

# TRACE: Learning 3D Gaussian Physical Dynamics from Multi-view Videos

Jinxi Li, Ziyang Song, Bo Yang\*

vLAR Group, The Hong Kong Polytechnic University

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

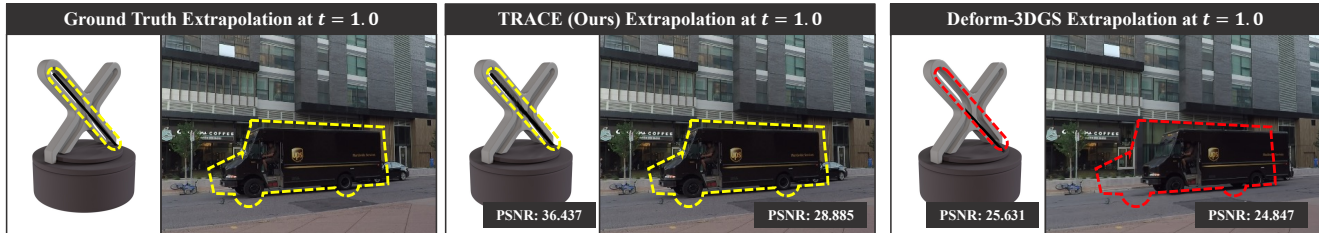


Figure 1. Given video frames of a real-world dynamic scene, our TRACE can learn the underlying physics and accurately predict the future motion of a rode passing through a hyperbolic slot and a track keep going forward, while the existing method cannot.

## Abstract

*In this paper, we aim to model 3D scene geometry, appearance, and physical information just from dynamic multi-view videos in the absence of any human labels. By leveraging physics-informed losses as soft constraints or integrating simple physics models into neural nets, existing works often fail to learn complex motion physics, or doing so requires additional labels such as object types or masks. We propose a new framework named **TRACE** to model the motion physics of complex dynamic 3D scenes. The key novelty of our method is that, by formulating each 3D point as a rigid particle with size and orientation in space, we directly learn a translation rotation dynamics system for each particle, explicitly estimating a complete set of physical parameters to govern the particle’s motion over time. Extensive experiments on three existing dynamic datasets and one newly created challenging synthetic datasets demonstrate the extraordinary performance of our method over baselines in the task of future frame extrapolation. A nice property of our framework is that multiple objects or parts can be easily segmented just by clustering the learned physical parameters. Our datasets and code are available at <https://github.com/vLAR-group/TRACE>.*

## 1. Introduction

Regarding our daily dynamic 3D scenes such as falling balls, rotating fans, and folding chairs, precisely modeling their geometry, appearance, and physical properties, and

further predicting their future states are crucial for emerging applications in robotics, mixed reality, and embodied AI. With the advancement of recent 3D representations such as NeRF [35] and 3DGS [19], a plethora of works [40, 52, 55] have been proposed to model various dynamic 3D scenes, achieving excellent performance in interpolating novel views within the observed time. However, they often fail to extrapolate future frames, essentially because they do not learn the underlying physics priors of 3D scenes.

To learn physics priors, existing methods mainly consist of two categories: 1) physics-informed neural network (PINN) based methods [43] which integrate the governing partial differential equations (PDEs) into loss functions to drive neural networks to learn physically plausible dynamic 3D scenes such as floating smoke [42] and simple moving objects [24]. While demonstrating promising results in modeling 3D geometry and physics such as velocity and viscosity, these methods usually need boundary constraints such as accurate object/foreground masks which may not always be available in practice. In addition, adding PINN losses is not a free lunch, but significantly sacrifices the efficiency in training and accuracy at boundary regions. 2) Physics model based methods [18, 51, 61] which encode various physics systems into neural networks to model elastic objects, fluids, *etc.*. Thanks to the explicit physics priors added, these methods obtain impressive results in physical properties learning and simulation. Nevertheless, they are often limited to specific types of objects, materials, or motions due to the lack of generality of encoded physics priors, thus being unable to predict future motions of complex dynamic 3D objects and scenes.

\*Corresponding Author

In this paper, we aim to introduce a new framework to model dynamic 3D scenes just from multi-view RGB videos, without needing any additional human labels such as object types or masks, ultimately being able to predict future frames viewing from arbitrary angles. Among various physical properties of a dynamic 3D scene, following the recent work NVFi [21], we also choose to learn a velocity field as it directly governs 3D scene movement. However, to accurately learn the physical velocity from RGB videos is extremely challenging, essentially due to the lack of sufficient physics constraints from raw color pixels. This problem is even harder when multiple objects or parts are undergoing rather different motion patterns. For example, regarding two adjacent objects moving in opposite directions in 3D space, the velocity of neighboring 3D surface points at the intersection region tends to have particularly distinct patterns. This means that the latent representation of per-point dynamics in 3D space could be discrete in nature. Therefore, it is more desirable to model per-point dynamics independently, thus every point having its unique motion. For generality, we regard each 3D point in space as a rigid particle with its size and orientation. If its size is zero, the rigid particle degenerates to a point.

With this insight, for each rigid particle in space, we propose to learn an independent dynamics system that includes a complete set of physical parameters to govern its motion over time. According to the laws of classical mechanics, for a specific rigid particle traversing 3D space over time, its motion can always be regarded as a rotational movement about a rotation center which has its own translation. In this regard, we choose to learn a translation rotation dynamics system for each rigid particle, allowing its future motion to be derived accordingly. Alongside learning the core dynamics, we must also model the geometry and appearance of 3D scenes. In this paper, we naturally choose 3D Gaussian Splatting (3DGS) [19] as the representation, thanks to its unprecedented fidelity in reconstruction and its particle (a Gaussian kernel) based representation in nature, which shares the basic concept of our defined rigid particle.

Our framework consists of two major components: 1) a **3D scene representation module** to learn dynamic scene geometry and appearance at a canonical timestamp, which is implemented by a vanilla 3DGS [19], though other variants can be adopted as well; 2) a **translation rotation dynamics system module** to learn a full set of physical parameters for each input rigid particle, which is just realized by multilayer perceptrons (MLPs). Based on these system parameters, the rigid particle’s velocity is then derived according to the laws of classical mechanics, without needing additional physics priors such as PINN [43] in training.

The key to our framework is the second module which simply regards each 3D Gaussian kernel as a rigid particle and takes it as input into MLPs. Nevertheless, we empiri-

cally find that it is hard to optimize this module due to the inaccuracy and instability of Gaussian kernels regressed at early training epochs. To tackle this issue, we simply train an auxiliary deformation field in parallel with our second module using an existing work such as [55] and [52].

Different from current works for modeling dynamic scenes, including NeRF-based methods, *e.g.*, D-NeRF[40]/TiNeuVox[13]/HexPlane[5], and 3DGS-based methods such as DefGS [55] and 4DGS [52], our core novelty is the introduced translation rotation dynamics system with an effective optimization strategy, allowing us to truly learn physical parameters, ultimately achieving future frame extrapolation. By comparison, all those existing methods fail to do so, as extensively verified in Tables 1.

By leveraging 3D Gaussians to model scene geometry and appearance, our method, named **TRACE**, learns translation rotation dynamics systems for estimating accurate physical dynamics. Figure 1 shows our qualitative results. Our contributions are:

- We introduce a new framework to model motion physics of complex dynamic 3D scenes, without needing prior knowledge of object shapes, types, or masks.
- We propose to learn a translation rotation dynamics system for each 3D rigid particle, thus allowing the velocity field to be derived without needing additional physics constraints in training.
- We demonstrate superior results in future frame extrapolation on three existing datasets, and one newly collected synthetic dataset with extremely challenging dynamics.

The concurrent work FreeGave [22] addresses a similar task to ours. However, the key difference is that TRACE explicitly models the changes (accelerations, jerks, or higher order) of physical motions, allowing continuous derivation of velocities over time, whereas FreeGave simply fits a velocity network over observed time in an implicit manner.

## 2. Related Works

**3D Shape Representations:** Static 3D objects and scenes are traditionally represented by voxels, point clouds, meshes, *etc.*, but they usually have limited representation capabilities due to the nature of discretization. Recently, implicit representations have been developed in the literature, including occupancy fields (OF) [7, 34], un/signed distance fields (U/SDF) [9, 37], and radiance fields (NeRF) [35]. Although demonstrating excellent performance in novel view synthesis and shape reconstruction, they are time-consuming to render 2D images or extract 3D shapes due to the integration of their continuous coordinate-based representations. To tackle this issue, the very recent 3D Gaussian Splatting [19] turns to represent a 3D shape as a set of explicit Gaussian kernels with various properties, achieving real-time rendering speed. In our framework, we

adopt this particle-based representation, as it is amenable to our particle-based physics learning framework.

**Dynamic 3D Reconstruction:** Recent advances in dynamic 3D reconstruction primarily follow the development of static 3D techniques such as SDF, NeRF, and Gaussian Splatting. To model the temporal relationship, existing works [3–5, 12, 13, 15, 16, 23, 25, 31, 32, 38, 39, 48–50, 53, 58] usually add the time dimension into static 3D representations to learn a motion or deformation field for rigid or deformable objects and scenes. Despite achieving excellent performance in novel view synthesis, especially when integrating 3DGS as the backbone [20, 26, 29, 33, 52, 55], these works can only interpolate 2D views within the observed time, instead of predicting physically meaningful future frames. Basically, this is because the commonly learned motion or deformation field does not encode physics priors in nature, but just fits the correlation between pixels. In this paper, the key difference between these works and ours is that we separately learn translation rotation dynamics systems for 3D rigid particles, thus allowing us to infer physically meaningful future frames, but they cannot.

**3D Physics Learning:** To learn various physical properties for 3D objects and scenes, the recent physics-informed neural networks (PINN) [2, 6, 17, 36, 42–44, 60] are widely applied to convert PDEs into loss functions as soft constraints, driving neural networks to learn physically meaningful targets. However, it is often inefficient to train PINNs due to the large amount of data samples needed to regularize, and the soft constraints are usually not sufficient to obtain satisfactory results. In this paper, we do not rely on such inefficient PINN losses to incorporate physics priors to train neural networks. Another line of works [10, 14, 41, 51, 54] integrate explicit physics systems such as springs, graphs, *etc.*, into the learning process to model elastic objects [30, 59, 61], fluids [18, 27], *etc.*, achieving impressive results in physics learning and simulation. In this paper, we also opt to learn physics systems. However, the core difference is that we learn a translation rotation dynamics system which is applicable to common deformation and transformation dynamics, whereas existing works often learn a spring or fluid system only applicable to elastic objects or fluids.

### 3. TRACE

Our framework mainly comprises two modules together with an auxiliary deformation field to model 3D geometry, appearance, and physics. Given dynamic multi-view RGB videos with known camera poses and intrinsics, the 3D scene representation module aims to learn a set of 3D Gaussian kernels to represent the 3D scene geometry and appearance in a canonical space. The auxiliary deformation field is designed to predict the translation and distortion of each Gaussian kernel given the current training time  $t$ . For

these two components, we simply follow the design of existing works [19, 55] briefly elaborated in Section 3.1. Notably, the deformation field alone cannot extrapolate frames beyond the training time. Our core module of the translation rotation dynamics system aims to learn a set of physical parameters for each 3D rigid particle, governing its motion dynamics over time, which is detailed in Section 3.2.

#### 3.1. Preliminary

For the input multi-view RGB videos,  $T$  represents the greatest timestamp in training and  $N$  the total number of cameras. For training stability, we first use all frames  $\{I_0^1 \cdots I_0^n \cdots I_0^N\}$  at time  $t = 0$  to train a reasonable static 3DGS model as an initialization of the 3D scene geometry and appearance, and then use the remaining frames to jointly optimize our translation and rotation dynamics system and the auxiliary deformation field.

**Canonical 3D Gaussians:** Following the vanilla 3DGS [19], we employ a set of learnable 3D Gaussian kernels  $G_0$  to represent the canonical scene geometry and appearance at  $t = 0$ . Each kernel is parameterized by a 3D position  $\mathbf{x}_0$ , covariance matrix obtained from quaternion  $\mathbf{r}_0$ , scaling  $\mathbf{s}_0$ , opacity  $\sigma$ , and color  $\mathbf{c}$  computed from spherical harmonics. Following prior works [52, 55], we assume the opacity  $\sigma$  and color  $\mathbf{c}$  of each Gaussian will not be updated, but constantly associated with the kernel and transported over time.

Given the  $N$  images at timestamp  $t = 0$ , we either initialize all canonical 3D Gaussian kernels  $G_0$  randomly or based on sparse points created by SfM [46]. To train all kernels, we exactly follow 3DGS [19] by 1) projecting Gaussian kernels into camera space, 2) rendering the projected kernels into image space, and 3) optimizing all kernel parameters via  $\ell_1$  and  $\ell_{ssim}$  losses used in 3DGS as follows.

$$\underbrace{\{\cdot (\mathbf{x}_0, \mathbf{r}_0, \mathbf{s}_0, \sigma, \mathbf{c}) \cdot\}}_{G_0} \xrightleftharpoons[\ell_1 + \ell_{ssim}]{\text{project+render}} \{I_0^1 \cdots I_0^N\} \quad (1)$$

**Auxiliary Deformation Field:** To aid the learning of our translation and rotation dynamics system, we leverage an existing deformation field [55], but we are also amenable to other deformable Gaussian methods such as 4DGS [52], as demonstrated in our experiments in Section 4.1. In particular, the 3D position  $\mathbf{x}_0$  of each canonical Gaussian kernel and the current timestamp  $t$  are fed into an MLP-based deformation network, denoted as  $f_{defo}$ , directly predicting the corresponding position displacement  $\delta\mathbf{x}$ , and the change of quaternion  $\delta\mathbf{r}$  and scaling  $\delta\mathbf{s}$  from timestamp 0 to  $t$ . All Gaussians  $G_t$  at time  $t$  can be easily computed as follows, where the operations  $\circ$  and  $\odot$  follow [55].

$$\mathbf{x}_t = \mathbf{x}_0 + \delta\mathbf{x}, \mathbf{r}_t = \mathbf{r}_0 \circ \delta\mathbf{r}, \mathbf{s}_t = \mathbf{s}_0 \odot \delta\mathbf{s}, \sigma, \mathbf{c} \quad (2)$$

All these deformed Gaussians will be projected and optimized by visual images at timestamp  $t$  in a later stage, as clarified in Section 3.3, where the deformation net  $f_{defo}$

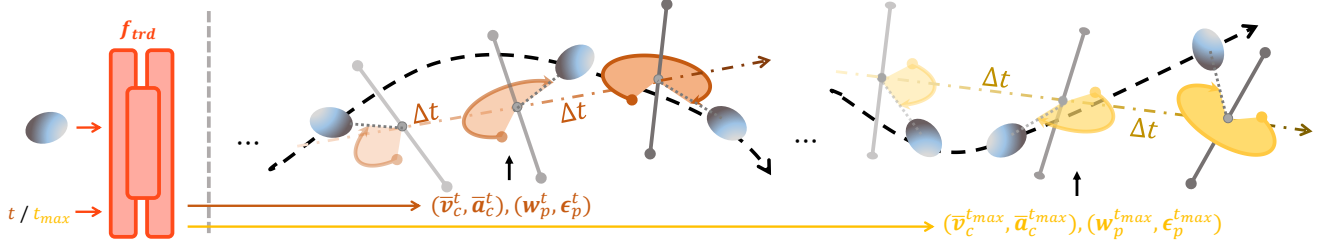


Figure 2. The proposed translation rotation dynamics system for a specific rigid particle. The rigid particle will be driven by its learned physical parameters over time, forming a trajectory in 3D space.

will be optimized from scratch. All details are provided in Appendix 5.1 and 5.2.

### 3.2. Translation Rotation Dynamics System

This module aims to learn physical parameters that govern the motion of 3D scenes. However, the dynamics of an entire space are extremely complex. Here, we simplify this problem and formulate it into just learning per rigid particle dynamics, where we treat each (canonical or deformed) Gaussian kernel as a rigid particle with size and orientation. According to the laws of classical mechanics, for a specific rigid particle  $\mathbf{P} \in \mathcal{R}^3$ , its motion in a 3D world coordinate system can be regarded as a rotational movement about a rotation center which has its own translation. To this end, we aim to learn the following two groups of physical parameters for each 3D rigid particle  $\mathbf{P}$ :

- Group #1 - Rotation Center Parameters including: 1) the center’s position  $\mathbf{P}_c \in \mathcal{R}^3$ , 2) the center’s velocity  $\mathbf{v}_c \in \mathcal{R}^3$ , and 3) acceleration  $\mathbf{a}_c \in \mathcal{R}^3$  in world coordinate.
- Group #2 - Rigid Particle Rotational Parameters including: 1) the rigid particle’s rotation vector  $\mathbf{w}_p \in \mathcal{R}^3$  with regard to its center  $\mathbf{P}_c$ , and 2) the rigid particle’s angular acceleration  $\epsilon_p \in \mathcal{R}^3$ .

As illustrated in Figure 2, our translation rotation dynamics system module, denoted as  $f_{trd}$ , takes a rigid particle  $\mathbf{P}$  and a time  $t$  as input, directly predicting the physical parameters of its rotation center and its own rotational information. Then, this rigid particle will be driven by its learned physical parameters, forming its motion dynamics, as illustrated by the trajectory in Figure 2. The composite velocity for the rigid particle  $\mathbf{P}$  at time  $t$  is derived as follows.

$$\mathbf{v}_p^t = \mathbf{w}_p^t \times (\mathbf{P} - \mathbf{P}_c^t) + \mathbf{v}_c^t = \mathbf{w}_p^t \times \mathbf{P} + (\mathbf{v}_c^t - \mathbf{w}_p^t \times \mathbf{P}_c^t) \quad (3)$$

Nevertheless, the rightmost block of above equation shows that the estimation of the center’s velocity  $\mathbf{v}_c^t$  and center’s position  $\mathbf{P}_c^t$  is compounded. Thus, instead of separately learning these two parameters, we turn to learn an equivalent (compounded) center velocity  $\bar{\mathbf{v}}_c^t$ . Similarly, we instead learn an equivalent center acceleration  $\bar{\mathbf{a}}_c^t$ :

$$\bar{\mathbf{v}}_c^t = \mathbf{v}_c^t - \mathbf{w}_p^t \times \mathbf{P}_c^t, \quad \bar{\mathbf{a}}_c^t = \mathbf{a}_c^t - \epsilon_p^t \times \mathbf{P}_c^t. \quad (4)$$

More details about the definition of equivalent parameters and the proof for equivalence are provided in Appendix

5.3. This  $f_{trd}$  module is implemented by simple MLPs as:

$$\{(\bar{\mathbf{v}}_c^t, \bar{\mathbf{a}}_c^t), (\mathbf{w}_p^t, \epsilon_p^t)\} = f_{trd}(\mathbf{P}, t) \quad (5)$$

Notably, for an input rigid particle  $\mathbf{P}$ , the elegance of this module  $f_{trd}$  is that it only needs to learn this full translation rotation dynamics system at one specific time  $t$ , and that particle’s future motion will be governed by the learned dynamics system according to the laws of mechanics.

Due to the complex change of direction of rotation vector  $\mathbf{w}_p$  driven by the angular acceleration  $\epsilon_p$  for a particle, we use Runge-Kutta  $2^{nd}$ -order method to numerically calculate the evolving dynamics to derive the future parameters, as detailed in Algorithm 1.

Instead of using  $2^{nd}$ -order updating scheme, we can also extend to higher orders or reduce to lower orders with regard to future time. Intuitively, a higher order relationship is expected to capture extremely complex dynamics such as a rolling ball suddenly breaking up into pieces due to unknown explosives inside, whereas a much lower order relationship tends to only capture static or constant speed scenes, thus being oversimplified. In this paper, we opt for the above  $2^{nd}$ -order scheme to update dynamics parameters for two primary reasons:

- In many applications such as robot manipulation, the need for future prediction usually involves a relatively short interval, e.g., in milliseconds. In this case, a  $2^{nd}$ -order update is usually sufficient to achieve decent approximations. Additionally, a simple sliding window based approach can be applied to continuously predict future frames given the newest visual observations from sensors.
- In our daily life, the majority of common physical movements such as rolling balls or moving cars can be generally described by a  $2^{nd}$ -order relationship. In fact, both Newton’s First and Second Law of Motion can be captured. Notably, since the whole 3D scene comprises a large number of rigid particles, each has up to  $2^{nd}$ -order dynamics. Therefore, the compounded dynamics for the entire 3D scene can be rather complex, including various deformations and transformations in our daily lives.

Nevertheless, it is still interesting yet non-trivial to learn much higher-order relationships, and we leave it for future exploration. Implementation details are in Appendix 5.5.

### 3.3. Training

With our translation rotation dynamics module and the auxiliary deformation field, we now discuss how to train them together, such that physical parameters can be truly learned.

**First**, for two timestamps  $t'$  and  $t$ , where  $t$  is usually sampled from the training set and  $\Delta t = t - t'$  is predefined to be small enough, we can easily obtain Gaussians  $G_{t'}$  from the deformation field  $f_{defo}$  based on Equation 2.

**Second**, having our translation rotation dynamics module  $f_{trd}$  at hand, we naturally regard the transportation of all kernels from  $t'$  to  $t$  is governed by the corresponding physical parameters estimated by  $f_{trd}$  at time  $t'$  as defined in Equation 5. Then we use the Runge-Kutta  $2^{nd}$ -order (RK2) method to numerically derive the future physical parameters. Thus we can easily compute the kernel's orientation  $\mathbf{r}_t$  from  $\mathbf{r}_{t'}$  and the kernel's position translation  $\mathbf{x}_t$  from  $\mathbf{x}_{t'}$ . The kernel's translation consists of two parts: 1) the equivalent translation of its rotation center, and 2) the equivalent displacement caused by the kernel's rotation with regard to its center. In particular, the RK2 method is defined as:

---

#### Algorithm 1 Runge-Kutta $2^{nd}$ -order (RK2) method

---

**Input:** (1) Gaussians at  $t'$ :  $\{\cdot \cdot (\mathbf{x}_{t'}, \mathbf{r}_{t'}, \mathbf{s}_{t'}, \sigma, \mathbf{c}) \cdot \cdot\}$ , and (2)  $\Delta t$   
**Output:** Gaussians at  $t = (t' + \Delta t)$ :  $\{\cdot \cdot (\mathbf{x}_t, \mathbf{r}_t, \mathbf{s}_t, \sigma, \mathbf{c}) \cdot \cdot\}$

- $\{(\bar{\mathbf{v}}_c^{t'}, \bar{\mathbf{a}}_c^{t'}), (\mathbf{w}_p^{t'}, \epsilon_p^{t'})\} \leftarrow f_{trd}(\mathbf{x}_{t'}, t')$ , (ref to Eq 5)
- $\bar{\mathbf{v}}_c^{mid} \leftarrow \bar{\mathbf{v}}_c^{t'} + \frac{\Delta t}{2} \bar{\mathbf{a}}_c^{t'}$
- $\mathbf{w}_p^{mid} \leftarrow \mathbf{w}_p^{t'} + \frac{\Delta t}{2} \epsilon_p^{t'}$
- $\mathbf{x}_t \leftarrow \mathbf{x}_{t'} + \Delta t(\bar{\mathbf{v}}_c^{mid} + \mathbf{w}_p^{mid} \times \mathbf{x}_{t'})$
- Convert quaternion to rotation matrix:  $\mathbf{R}_{t'} \leftarrow \mathbf{r}_{t'}$
- Calculate the Cross-Product Matrix, denoted by  $\mathbf{W}_p \in \mathcal{R}^{3 \times 3}$ , for the normalized vector  $\mathbf{w}_p^{mid} / \|\mathbf{w}_p^{mid}\|$ . Details are in Appendix 5.4.
- $\Delta \mathbf{R} \leftarrow \mathbf{I} + (\sin \Delta \theta) \mathbf{W}_p + (1 - \cos \Delta \theta) \mathbf{W}_p^2$ , where  $\Delta \theta = \Delta t \mathbf{w}_p^{mid}$ , following Rodrigue's Formula [45]
- $\mathbf{R}_t \leftarrow (\Delta \mathbf{R}) \mathbf{R}_{t'}$
- Convert rotation matrix to quaternion:  $\mathbf{r}_t \leftarrow \mathbf{R}_t$
- Assign  $\mathbf{s}_{t'}$  to  $\mathbf{s}_t$ :  $\mathbf{s}_t \leftarrow \mathbf{s}_{t'}$ , as each Gaussian is rigid with the same scale by definition

**Return:** All Gaussians at  $t$ :  $\{\cdot \cdot (\mathbf{x}_t, \mathbf{r}_t, \mathbf{s}_t, \sigma, \mathbf{c}) \cdot \cdot\}$

---

**Third**, we render all the above Gaussians kernels at time  $t$  to 2D image space following 3DGS, comparing with the training images at time  $t$ . All parameters are trained with:

$$(\mathbf{G}_0, f_{defo}, f_{trd}) \leftarrow (\ell_1 + \ell_{ssim}) \quad (6)$$

## 4. Experiments

**Datasets:** Our method is designed to learn meaningful physical information of 3D dynamic scenes, aiming at accurately predicting future motions, instead of just fitting observed video frames. In this regard, the closest work to us is the recent NVFi [21]. Following NVFi, we primarily evaluate our method on its three dynamic datasets: **1) Dynamic Object dataset** consisting of 6 dynamic objects. Each object displays a unique motion pattern belonging to either

rigid or deformable movement. **2) Dynamic Indoor Scene dataset** with 4 complex indoor scenes. Each scene has multiple objects undergoing different rigid body motions. **3) NVIDIA Dynamic Scene dataset** [57]. It consists of two chosen real-world dynamic 3D scenes.

Upon a closer look at the above three datasets, we find that their dynamics captured are relatively simple. In our daily life, the majority of objects and scenes have multiple parts undergoing radically different motions over time, showing challenging physical patterns to learn. To further evaluate the effectiveness of our design, we collect a new synthetic dataset, named **4) Dynamic Multipart dataset**.

Our new synthetic dataset comprises 4 objects. Each has 2 to 5 distinct motion patterns on different object parts. Following [21], for each object, we collect RGBs at 15 different viewing angles, where each viewing angle has 60 frames captured. We reserve the first 46 frames at randomly picked 12 viewing angles as the training split, *i.e.*, 552 frames, while leaving the 46 frames at the remaining 3 viewing angles for testing *interpolation* ability, *i.e.*, 138 frames for novel view synthesis within the training time period, and keeping the last 14 frames at all 15 viewing angles for evaluating future frame *extrapolation*, *i.e.*, 210 frames.

**Baselines:** We select the following baselines: **1) NVFi** [21]: This is the closest work to us, but differs from us in two folds. First, NVFi relies on PINN losses to learn physics priors, but we directly learn physical parameters. Second, NVFi adopts NeRF as a backbone, being short in 3D scene geometry and appearance modeling, but our method is amenable to and adopts the powerful 3DGS in nature. **2) T-NeRF** [40]. **3) D-NeRF** [40]. **4) TiNeuVox** [13]. The latter four methods are based on NeRF and designed for novel view interpolation. Therefore they are expected to be rather weak for future frame extrapolation. For a fair and extensive comparison, we also include the following two baselines. **5) DefGS** [55], and **6) 4DGS** [52]. Both very recent deformable 3D Gaussians methods are particularly strong to model dynamic 3D scenes for novel view synthesis using 3DGS as a backbone. **7) DefGS<sub>nvfi</sub>**. We build this baseline by combining DefGS with the velocity field proposed by NVFi. This baseline has the powerful 3DGS as a backbone as well as the current state-of-the-art NVFi learning strategy. It is trained with exactly the same settings as our method. To demonstrate the flexibility of our framework, we also adopt 4DGS as our auxiliary deformation field, denoted as TRACE<sub>4dgs</sub>.

**Metrics:** We report PSNR/ SSIM/ LPIPS scores for RGB synthesis in future frame extrapolation.

### 4.1. Main Results of Future Frame Extrapolation

All methods are trained in a scene-specific fashion. As a common practice, the actual time length in all datasets is normalized to be one time unit, where the first ( $0 \sim 0.7$ )

Table 1. Quantitative results of all methods for future frame extrapolation on all four datasets.

	Dynamic Object			Dynamic Indoor Scene			NVIDIA Dynamic Scene			Dynamic Multipart		
	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[40]	13.818	0.739	0.324	22.242	0.700	0.363	21.120	0.707	0.358	10.064	0.576	0.537
D-NeRF[40]	14.660	0.737	0.312	20.791	0.692	0.349	20.633	0.709	0.327	13.344	0.767	0.340
TiNeuVox[13]	19.612	0.940	0.073	21.029	0.770	0.281	24.556	0.863	0.215	20.804	0.923	0.090
T-NeRF <sub>PINN</sub>	16.189	0.835	0.230	17.290	0.477	0.618	17.975	0.605	0.428	-	-	-
HexPlane <sub>PINN</sub>	21.419	0.946	0.067	23.091	0.742	0.401	24.473	0.818	0.279	-	-	-
NVFi[21]	27.594	0.972	0.036	29.745	0.876	0.204	<u>28.462</u>	0.876	0.214	25.235	0.955	0.046
DefGS[55]	19.849	0.949	0.045	21.380	0.819	0.188	24.240	0.895	0.140	20.664	0.930	0.067
DefGS <sub>nvfi</sub>	28.749	<u>0.984</u>	<u>0.013</u>	<u>31.096</u>	<u>0.945</u>	<u>0.077</u>	27.529	<u>0.927</u>	<u>0.102</u>	28.455	0.979	0.017
4DGS[52]	20.354	0.950	0.052	21.107	0.793	0.274	22.510	0.703	0.408	20.564	0.935	0.067
<b>TRACE<sub>Adgs</sub> (Ours)</b>	<u>30.327</u>	0.983	0.019	30.607	0.896	0.209	22.554	0.721	0.390	<u>32.317</u>	<u>0.985</u>	<u>0.016</u>
<b>TRACE (Ours)</b>	<b>31.597</b>	<b>0.987</b>	<b>0.009</b>	<b>34.824</b>	<b>0.965</b>	<b>0.054</b>	<b>29.341</b>	<b>0.933</b>	<b>0.074</b>	<b>33.481</b>	<b>0.990</b>	<b>0.007</b>

time unit is training time and the remaining is test time. In training and test, we set  $\Delta t$  to be 2 divided by the training set frame rate. To extrapolate future frames, the time  $t'$  for our auxiliary deformation field  $f_{defo}$  is set as 0.7. The future frames are progressively calculated by Algorithm 1 with the same time interval  $\Delta t$ .

Our primary goal is to extrapolate meaningful future frames as a continuum of the last training observations. In our evaluation, we follow the second step in Section 3.3 to extrapolate future frames from the last timestamp of training frames. We also compare novel view (past frame) interpolation with baselines, but this is less important. Details of interpolation and the results are provided in Appendix 5.7.

**Results & Analysis:** Table 1 compare future frame extrapolation of all methods on four datasets. We can see that:

- Compared with NeRF and 3DGS based dynamic scene modeling methods such as T-NeRF/ D-NeRF/ TiNeuVox/ DefGS/ 4DGS, both versions of our method achieve about 10 points higher on PSNR for future frame extrapolation. This means that, without explicitly learning physical information like us, these methods completely fail to predict the future, highlighting the core value of our method.
- Compared with the closest and also strongest baselines NVFi/ DefGS<sub>nvfi</sub>, our method is still constantly better than them on all datasets for future frame extrapolation. Notably, on the Dynamic Indoor Scene dataset and our newly collected Dynamic Multipart dataset, there are much more complex motion dynamics such as different objects or parts moving in distinct directions, but our best results are constantly about 3 points higher on PSNR than them. Fundamentally, this is because both NVFi and DefGS<sub>nvfi</sub> rely on PINN losses as soft constraints to incorporate physics priors, whereas we directly integrate hard physics by learning translation rotation parameters, thus being more effective in learning dynamics.
- Lastly, our framework is indeed amenable to existing deformation fields such as DefGS and 4DGS, and both versions achieve superior results for future frame extrapolation on most datasets.

Figure 4 shows qualitative results. More results are in Appendix 5.16 / 5.17 / 5.18 / 5.19. We also report the training/test time, memory cost, *etc.*, in Appendix 5.6.

## 4.2. Analysis of Dynamics Parameters

Our core translation rotation dynamics system module is designed to learn per rigid particle’s physical parameters. Ideally, for those particles undergoing the same motion pattern such as all surface points of a single rigid part, they should have the same or similar physical parameters. Given this, multiple dynamic objects or parts with distinct motions can be automatically segmented based on the similarity of learned physical parameters. By comparison, the prior work NVFi [21] can hardly achieve this autonomous dynamic segmentation by its own design, unless an additional motion grouping method is applied. To further evaluate this nice property of our method, we conduct the following steps to analyze the learned dynamics parameters.

First, after training our method on the Dynamic Object dataset, Dynamic Multipart dataset, and Dynamic Indoor Scene dataset, for each dynamic scene, we have a set of well-trained canonical 3D Gaussians, an auxiliary deformation field, and our translation rotation dynamics parameters.

Then, we use the auxiliary deformation field  $f_{defo}$  to deform the canonical 3D Gaussians  $\mathcal{G}_0$  to time  $t = 0.7$ , which is the maximum time the deformation field can query in our training. At this timestamp, the motions of different objects and parts normally achieve a steady state.

Lastly, we query all the learned physical parameters at this time, *i.e.*,  $\{(\bar{\mathbf{v}}_c^t, \bar{\mathbf{a}}_c^t), (\mathbf{w}_p^t, \epsilon_p^t)\}$ . We choose  $(\|\bar{\mathbf{v}}_c^t\|, \bar{\mathbf{v}}_c^t, \|\mathbf{w}_p^t\|, \mathbf{w}_p^t / \|\mathbf{w}_p^t\|)$  as the features to cluster Gaussians via a simple K-means algorithm. As shown in Figure 3, all Gaussians can be grouped into physically meaningful objects or parts according to their actual motion patterns. More results are in Appendix 5.21 / 5.22.

We further quantitatively evaluate our motion grouping results on Dynamic Indoor Scene Dataset. In particular, we follow Gaussian Grouping [56] to render 2D object segmentation masks for all 30 views over 60 timestamps on

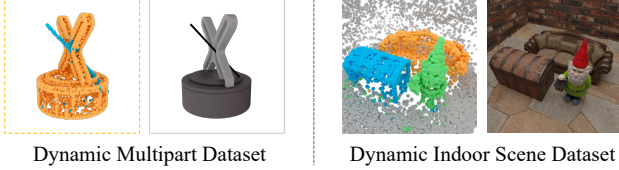


Figure 3. Clustering of translation rotation physics parameters.

Table 2. Quantitative results of motion segmentation results on Dynamic Indoor Scene dataset.

	AP $\uparrow$	PQ $\uparrow$	F1 $\uparrow$	Pre $\uparrow$	Rec $\uparrow$	mIoU $\uparrow$
M2F[8]	65.37	73.14	78.29	94.83	68.88	64.42
D-NeRF[40]	57.26	46.15	59.02	56.55	62.94	46.58
NVFi[21]	91.21	78.74	93.75	93.76	93.74	67.64
DefGS[55]	51.73	57.60	66.43	63.21	70.07	54.46
DefGS <sub>nvfi</sub>	55.26	62.75	69.83	69.39	72.91	56.82
<b>TRACE (Ours)</b>	<b>95.82</b>	<b>93.28</b>	<b>97.90</b>	<b>96.21</b>	<b>99.86</b>	<b>79.55</b>

all 4 scenes, *i.e.*, 7200 images in total. We compare with **D-NeRF**, **NVFi**, **DefGS** and **DefGS<sub>nvfi</sub>**. We follow NVFi to obtain segmentation results of D-NeRF and NVFi. For the 3DGS-based baselines, we also adopt OGC [47] to segment Gaussians based on scene flows induced from their learned deformation fields. All implementation details are in Appendix 5.9. We also include a strong 2D object segmentation method, Mask2Former [8] pre-trained by human annotations on COCO [28] as a fully supervised baseline.

As shown in Table 2, our method achieves almost perfect object segmentation results on all metrics, significantly outperforming all baselines. This shows that our learned physical parameters correctly model object physical motion patterns and can be easily used to identify objects according to their motions, without needing any human annotations.

### 4.3. Continual Learning

We include an additional continual learning experiment to show that our framework can easily adapt to new observations to learn rapidly changing dynamics. We test on **ParticleNeRF dataset** [1]. It has 6 dynamic objects. Each object undergoes various rigid motions from robot manipulation to harmonic oscillation or deformable motions such as cloth. Details of the dataset are in Appendix 5.11.

Particularly, we first feed frames of (0 ~ 0.15) time unit to train the network, and evaluate future frame extrapolation on (0.15 ~ 0.30). Next, we include (0.15 ~ 0.30) to continue training, and then evaluate future frame extrapolation on (0.30 ~ 0.45). We keep adding a time unit of 0.15 till training on (0 ~ 0.75), and extrapolate on (0.75 ~ 0.9).

We compare our extrapolation performance with DefGS and DefGS<sub>nvfi</sub>. Table 3 shows quantitative results. It can be seen that DefGS and DefGS<sub>nvfi</sub> struggle in making right future predictions based on novel observations, while our model can stably adapt to new observations and achieve excellent future frame predictions. This means that even though the 3D scene dynamics are radically and rapidly changing over time, our model can continue capturing those dynamics given new observations. We believe this would

be a key enabler for robot perception and manipulation in highly dynamic environments. Qualitative extrapolation and past time interpolation results are in Appendix 5.11.

Table 3. Quantitative results (PSNR) of continual learning.

Observed Time	0.15	0.30	0.45	0.60	0.75	Average
Extrap Till	0.30	0.45	0.60	0.75	0.90	
DefGS[55]	19.856	18.652	20.141	20.813	19.131	19.718
DefGS <sub>nvfi</sub>	26.105	25.987	27.122	26.051	25.640	26.181
<b>TRACE (Ours)</b>	<b>26.731</b>	<b>27.623</b>	<b>27.168</b>	<b>28.422</b>	<b>27.777</b>	<b>27.544</b>

### 4.4. Ablation Study

To verify different choices of our method, we conduct the following five groups of ablation experiments.

**(1) Different choices of time difference  $\Delta t$  in training stage:** Given the time interval between two consecutive frames in the training set as  $\delta t$ , we compare three choices of the time difference  $\Delta t$  in the training stage:  $\{\delta t, 2\delta t, 3\delta t\}$ . We choose  $\Delta t = 2\delta t$  in our main experiments.

**(2) Different choices of the order of physical dynamics:** Our translation rotation dynamic systems choose to learn 2<sup>nd</sup>-order dynamics in our main experiments. We compare the choices for the 1<sup>st</sup> order (no acceleration) and the 3<sup>rd</sup> order (acceleration of acceleration).

**(3) Removing the auxiliary deformation field  $f_{defo}$ :** We feed Gaussian  $s$  at time  $t' = 0$  directly into  $f_{trd}$ , and use the output physical parameters to progressively move  $s$  to a target future time  $t$  with  $\Delta t = 2\delta t$ .

**(4) Removing the physics derivation:** Instead of using the physics parameters queried at time  $t'$  to derive the future motion states at time  $t > t'$ , we query the physics parameters from  $f_{trd}(\mathbf{P}, t)$  for future timestamps.

**(5) Removing the equivalence parametrization:** Instead of learning the equivalent velocity and acceleration for the centers as Equation 4, we directly learn the original parameters, implemented by a same MLP only with a modified output layer:  $\{(\mathbf{P}_c^t, \mathbf{v}_c^t, \mathbf{a}_c^t), (\mathbf{w}_p^t, \epsilon_p^t)\} = f_{trd'}(\mathbf{P}, t')$ .

Table 4. Ablation studies on three datasets.

order	$f_{defo}$	physics equiv	Dynamic Multipart			Dynamic Object			Dynamic Indoor Scene					
			PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$			
(1)	$\delta t$	2	✓	✓	✓	32.852	0.989	0.010	31.597	0.987	0.009	33.831	0.958	0.078
(1)	$2\delta t$	2	✓	✓	✓	33.481	0.990	0.007	31.511	0.988	0.006	34.824	0.965	0.054
(1)	$3\delta t$	2	✓	✓	✓	33.003	0.989	0.008	29.787	0.983	0.011	34.778	0.965	0.054
(2)	$2\delta t$	1	✓	✓	✓	33.125	0.989	0.010	28.522	0.984	0.012	34.576	0.963	0.076
(2)	$2\delta t$	3	✓	✓	✓	33.312	0.989	0.008	31.044	0.987	0.007	34.086	0.961	0.056
(3)	$2\delta t$	2	✗	✓	✓	19.206	0.907	0.120	-	-	-	-	-	-
(4)	$2\delta t$	2	✓	✗	✓	29.602	0.983	0.012	-	-	-	-	-	-
(5)	$2\delta t$	2	✓	✓	✗	29.986	0.985	0.012	-	-	-	-	-	-

**Results & Analysis:** Table 4 shows all ablation results for future frame extrapolation. To demonstrate the robustness of our hyperparameters selected, the first two groups of ablations are extensively conducted on three datasets. The remaining ablations are primarily conducted on our new Dynamic Multipart dataset. It can be seen that: 1) The greatest impact is caused by the removal of the deformation field  $f_{defo}$ . While this deformation field itself is unable to

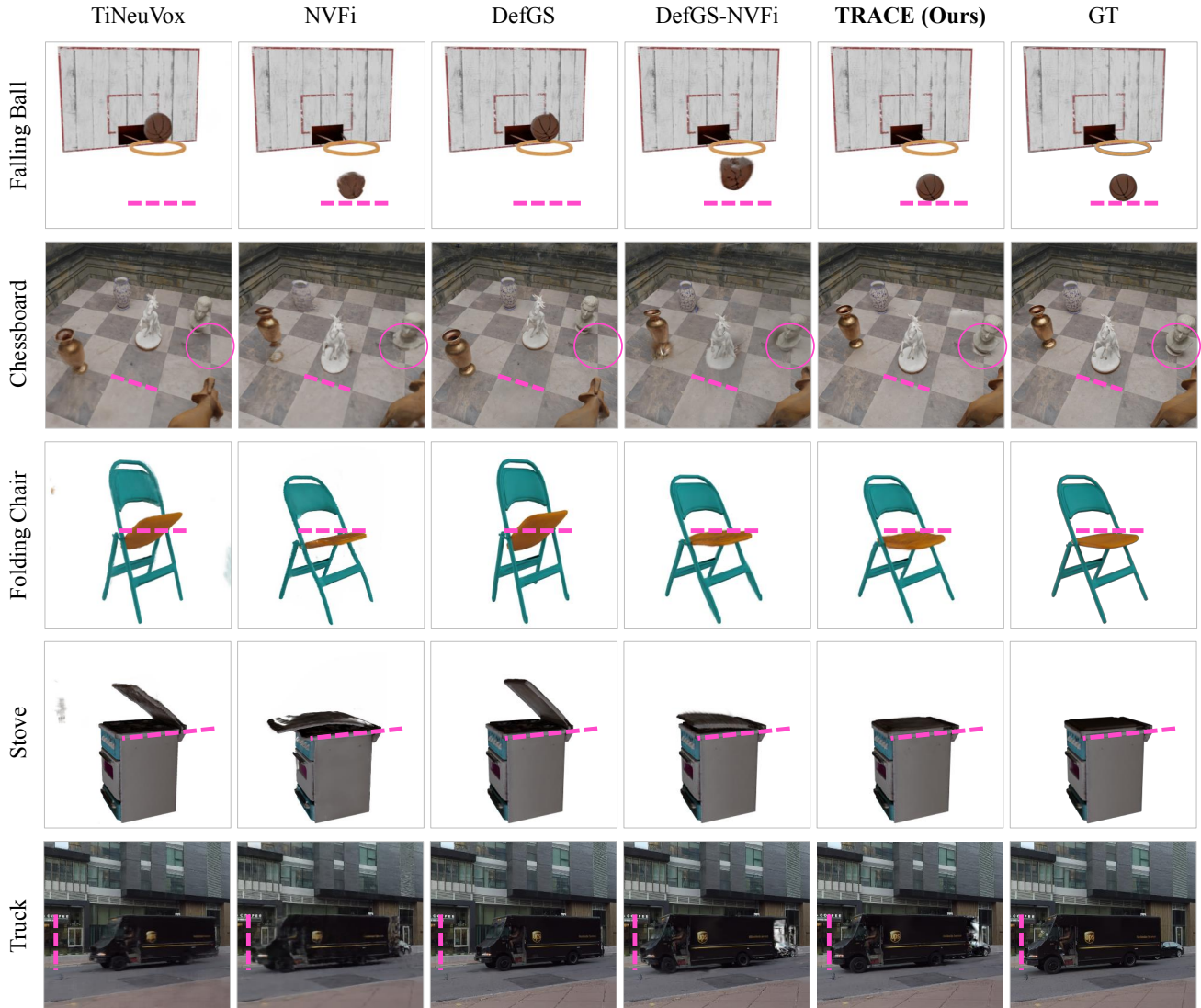


Figure 4. Qualitative results of RGB view synthesis for future frame extrapolation on four datasets.

learn physics, it significantly aids our core translation rotation dynamics system module to learn physical parameters given motion information. 2) The choice of time difference  $\Delta t$  is also important. Once it is as small as the interval between two consecutive frames, the performance drops, because the motion in short intervals is too subtle to be distinguished. For example, a rotation may be learned as a translation. However, if  $\Delta t$  is too large, the appearance fitting could be sacrificed, so the performance is slightly weaker. 3) If physical parameters are learned but not derived, the lack of physics consistency will influence motion learning. Ablation results for interpolation are in Appendix 5.13.

## 5. Conclusion

In this paper, we have demonstrated that complex motion dynamics can be explicitly learned just from multi-view

RGB videos without needing additional human labels such as object types and masks. This is achieved by a new generic framework that simultaneously models 3D scene geometry, appearance and physics by extending the appealing 3D Gaussian Splatting technique. In contrast to existing works which usually rely on PINN losses as soft constraints to learn physics priors, we instead directly learn a complete set of physical parameters to govern the motion pattern of each 3D rigid particle in space via our core translation rotation dynamics system module. Extensive experiments on three public dynamic datasets and a newly created dynamic multipart dataset have shown the extraordinary performance of our method in the challenging task of future frame extrapolation over all baselines. In addition, the learned physical parameters can be directly used to segment objects or parts according to the similarity of parameters.

**Acknowledgment:** This work was supported in part by Research Grants Council of Hong Kong under Grants 25207822 & 15225522 & 15219125, in part by Otto Poon Charitable Foundation Smart Cities Research Institute (8-CDCQ), in part by Research Centre for Unmanned Autonomous Systems (1-CE3D), The Hong Kong Polytechnic University.

## References

- [1] Jad Abou-Chakra, Feras Dayoub, and Niko Sünderhauf. Particlenerf: Particle based encoding for online neural radiance fields. *arXiv preprint arXiv:2211.04041*, 2022. 7
- [2] Daniele Baieri, Stefano Esposito, Filippo Maggioli, and Emanuele Rodolà. Fluid Dynamics Network: Topology-Agnostic 4D Reconstruction via Fluid Dynamics Priors. *arXiv:2303.09871*, 2023. 3
- [3] Jonathan T Barron, Keunhong Park, Steven M Seitz, and Ricardo Martin-brualla. Nerfies: Deformable Neural Radiance Fields. *ICCV*, 2021. 3
- [4] Hongrui Cai, Wanquan Feng, Xuetao Feng, Yan Wang, and Juyong Zhang. Neural Surface Reconstruction of Dynamic Scenes with Monocular RGB-D Camera. *NeurIPS*, 2022.
- [5] Ang Cao and Justin Johnson. HexPlane: A Fast Representation for Dynamic Scenes. *CVPR*, 2023. 2, 3
- [6] Nithin Chalapathi, Yiheng Du, and Aditi S Krishnapriyan. Scaling physics-informed hard constraints with mixture-of-experts. *ICLR*, 2024. 3
- [7] Zhiqin Chen and Hao Zhang. Learning Implicit Fields for Generative Shape Modeling. *CVPR*, 2019. 2
- [8] Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention Mask Transformer for Universal Image Segmentation. *CVPR*, 2022. 7
- [9] Julian Chibane, Aymen Mir, and Gerard Pons-Moll. Neural Unsigned Distance Fields for Implicit Function Learning. *NeurIPS*, 2020. 2
- [10] Yitong Deng, Hong-Xing Yu, Jiajun Wu, and Bo Zhu. Learning Vortex Dynamics for Fluid Inference and Prediction. *ICLR*, 2023. 3
- [11] Carl Doersch, Pauline Luc, Yi Yang, Dilara Gokay, Skanda Koppula, Ankush Gupta, Joseph Heyward, Ignacio Rocco, Ross Goroshin, João Carreira, and Andrew Zisserman. BootsTAP: Bootstrapped training for tracking-any-point. *Asian Conference on Computer Vision*, 2024. 6
- [12] Yilun Du, Yinan Zhang, and Joshua B Tenenbaum. Neural Radiance Flow for 4D View Synthesis and Video Processing. *ICCV*, 2021. 3
- [13] Jiemin Fang, Xingang Wang, and Matthias Nießner. Fast Dynamic Radiance Fields with Time-Aware Neural Voxels. *SIGGRAPH Asia*, 2022. 2, 3, 5, 6, 7, 8, 9
- [14] Erik Franz, Barbara Solenthaler, and Nils Thuerey. Learning to Estimate Single-view Volumetric Flow Motions without 3D Supervision. *ICLR*, 2023. 3
- [15] Sara Fridovich-Keil, Giacomo Meanti, Frederik Warburg, Benjamin Recht, and Angjoo Kanazawa. K-Planes: Explicit Radiance Fields in Space, Time, and Appearance. *CVPR*, 2023. 3
- [16] Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang. Dynamic View Synthesis from Dynamic Monocular Video. *ICCV*, 2021. 3
- [17] Zhongkai Hao, Chengyang Ying, Hang Su, Jun Zhu, Jian Song, and Ze Cheng. Bi-level Physics-Informed Neural Networks for PDE Constrained Optimization Using Broyden’s Hypergradients. *ICLR*, 2023. 3
- [18] 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, 3
- [19] 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
- [20] Jiahui Lei, Yijia Weng, Adam Harley, Leonidas Guibas, and Kostas Daniilidis. MoSca: Dynamic Gaussian Fusion from Casual Videos via 4D Motion Scaffolds. *arXiv:2405.17421*, 2024. 3
- [21] Jinxi Li, Ziyang Song, and Bo Yang. NVFi: Neural Velocity Fields for 3D Physics Learning from Dynamic Videos. *NeurIPS*, 2023. 2, 5, 6, 7, 3, 8, 9
- [22] Jinxi Li, Ziyang Song, Siyuan Zhou, and Bo Yang. FreeGave: 3D Physics Learning from Dynamic Videos by Gaussian Velocity. *CVPR*, 2025. 2
- [23] Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, and Zhaoyang Lv. Neural 3D Video Synthesis From Multi-View Video. *CVPR*, 2022. 3
- [24] 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
- [25] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural Scene Flow Fields for Space-Time View Synthesis of Dynamic Scenes. *CVPR*, 2021. 3, 8
- [26] Zhan Li, Zhang Chen, Zhong Li, and Yi Xu. Spacetime Gaussian Feature Splatting for Real-Time Dynamic View Synthesis. *CVPR*, 2024. 3
- [27] Marten Lienen, David Lüdke, Jan Hansen-Palmus, and Stephan Gunnemann. From Zero to Turbulence: Generative Modeling for 3D Flow Simulation. *ICLR*, 2024. 3
- [28] 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. 7
- [29] Youtian Lin, Zuozhuo Dai, Siyu Zhu, and Yao Yao. Gaussian-Flow: 4D Reconstruction with Dynamic 3D Gaussian Particle. *CVPR*, 2024. 3
- [30] Fangfu Liu, Hanyang Wang, Shunyu Yao, and Shengjun Zhang. Physics3D: Learning Physical Properties of 3D Gaussians via Video Diffusion. *arXiv:2406.04338*, 2024. 3
- [31] Isabella Liu, Hao Su, and Xiaolong Wang. Dynamic Gaussians Mesh: Consistent Mesh Reconstruction from Monocular Videos. *arXiv:2404.12379*, 2024. 3

- [32] 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. 3
- [33] 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. 3
- [34] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy Networks: Learning 3D Reconstruction in Function Space. *CVPR*, 2019. 2
- [35] 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
- [36] Siddhartha Mishra and Roberto Molinaro. Estimates on the generalization error of physics-informed neural networks for approximating PDEs. *IMA Journal of Numerical Analysis*, 2023. 3
- [37] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. *CVPR*, 2019. 2
- [38] 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. 3
- [39] Sunghoon Park, Minjung Son, Seokhwan Jang, Young Chun Ahn, Ji-Yeon Kim, and Nahyup Kang. Temporal Interpolation Is All You Need for Dynamic Neural Radiance Fields. *CVPR*, 2023. 3
- [40] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural Radiance Fields for Dynamic Scenes. *CVPR*, 2021. 1, 2, 5, 6, 7, 3, 8, 9
- [41] Yi-Ling Qiao, Alexander Gao, and Ming C. Lin. NeuPhysics: Editable Neural Geometry and Physics from Monocular Videos. *NeurIPS*, 2022. 3
- [42] Jiaxiong Qiu, Ruihong Cen, Zhong Li, Han Yan, Mingming Cheng, and Bo Ren. NeuSmoke: Efficient Smoke Reconstruction and View Synthesis with Neural Transportation Fields. *SIGGRAPH Asia*, 2024. 1, 3
- [43] 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*, 378, 2019. 1, 2
- [44] Maziar Raissi, Alireza Yazdani, and George Em Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 2020. 3
- [45] Olinde Rodrigues. Des lois géométriques qui régissent les déplacements d'un système solide dans l'espace, et de la variation des coordonnées provenant de ces déplacements considérés indépendants des causes qui peuvent les produire. *Journal de Mathématiques Pures et Appliquées*, 1840. 5
- [46] Johannes L. Schonberger and Jan-Michael Frahm. Structure-from-Motion Revisited. *CVPR*, 2016. 3
- [47] Ziyang Song and Bo Yang. OGC: Unsupervised 3D Object Segmentation from Rigid Dynamics of Point Clouds. *NeurIPS*, 2022. 7, 4
- [48] Fengrui Tian, Shaoyi Du, and Yueqi Duan. MonoNeRF: Learning a Generalizable Dynamic Radiance Field from Monocular Videos. *ICCV*, 2023. 3
- [49] 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.
- [50] Chaoyang Wang, Ben Eckart, Simon Lucey, and Orazio Gallo. Neural Trajectory Fields for Dynamic Novel View Synthesis. *arXiv:2105.05994*, 2021. 3
- [51] 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, 3
- [52] GuanJun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4D Gaussian Splatting for Real-Time Dynamic Scene Rendering. *CVPR*, 2024. 1, 2, 3, 5, 6
- [53] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time Neural Irradiance Fields for Free-Viewpoint Video. *CVPR*, 2021. 3
- [54] 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. 3
- [55] 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, 2, 3, 5, 6, 7, 8, 9
- [56] Mingqiao Ye, Martin Danelljan, Fisher Yu, and Lei Ke. Gaussian grouping: Segment and edit anything in 3d scenes. In *ECCV*, 2024. 6
- [57] 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
- [58] Meng You and Junhui Hou. Decoupling Dynamic Monocular Videos for Dynamic View Synthesis. *arXiv:2304.01716*, 2023. 3
- [59] 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. *arXiv:2404.13026*, 2024. 3
- [60] Zhiyuan Zhao, Xueying Ding, and B. Aditya Prakash. PINNsFormer: A Transformer-Based Framework For Physics-Informed Neural Networks. *ICLR*, 2024. 3
- [61] Licheng Zhong, Hong-xing Yu, Jiajun Wu, and Yunzhu Li. Reconstruction and Simulation of Elastic Objects with Spring-Mass 3D Gaussians. *ECCV*, 2024. 1, 3
- [62] Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus H. Gross. EWA volume splatting. *IEEE Visualization*, 2001. 1