#)
- [72] Haotian Xue, Antonio Torralba, Daniel LK Yamins, Joshua B. Tenenbaum, Yunzhu Li, and Hsiao-Yu Tung. 3D-IntPhys: Learning 3D Visual Intuitive Physics for Fluids, Rigid Bodies, and Granular Materials. *NeurIPS*, 2023. [2](#)
- [73] Bo Yang, Stefano Rosa, Andrew Markham, Niki Trigoni, and Hongkai Wen. Dense 3D Object Reconstruction from a Single Depth View. *TPAMI*, 2019. [2](#)
- [74] Jing-wen Yang, Jia-mu Sun, Yong-liang Yang, and Jie Yang. DMiT: Deformable Mipmapped Tri-Plane Representation for Dynamic Scenes. *ECCV*, 2024. [2](#)
- [75] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3D Gaussians for High-Fidelity Monocular Dynamic Scene Reconstruction. *CVPR*, 2024. [1](#), [3](#), [4](#), [5](#), [6](#), [7](#), [8](#)
- [76] Zeyu Yang, Hongye Yang, Zijie Pan, Xiatian Zhu, and Li Zhang. Real-time Photorealistic Dynamic Scene Representation and Rendering with 4D Gaussian Splatting. *ICLR*, 2024. [2](#)
- [77] Yuxin Yao, Siyu Ren, Junhui Hou, Zhi Deng, Juyong Zhang, and Wenping Wang. DynoSurf: Neural Deformation-based Temporally Consistent Dynamic Surface Reconstruction. *ECCV*, 2024. [2](#)
- [78] Mingqiao Ye, Martin Danelljan, Fisher Yu, and Lei Ke. Gaussian grouping: Segment and edit anything in 3d scenes. In *ECCV*, 2024. [6](#), [3](#)
- [79] 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. *CVPR*, 2020. [5](#), [2](#)
- [80] Lijun Yu, Yong Cheng, Kihyuk Sohn, José Lezama, Han Zhang, Huiwen Chang, Alexander G. Hauptmann, Ming-Hsuan Yang, Yuan Hao, Irfan Essa, and Lu Jiang. MAGVIT: Masked Generative Video Transformer. *CVPR*, 2023. [2](#)
- [81] Yifan Zhan, Zhuoxiao Li, Muyao Niu, Zhihang Zhong, Shohei Nobuhara, Ko Nishino, and Yinqiang Zheng. KFD-NeRF: Rethinking Dynamic NeRF with Kalman Filter. *ECCV*, 2024. [2](#)
- [82] Tianyuan Zhang, Hong-Xing Yu, Rundi Wu, Brandon Y. Feng, Changxi Zheng, Noah Snively, Jiajun Wu, and William T. Freeman. PhysDreamer: Physics-Based Interaction with 3D Objects via Video Generation. *ECCV*, 2024. [2](#)
- [83] Licheng Zhong, Hong-xing Yu, Jiajun Wu, and Yunzhu Li. Reconstruction and Simulation of Elastic Objects with Spring-Mass 3D Gaussians. *ECCV*, 2024. [1](#), [2](#)
- [84] Chuhan Zou, Ersin Yumer, Jimei Yang, Duygu Ceylan, and Derek Hoiem. 3D-PRNN: Generating Shape Primitives with Recurrent Neural Networks. *ICCV*, 2017. [2](#)