NeuroFluid: Fluid Dynamics Grounding with Particle-Driven Neural Radiance Fields

Shanyan Guan, Huayu Deng, Yunbo Wang (✉), Xiaokang Yang
MoE Key Lab of Artificial Intelligence, AI Institute
Shanghai Jiao Tong University, Shanghai 200240, China
{shyanguan, deng_hy99, yunbow, xkyang}@sjtu.edu.cn

March 4, 2022

Abstract

Deep learning has shown great potential for modeling the physical dynamics of complex particle systems such as fluids (in Lagrangian descriptions). Existing approaches, however, require the supervision of consecutive particle properties, including positions and velocities. In this paper, we consider a partially observable scenario known as *fluid dynamics grounding*, that is, inferring the state transitions and interactions within the fluid particle systems from sequential visual observations of the fluid surface. We propose a differentiable two-stage network named *NeuroFluid*. Our approach consists of (i) a particle-driven neural renderer, which involves fluid physical properties into the volume rendering function, and (ii) a particle transition model optimized to reduce the differences between the rendered and the observed images. NeuroFluid provides the first solution to unsupervised learning of particle-based fluid dynamics by training these two models jointly. It is shown to reasonably estimate the underlying physics of fluids with different initial shapes, viscosity, and densities. It is a potential alternative approach to understanding complex fluid mechanics, such as turbulence, that are difficult to model using traditional methods of mathematical physics.

1 Introduction

Intuitive physics or intuitive physical inference is a research area that gathers a lot of interest in the machine learning community. Recent methods mainly focus on building neural models to reason about stability, collisions, forces, and velocities from images or videos [Battaglia et al., 2013, Wu et al., 2017a]. In this paper, we explore a new problem in intuitive physics, namely *fluid dynamics grounding*, defined as inferring the physical dynamics of fluids from a sequence of 2D visual observations collected from a sparse set of views. As shown in Figure 1, to infer the underlying physics, the key is to solve the inverse problem of synthesizing the visual scenes. *It is a potential alternative approach to understanding complex fluid mechanics, such as turbulence, that are difficult to model using traditional methods of mathematical physics.*

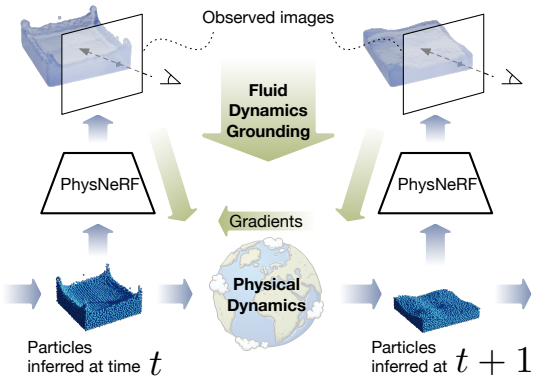

Figure 1: Fluid dynamics grounding is to reason about the underlying physical dynamics in fluid particle systems from sequential visual observations. We propose *NeuroFluid*, the first fully differentiable solution to this problem, in which a key component is the particle-driven neural renderer named *PhysNeRF*.

A typical forward modeling process includes two steps: fluid simulation and dynamic scene rendering. Despite recent progress in learning particle-based simulators from Lagrangian descriptions in fluid mechanics [Li et al., 2018b, Ummenhofer et al., 2019, Kim et al., 2019, Sanchez-Gonzalez et al., 2020], it remains an open question *whether neural networks can infer fluid dynamics directly from observed images*, since all existing work requires engineered simulators to provide consecutive particle locations as training data. To answer this question, we propose *NeuroFluid*, the first fully differentiable solution to fluid dynamics grounding. The key idea is to link particle-based fluid simulation with particle-driven

neural rendering in an end-to-end trainable framework, such that the two networks can be jointly optimized to obtain reasonable particle representations between them.

However, since we generally do not have any prior knowledge of the physical properties of the fluid in this intuitive physics setup, it is difficult to impose learning constraints on the particle representations directly. Therefore, to keep fluid dynamics grounding from an undesirable trivial solution to "de-rendering" the visual observations, we propose *PhysNeRF*, a neural renderer in forms of a particle-driven and geometry-dependent *neural radiance field* (NeRF). It takes as input the estimated particles and calculates their geometric relations with the naïve sampling points of NeRF that emit view-dependent radiance. We use PhysNeRF as a key component of NeuroFluid to allow grounding non-trivial physical dynamics from visual observations.

NeuroFluid is evaluated on dynamic scenes of fluids with different initial shapes, viscosity, and densities. It achieves competitive results in the accuracy of particle state inference and the quality of novel view synthesis and future rollouts.

The contributions of this paper are summarized as follows:

- We present and explore a new research field in intuitive physics, which we refer to as fluid dynamics grounding.
- We propose NeuroFluid, the first differentiable model that attempts to understand the fluid dynamics by solving the inverse problem of synthesizing the visual scenes.
- We propose PhysNeRF, which allows for joint optimization of dynamics propagation and rendering by geometrically correlating fluid particles with neural radiance fields.

2 Problem Formulation

We solve the problem of grounding the physical dynamics of particle-based fluid systems from a sequence of visual observations $\{I_t^{\boldsymbol{d}}\}_{t=1:T}$ collected from a sparse set of views $\{\boldsymbol{d}\}$. The dynamics can be represented by a particle state transition model $\mathcal{T}_\theta$. Unlike existing work for fluid simulation, in our problem setup, the parameters θ cannot be optimized with ground-truth particle states. Given the initial particles state $\boldsymbol{s}_0$, $\mathcal{T}_\theta$ estimates the transitions of particle states from $\boldsymbol{s}_t$ to $\boldsymbol{s}_{t+1}$ with probability $p(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t;\theta)$. We then receive a new observation $I_{t+1}^{\boldsymbol{d}}$ that can be used to solve the inverse graphics problem of particle-based fluid rendering, that is, to analyze the underlying physical properties of visual scenes by learning a differentiable renderer $\mathcal{R}_\phi$ to synthesize them. The forward modeling process of fluid dynamics grounding can be formulated as:

$$\boldsymbol{s}_{t+1} \leftarrow \mathcal{T}_\theta(\boldsymbol{s}_t), \quad \widehat{I}_{t+1}^{\boldsymbol{d}} \leftarrow \mathcal{R}_\phi(\boldsymbol{s}_t, \boldsymbol{d}), \tag{1}$$

where $\widehat{I}_{t+1}^{\boldsymbol{d}}$ is the synthesized image at time $(t+1)$ with view direction $\boldsymbol{d}$. The main objective of fluid dynamics grounding is to obtain optimal θ^* and ϕ^* that can maximize the log-likelihood function of the observed images:

$$\arg\max_{\theta,\phi} \sum_{t,\boldsymbol{d}} \log\left(p(I_{t+1}^{\boldsymbol{d}}|\boldsymbol{s}_{t+1}, \boldsymbol{d};\phi)\ p(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t;\theta)\right). \tag{2}$$

We evaluate $\mathcal{T}_\theta$ by (i) measuring the distance between the optimized particle positions $\{\widehat{\boldsymbol{P}}_t\}_{1:T}$ and the true positions $\{\boldsymbol{P}_t\}_{1:T}$. Further, it can be partly assessed through forward modeling of (ii) novel view synthesis and (iii) rolling-out $\mathcal{T}_\theta$ iteratively to predict multiple steps into the future.

3 NeuroFluid

In this section, we present the details of NeuroFluid, which is an optimization-based approach that understands fluid physics by learning to synthesize sequences of visual observations. The overall framework in the forward modeling phase includes two steps: fluid simulation and dynamic scene rendering (see Figure 1). They are in form of neural networks parametrized by θ and ϕ respectively.

In Section 3.1, we first describe the particle-based representations that connect fluid dynamics modeling with neural rendering and then describe the particle-based dynamics modeling approach in NeuroFluid. In Section 3.2, we revisit the rendering principles of NeRF and present the particle-driven neural radiance fields that are specifically designed for fluid dynamic scenes. In Section 3.3, we discuss

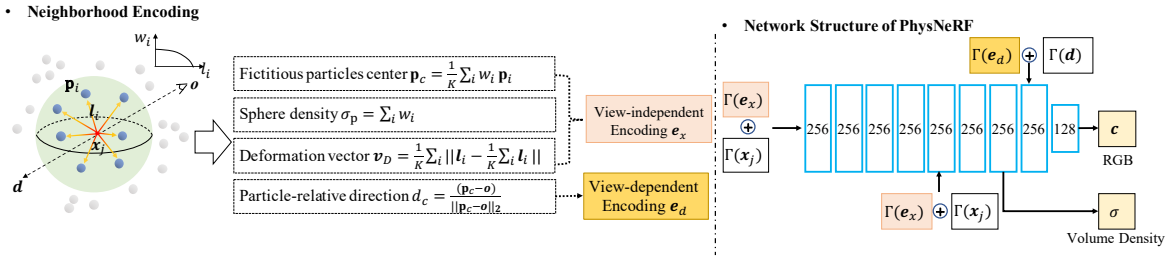

Figure 2: Overview of PhysNeRF. (**Left**) The physical properties of fluid particles are involved in the radiance fields through the proposed neighborhood encoding scheme; (**Right**) PhysNeRF employs view-independent particle encoding $\boldsymbol{e}_x$ and view-dependent particle encoding $\boldsymbol{e}_d$ to estimate the volume density σ and the emitted color $\boldsymbol{c}$.

the optimization procedure of NeuroFluid, which is the key to solving the inverse problem of rendering the physical scenes. Notably, NeuroFluid is the first fully differentiable model for fluid dynamics grounding.

3.1 Particle-Based Dynamics Modeling

3.1.1 Why particle-based representations?

Particle-based representations facilitate fluid dynamics grounding. Fluid representation techniques have been fully explored in previous literature, in which the particle-based approaches have the advantage of simulating complex fluid dynamics by modeling the interactions of a group of particles. Another advantage of particle-based representations is that they can easily back-propagate gradients, and they are therefore widely used in differentiable fluid simulators.

These properties make the particle-based representations particularly suitable for our optimization-based dynamics grounding approach: First, in the forward modeling process, they can intuitively reflect fluid dynamics and 3D geometries, and thus effectively drive the subsequent neural renderer. Second, during the optimization process, the particles estimated by the state transition model ($\mathcal{T}_\theta$) receive the gradients generated by the error of the rendering results, allowing $\mathcal{T}_\theta$ to gradually approach the real fluid dynamics.

In practice, we explicitly specify the particle states $\boldsymbol{s}_t$ in Eq. (1) and Eq. (2) as the particle positions $\boldsymbol{P}_t$ and velocities $\boldsymbol{V}_t$ over the entire particle set.

3.1.2 Particle Transition Model

In forward modeling, the particle transition model $\mathcal{T}_\theta$ learns the movement and interactions of particles between consecutive time steps. Starting with the initial particle states $(\boldsymbol{P}_0, \boldsymbol{V}_0)$, which are known during training and testing phases, it roll-outs over time to estimate a sequence of particle states $\{(\boldsymbol{P}_t, \boldsymbol{V}_t)\}_{1:T}$. At each time step, $(\boldsymbol{P}_t, \boldsymbol{V}_t)$ represent the underlying geometric information of the fluid; They are then fed into a particle-driven neural renderer to synthesize view-dependent images. A well-performed particle transition model can effectively predict multiple time steps into the future beyond the observed time scope during training time.

It is worth noting that, NeuroFluid does not have special requirements for the transition model, in the sense that any particle-based fluid simulation networks can plug and play in our framework as the particle transition model. Without loss of generality, we here use a state-of-the-art differentiable fluid simulator named Deep Lagrangian Fluids (DLF) [Ummenhofer et al., 2019]. It leverages an improved convolutional operator to capture the physical relations between the particles in small neighborhoods of 3D space, thus achieving a compromise between prediction accuracy and computational efficiency.

Due to the lack of ground-truth $\{(\boldsymbol{P}_t, \boldsymbol{V}_t)\}_{1:T}$, the key challenge in dynamics grounding is to find an appropriate way to supervise $\mathcal{T}_\theta$ such that it can intuitively approximate the physical dynamics of the fluid. This appropriate way, in our case, is the particle-driven neural radiance fields.

3.2 Particle-Driven Neural Radiance Fields

3.2.1 Revisiting NeRF

In the conventional approach of neural radiance fields (NeRF) [Mildenhall et al., 2020], the color of each pixel on the rendered image is decided in the following steps. First, a ray is emitted from the camera location $\boldsymbol{o}$ along the view direction $\boldsymbol{d}$, which can be denoted by $\boldsymbol{r} = \boldsymbol{o} + n\boldsymbol{d}$. Then, N points with 3D coordinates $\{\boldsymbol{x}_i\}_{1:N}$ are sampled along $\boldsymbol{r}$ between the near and far bounds of the physical scene. Next, a multilayer perceptron (MLP) is trained to map $(\boldsymbol{x}_i, \boldsymbol{d})$ to the intrinsic volume density $\sigma(\boldsymbol{x}_i)$ and the view-dependent emitted color $\boldsymbol{c}(\boldsymbol{x}_i, \boldsymbol{d})$. Finally, the RGB value $C(\boldsymbol{r})$ is decided using a classic volume rendering function [Kajiya and Von Herzen, 1984]:

$$\begin{aligned} C(\boldsymbol{r}) &= \sum_{i=1}^{N} T_i \Big(1 - \exp\big(-\sigma(\boldsymbol{x}_i)\delta_i\big)\Big)\boldsymbol{c}(\boldsymbol{x}_i, \boldsymbol{d}), \\ T_i &= \exp\left(-\sum_{j=1}^{i-1} \sigma(\boldsymbol{x}_j)\delta_j\right), \end{aligned} \tag{3}$$

where $\delta_j = \|\boldsymbol{x}_{j+1} - \boldsymbol{x}_j\|_2$ is the interval between two adjacent sampling points. Note that the positional encoding and hierarchical sampling strategy are omitted for brevity. In our work, $C(\boldsymbol{r})$ is represented as $I_t^{\boldsymbol{d}}(u, v)$, where (u, v) is the interaction between the ray and the image plane.

3.2.2 PhysNeRF: Linking Lagrangian Particles with Radiance Fields

We consider how fluid particles are associated with the radiance field in neural rendering. The motivation is that *a reasonable design of the renderer that conforms to physics can intuitively facilitate grounding fluid dynamics through joint optimization of particle transition and neural rendering.*

One hint is that the volume rendering result along a given ray is closely related to the geometric distribution of its neighboring physical particles. In extreme cases, the fewer particles around the ray, the closer the accumulated color $I(u, v)$ is to the background color. We propose PhysNeRF based on this assumption. Given a sampling point at $\boldsymbol{x}_j$, PhysNeRF first conducts ball query [Qi et al., 2017] to search its neighboring fluid particles $\boldsymbol{P}^{\boldsymbol{x}_i} \in \boldsymbol{P}_t$:

$$\boldsymbol{P}^{\boldsymbol{x}_j} \leftarrow \mathcal{S}(\|\mathbf{p}_i - \boldsymbol{x}_j\|_2, r_s), \tag{4}$$

where $\mathbf{p}_i$ refers to the i-th particle in $\boldsymbol{P}_t$, produced by $\mathcal{T}_\theta$ at time step t, and r_s is the search radius. In line with the particle transition model from [Ummenhofer et al., 2019], we set $r_s = 9 \cdot r_p$, where r_p is radius of the fluid particle. Furthermore, we assume that both the view-independent volume density and view-dependent color of the sampling point should be dependent on the geometric properties $\boldsymbol{P}^{\boldsymbol{x}_i}$ in its 3D neighborhood. Therefore, we parameterize $\boldsymbol{P}^{\boldsymbol{x}_i}$ to view-independent encoding $\boldsymbol{e}_x$ and view-dependent encoding $\boldsymbol{e}_d$ that can be written as follows:

$$(\boldsymbol{e}_x, \boldsymbol{e}_d) \leftarrow \mathcal{E}(\boldsymbol{P}^{\boldsymbol{x}_j}, \boldsymbol{x}_j), \tag{5}$$

where $\mathcal{E}(\cdot)$ refers to the parameterization operators, which will be described later. Finally, we train an MLP network, denoted by $\mathcal{R}_\phi$, to map the compositional inputs $(\boldsymbol{e}_x, \boldsymbol{e}_d)$ and $(\boldsymbol{x}_i, \boldsymbol{d})$ to volume density and emitted color. The rendering mechanism of PhysNeRF can be formulated as follows:

$$\big(\sigma(\boldsymbol{x}_j, \boldsymbol{e}_x), \boldsymbol{c}(\boldsymbol{x}_j, \boldsymbol{d}, \boldsymbol{e}_d)\big) \leftarrow \mathcal{R}_\phi(\boldsymbol{x}_j, \boldsymbol{d}, \boldsymbol{e}_x, \boldsymbol{e}_d). \tag{6}$$

Notably, the parameters of the MLP are shared at different time steps throughout the dynamic scene. With a fixed viewpoint, the differences in the rendering results are only determined by the geometric distribution of $\boldsymbol{P}_t$. It is the unique property of PhysNeRF that makes it different from all existing NeRF-based neural renderers.

3.2.3 Neighborhood Encoding in PhysNeRF

We here introduce how to encode the neighboring particles $\boldsymbol{P}^{\boldsymbol{x}_j}$ to $(\boldsymbol{e}_x, \boldsymbol{e}_d)$. As shown in Figure 2, we first define $\mathbf{p}_c$ as the weighted average position over $\boldsymbol{P}^{\boldsymbol{x}_j}$:

$$\mathbf{p}_c = \frac{1}{K} \sum_i w_i \mathbf{p}_i, \quad \mathbf{p}_i \in \boldsymbol{P}^{\boldsymbol{x}_j}, \tag{7}$$

where K is the number of neighboring particles. We use w_i to indicate the contribution weight of each particle to form the fictitious center particle, which is calculated by

$$w_i = \max\Big(1 - \big(\frac{\|\boldsymbol{l}_i\|_2}{r_s}\big)^3, 0\Big), \tag{8}$$

where $\boldsymbol{l}_i = \mathbf{p}_i - \boldsymbol{x}_j$ is the local vector from the sampling point $\boldsymbol{x}_j$ to particle $\mathbf{p}_i$. Intuitively, w_i describes the fact that the closer $\mathbf{p}_i$ is to the sampling point $\boldsymbol{x}_j$, the greater its contribution to the rendering result at $\boldsymbol{x}_j$. Having the fictitious center particle $\mathbf{p}_c$, we calculate the normalized view direction from the camera location $\boldsymbol{o}$ to $\mathbf{p}_c$, which is an importance reference direction for network to inference ray refraction and reflection, as $\boldsymbol{d}_c = (\mathbf{p}_c - \boldsymbol{o})/\|\mathbf{p}_c - \boldsymbol{o}\|_2$.

To infer the volume density σ at $\boldsymbol{x}_j$, a straightforward idea is to use the number of particles in $\boldsymbol{P}^{\boldsymbol{x}_j}$ as a closely related physical condition. Nevertheless, to avoid optimizing a discrete variable, we define a soft version of the particle density around $\boldsymbol{x}_j$ by calculating $\sigma_p = \sum_i w_i$. It meets the fact that the closer of $\mathbf{p}_i$ is to sampling point $\boldsymbol{x}_j$, the less likely the ray is to pass through $\boldsymbol{x}_j$.

Although $\boldsymbol{P}^{\boldsymbol{x}_j}$ is searched by a isotropic sphere, the actual particle distribution within it is anisotropic. Partly inspired by the work from [Biedert et al., 2018], we calculate a radial vector to represent the deformation in the particle set:

$$\boldsymbol{e}_r = \frac{1}{K}\sum_i \big\|\boldsymbol{l}_i - \frac{1}{K}\sum_i \boldsymbol{l}_i\big\|_2. \tag{9}$$

Finally, we take into account all of the above physical quantities and derive the view-independent encoding and the view-dependent encoding as

$$\boldsymbol{e}_x = \big(\Gamma(\mathbf{p}_c), \Gamma(\sigma_p), \Gamma(\boldsymbol{e}_r)\big), \quad \boldsymbol{e}_d = \Gamma(\boldsymbol{d}_c), \tag{10}$$

where $\Gamma(\cdot)$ refers to the positional encoding functions in NeRF [Mildenhall et al., 2020].

3.3 Optimizing NeuroFluid

Similar to NeRF, at a specific time step, we use the mean squared error to constrain the rendering result:

$$\mathcal{L}(\widehat{I}_t^{\boldsymbol{d}}, I_t^{\boldsymbol{d}}) = \sum_{u,v} \big\|\widehat{I}_t^{\boldsymbol{d}}(u,v) - I_t^{\boldsymbol{d}}(u,v)\big\|_2^2, \tag{11}$$

where $I_t^{\boldsymbol{d}}(u,v)$ is typically represented as $C(\boldsymbol{r})$ in NeRF. Since we generally do not have any prior knowledge of the physical properties of the fluid, we can hardly impose further constraints on particle states. Under these circumstances, jointly training $\mathcal{T}_\theta$ and $\mathcal{R}_\phi$ from a cold start makes the optimization prone to collapse to some undesirable local minima. Consequently, we first warm-up PhysNeRF on the initial fluid scene with a sparse set of $N_{\boldsymbol{d}}$ views conditioned on $(\boldsymbol{P}_0, \boldsymbol{V}_0)$ to obtain an effective initialization of the renderer. The objective function in the warm-up phase is

$$\underset{\phi}{\text{minimize}} \sum_{\boldsymbol{d}} \mathcal{L}(\widehat{I}_0^{\boldsymbol{d}}, I_0^{\boldsymbol{d}}; \boldsymbol{P}_0, \boldsymbol{V}_0, \phi). \tag{12}$$

After warming-up PhysNeRF, it roughly learns the correlation between view-dependent observations and particle geometries, we then jointly train $\mathcal{T}_\theta$ and $\mathcal{R}_\phi$ on $\{I_t^{\boldsymbol{d}}\}_{t=1:T}$, collected from a static camera with a fixed direction of $\boldsymbol{d}$. The objective function in this end-to-end training phase is

$$\underset{\theta,\phi}{\text{minimize}} \sum_{t=1}^{T} \mathcal{L}(\widehat{I}_t^{\boldsymbol{d}}, I_t^{\boldsymbol{d}}; \boldsymbol{P}_0, \boldsymbol{V}_0, \theta, \phi). \tag{13}$$

3.3.1 Implementation and Training Details

The model architecture of PhysNeRF is same as NeRF, except that the input channels of the input layers are changed for taking as input $(\boldsymbol{e}_x, \boldsymbol{e}_d)$. The hierarchical sampling strategy is adopted. We coarsely sample 64 points and finely sample 128 points along each ray. The positional encoding parameters of

Table 1: The typical simulation or rendering related properties of the used benchmarks.

Benchmark	Initial shape	Material	Viscosity	Density (kg/m^3)
HoneyCone	Cone	Principled BSDF	0.8	1420
WaterCube	Cube	Glass BSDF	0.08	1000
WaterSphere	Sphere	Glass BSDF	0.08	1000

the components in $\boldsymbol{e}_x$ and $\boldsymbol{e}_d$ are same as those for $\boldsymbol{x}$ and $\boldsymbol{d}$ respectively. We adopt the Adam optimizer for training. In the warm-up phase of $\mathcal{R}_\phi$, it is trained for 60k steps with a fixed learning rate of $5e^-4$. In the joint training stage, $\mathcal{T}_\theta$ and $\mathcal{R}_\phi$ are trained for 450k steps. The initial learning rate of $\mathcal{T}_\theta$ is set as $5e^{-4}$ and is decayed by 0.5 after 50k and 200k steps. The initial learning rate is set as $1e^{-6}$ and is decayed by 0.5 after 10k, 30k, 50k, and 100k steps.

4 Experiments

4.1 Experimental Setup

The effect of fluid dynamics grounding is evaluated from three aspects: (i) distances between the estimated and the ground-truth particles (Section 4.2); (ii) novel view synthesis (Section 4.3); (iii) future prediction (Section 4.4).

Benchmarks. We train NeuroFluid on fluids generated by DFSPH [Bender and Koschier, 2015] and Blender [Community, 2018]. DFSPH can simulate complex fluid dynamics while preserving low volume compression features. Blender is used to render the simulation results with different materials to produce high-fidelity and realistic images. We generate three benchmarks, named "HoneyCone", "WaterCube", and "WaterSphere", and simulate them for 50 time steps. The benchmarks have different properties of initial fluid shapes, materials, viscosity, and densities, as shown in Table 1. We simulate the fluids with various bodies for 2 seconds. Since the fluids become stable over time, we use 50 consecutive frames with significant physical dynamics for both training and test data. We use a cube box as the common practice for the fluid container.

Evaluation metrics. Following [Ummenhofer et al., 2019], we use the Pred2GT distance to measure the accuracy of estimated particles, which is formulated as $d = \frac{1}{N}\sum_{i=1}^{N} \min_{\mathbf{p}_i \in \boldsymbol{P}_t} \|\mathbf{p}_i - \widetilde{\mathbf{p}}_i\|_2$, where $\boldsymbol{P}_t$ is the set of estimated particles at a specific time step, and $\widetilde{\mathbf{p}}_i$ is the ground-truth particle for $\mathbf{p}_i$. We report results in terms of (i) d-avg: the average Pred2GT distance of the whole sequence; (ii) d-end: the distance at the end time-step. To evaluate the rendering quality, we use PSNR/SSIM [Wang et al., 2004] (higher is better) and LPIPS [Zhang et al., 2018] (lower is better) as the metrics, following [Mildenhall et al., 2020].

Compared methods. To evaluate the accuracy of estimated fluid dynamics, NeuroFluid is compared with a pre-trained DLF [Ummenhofer et al., 2019] and its variants fine-tuned on our benchmarks using true particle states. To evaluate the rendering quality, it is compared with

- **NeRF** [Mildenhall et al., 2020]: It represents a static scene as 5D radiance fields of locations and view directions.
- **D-NeRF** [Pumarola et al., 2021]: It extends NeRF to dynamic scenes with a deformation network that estimates 3D transitions $\Psi : (\boldsymbol{x}, t) \rightarrow \Delta\boldsymbol{x}$ in a canonical space.
- **NeRF-T**: Referring to D-NeRF, NeRF can be extended on dynamic scenes represented with an additional time input, deriving a 6D radiance field of $(\boldsymbol{x}, \boldsymbol{d}, t)$.
- **3D-aware model from** [Li et al., 2022]: [Li et al., 2022] used an image encoder to learn latent fluid state, and fed the latent state to a NeRF to render fluids. Unlike NeuroFluid, it is not based on particles and learns the dynamics model and NeRF separately.

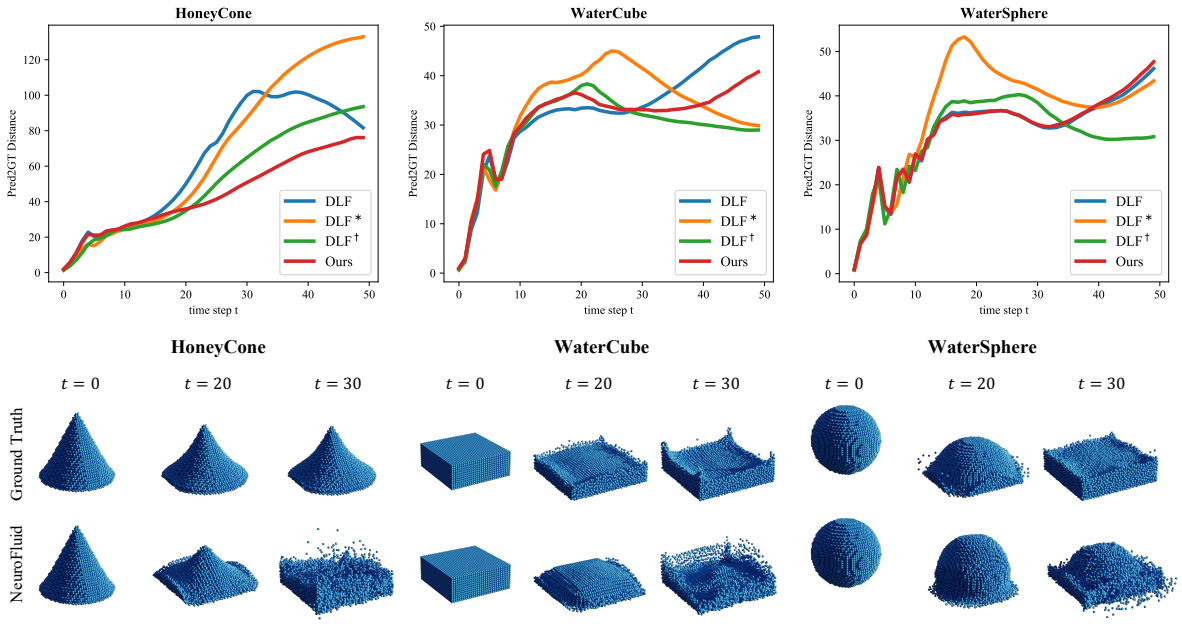

Figure 3: Fluid dynamics grounding results of NeuroFluid on 3 benchmarks. **Top**: Pred2GT distances over 50 time steps of the learned transition model. Note that DLF† and DLF* are trained with ground-truth particles, rather than grounding particles from visual observations. **Bottom**: Visualization of ground-truth and estimated fluid particles by the particle transition model.

Table 2: Novel view synthesis evaluation averaged on two novel views.

	WaterCube			WaterSphere			HoneyCone		
Method	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
NeRF	24.21	0.91	0.19	21.55	0.92	0.26	**24.33**	0.94	0.18
NeRF-T	25.09	0.91	0.24	24.10	0.92	0.24	24.81	0.91	0.22
D-NeRF	28.23	0.94	0.15	24.88	0.89	0.25	23.63	0.92	0.21
[Li et al., 2022]	23.93	0.93	0.15	19.58	0.91	0.19	21.25	0.94	0.13
NeuroFluid	**30.06**	**0.95**	**0.10**	**28.50**	**0.93**	**0.13**	24.25	**0.96**	**0.11**

4.2 Performance on Fluid Dynamics Grounding

In this section, we compare with DLF in terms of Pred2Gt distance. DLF has the same structure as our transition model but is trained under the supervision of ground-truth particles. By comparing with DLF, we can examine whether the image observations can encourage the transition model to learn reasonable fluid dynamics. For a fair comparison, in addition to comparing with the pretrained DLF, we also finetune DLF with two settings: (1) finetuning DLF on the data generated from our simulator, termed as DLF*. The generation process covers the setting used in our benchmarks, except for the initial fluid shape is unknown. (2) finetuning DLF on our benchmark, termed as DLF†.

The top row of Fig 3 plots the Pred2GT distance for each time step on the used benchmarks. From this sub-figure, we observe that all models behave similarly at the first 20 time steps. At the middle time steps, the estimation errors increased since the fluid flows more rapidly. However, we observe that the estimation error of DLF models, especially DLF*, will increase larger than our transition model. This implies that visual supervision is more robust than particle supervision in terms of learning complex fluid dynamics. The lower of Fig 3 visualizes the estimated particles by NeuroFluid, from which we can observe reasonable particle positions and natural dynamics.

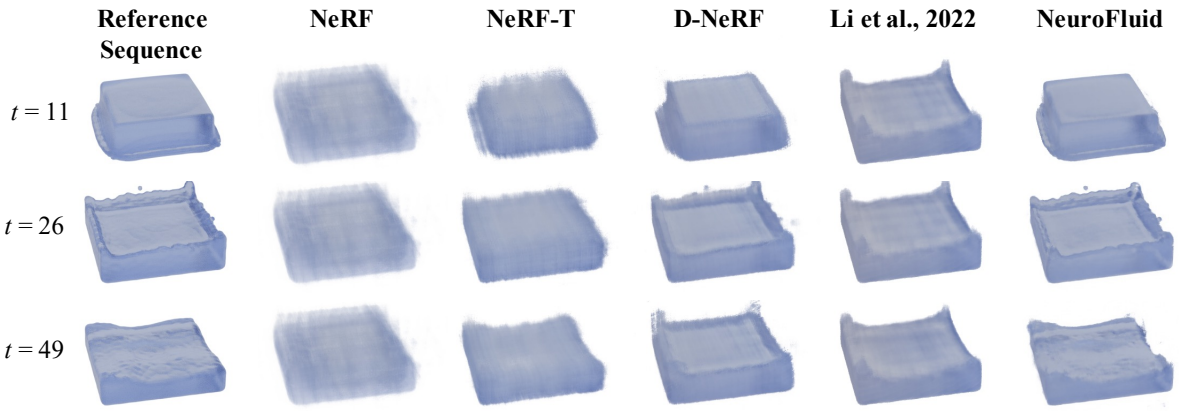

Figure 4: Novel view synthesis on the WaterCube sequence. The first column indicates the reference sequential images in the novel view. NeuroFluid significantly outperforms all of the compared methods. It demonstrates (i) the effect of fluid dynamics modeling based on particle representations; (ii) the effectiveness of PhysNeRF for rendering the rapidly changing details of fluid dynamic scenes.

4.3 Performance on Novel View Synthesis

We prove that NeuroFluid is able to ground accurate particles positions by checking the novel view synthesis. The more accurate particles are estimated, the more clear and realistic the synthesized images should be. Since we pretrained PhysNeRF on the initial scene with observations from 4 views, we also apply this scheme to compared methods for a fair comparison.

Figure 4 shows a quantitative comparison of the WaterCube sequence. Due to the input is not time-dependent, the results of NeRF are nearly static and blurry. By simply encoding time t as input (NeRF-T), learning spatial deformation field (D-NeRF), or learning temporal latent representations, the other compared methods can synthesize images with temporal variant texture. However, the artifacts of blurry textures still exist. Particularly, the misaligned fluid shape implies that these methods don't accurately capture the 3D geometry of the fluid. In contrast, thanks to the neighborhood encoding introducing particle-based guidance, NeuroFluid gives photorealistic novel views, which is highly consistent with the reference images.

Table 2 summarizes the qualitative results on the three fluid benchmarks. NeuroFluid has overall higher performance in terms of adopted metrics. Compared to NeRF, NeuroFluid decreases the LPIPS about 50% in all benchmarks, such as from 0.19 to 0.10 on WaterCube. As introduced in [Zhang et al., 2018], LPIPS is more in line with human perception. This indicates that the synthesized images of NeuroFluid gain significant improvement in realism.

4.4 Performance on Future Prediction

We conduct forward model prediction for 10 steps and evaluate the results to assess whether the transition model has learned intrinsic physical transition principles. For a comprehensive study, both the accuracy of predicted particles and the quality of synthesized images are evaluated and reported in Table 3 and Table 4. In Table 3, in addition to d-avg and d-end, we also calculate the accumulated Pred2GT distance at 5-th time step, termed as d-5. Overall, NeuroFluid performs better than other compared methods. The lowest d-avg and d-5 values of NeuroFluid verify its accuracy and stability in terms of particle transition.

In addition, we conduct forward prediction on NeRF-T, D-NeRF and [Li et al., 2022], and then qualitatively compare with them. Table 4 given the comparison results. NeuroFluid clearly outperforms compared methods for a large margin. This verifies the predicted particles are highly accurate, so that NeuroFluid can render photorealistic images.

Table 3: Fluid simulation accuracy by rolling-out the transition model for the next 10 time steps into the future (HoneyCone).

Methods	d-avg	d-5	d-end
DLF	71.43	76.99	**61.02**
DLF*	133.53	133.75	132.26
DLF†	94.53	94.43	93.64
NeuroFluid	**64.31**	**74.37**	80.66

Table 4: Image rendering results for the next 10 time steps (HoneyCone). For NeuroFluid, we predict particle positions from $t = 1$ and evaluate the results at $t \in [50, 59]$.

Method	PSNR↑	SSIM↑	LPIPS↓
NeRF-T	24.13	0.93	0.14
D-NeRF	20.64	0.91	0.18
[Li et al., 2022]	26.23	0.94	0.13
NeuroFluid	**27.59**	**0.96**	**0.09**

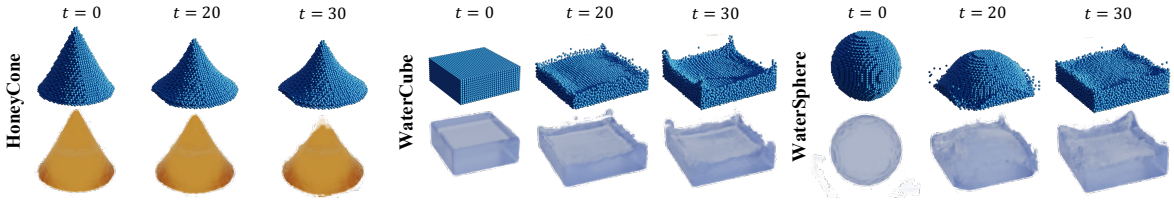

Figure 5: Rendering results of PhysNeRF conditioned on ground-truth particles positions, showing that PhysNeRF effectively renders the details in lighting, geometries, and deformations of fluids.

4.5 Model Analysis

Novel particle dynamics rendering. One of our contributions is using neighborhood encoding to correlate the particles with the neural radiance field. To verify that PhysNeRF really learns particle-related features, we conduct the novel particle dynamics rendering experiments. Specifically, given a sequence of particles with novel dynamics, we study whether PhysNeRF can render corresponding images. The results is shown in Figure 5. The 1-st, 3-rd, and 5-th rows are the input particles. From Figure 5, we can observe that the rendered images have a clear and detailed appearance. This indicates that PhysNeRF learned representation related to intrinsic particles dynamics, and can synthesize a new image driven by novel particles.

Joint Training *vs.* Separate Training. Recall that in Section 3.3, we first pre-train PhysNeRF $\mathcal{R}_\phi$ and then jointly train $\mathcal{R}_\phi$ and the transition model $\mathcal{T}_\theta$. A natural question is that how about fixing $\mathcal{R}_\phi$ and individually training $\mathcal{T}_\theta$. Here we study the influence of the two training strategies. First, both the average GT2Pred distance on the training sequence and the roll-out sequence are reported in Table 5. From Table 5, we observe an obvious decrease in terms of d-avg. The quantitative comparison results in Figure 6 show that the model trained jointly preserves more clear and realistic appearance details. Both experimental results indicate that joint training is better than the separate training scheme. The possible reason is that accurate particle is beneficial for $\mathcal{R}_\phi$ to better capture geometric information, and the better rendering quality also can improve the learning of $\mathcal{T}_\theta$. Jointly training $\mathcal{R}_\phi$ and $\mathcal{T}_\theta$ with an appropriate hyper-parameter setting can form a self-boosting loop.

Table 5: Quantitative evaluation on fluid dynamics grounding. We compare the end-to-end training approach and the one with a fixed PhysNeRF.

Method	d-avg	d-avg-rollout
Joint training	**30.90**	**43.47**
Separate training	34.07	54.64

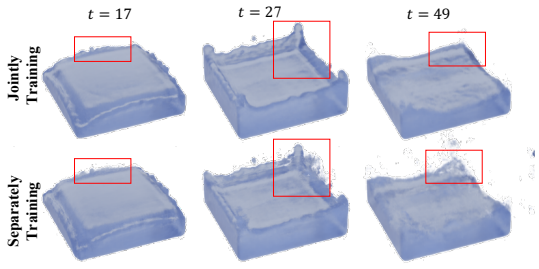

Figure 6: The ablation study of NeuroFluid on joint training *vs.* separate training.

5 Related Work

Intuitive physics. Recent work in intuitive physics and inverse graphics has attempted to build neural models that can reason about 3D structures, stability, collisions, forces, and velocities from images or videos [Battaglia et al., 2013, Wu et al., 2017a, Wu et al., 2017b, Nguyen-Phuoc et al., 2018, Han et al., 2020, Chen et al., 2021]. The most relevant work to fluid dynamics grounding is presented in [Li et al., 2022]. The difference, however, is that the model learns fluid dynamics on latent states encoded from visual observations, whereas our model infers the fluid dynamics in explicit particle space.

Particle-based fluid simulation. Fluid simulation is a long-standing research field in fluid mechanics and computer science [Chorin, 1968, Stam, 1999, Chentanez and Müller, 2013, Sulsky et al., 1995, Zhu and Bridson, 2005, Stomakhin et al., 2013, Jiang et al., 2015, Fang et al., 2020]. Particle-based methods represent fluid dynamics in Lagrangian descriptions through the transitions and interactions of a group of particles [Monaghan, 1992, Müller et al., 2003, Price, 2012]. They can naturally simulate complex collisions and be easily integrated into interactive physical environments [Müller et al., 2003, Amada et al., 2004] and differentiable simulators [Hu et al., 2019, Holl et al., 2020]. Learning fluid simulators from data greatly improves the efficiency of predicting complex fluid dynamics [Müller et al., 1999, Poloni et al., 2000, Ladický et al., 2015]. Recent advances in deep learning typically use graph networks [Li et al., 2018b, Mrowca et al., 2018, Sanchez-Gonzalez et al., 2020, Li et al., 2020] or modified convolution operators [Schenck and Fox, 2018, Ummenhofer et al., 2019, Kim et al., 2019] to simulate particle state transitions and interactions in local neighborhoods. However, these models need to be trained with consecutive true particle states, rather than reasoning about fluid dynamics directly from observed images, as NeuroFluid does.

Differentiable rendering. Our fluid renderer is designed on top of NeRF [Mildenhall et al., 2020], a technique that represents 3D scenes as learnable continuous functions. Different from other neural renderers, including mesh-based rasterizators [Loper and Black, 2014, Kato et al., 2018, Liu et al., 2019] and ray tracers [Li et al., 2018a], we find that NeRF is more compatible with particle-based fluid representations. Existing approaches extend NeRF to deformed scenes with strong inductive biases of canonical displacement [Pumarola et al., 2021, Park et al., 2021], scene flow warping [Li et al., 2021], or even 3D human meshes and poses [Peng et al., 2021, Liu et al., 2021, Su et al., 2021], which do not perfectly match the geometric properties of particle systems. In contrast, PhysNeRF is a particle-driven neural renderer and thus can facilitate fluid dynamics grounding through end-to-end optimization.

6 Conclusions and Future Directions

We studied a new research problem named fluid dynamics grounding, in which models are learned to reason about the underlying physical dynamics of the particle systems of fluids from sequential visual observations. We proposed NeuroFluid, a differentiable deep learning framework that learns the 3D structures and physical dynamics of fluids by connecting particle-based dynamics modeling and particle-driven neural rendering. The renderer learns to consider the geometric properties of fluid particles in volume rendering functions. It back-propagates the reconstruction errors between the synthesized and the observed images, thus allowing the transition model in NeuroFluid to ground physical dynamics from visual observations. NeuroFluid was evaluated on tasks of fluid dynamics grounding, novel view synthesis, and future dynamics prediction. An unsolved problem of this work is the error accumulation of particle transitions, which may require more effective and physics-informed constraints defined in particle space.

Acknowledgement

This work was supported by NSFC (U19B2035, 62106144), Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102), and Shanghai Sailing Program (21Z510202133) from the Science and Technology Commission of Shanghai Municipality.

References

[Amada et al., 2004] Amada, T., Imura, M., Yasumuro, Y., Manabe, Y., and Chihara, K. (2004). Particle-based fluid simulation on gpu. In *ACM workshop on general-purpose computing on graphics processors*, volume 41, page 42. Citeseer.

[Battaglia et al., 2013] Battaglia, P. W., Hamrick, J. B., and Tenenbaum, J. B. (2013). Simulation as an engine of physical scene understanding. *PNAS*, 110(45):18327–18332.

[Bender and Koschier, 2015] Bender, J. and Koschier, D. (2015). Divergence-free smoothed particle hydrodynamics. In *SCA*, pages 147–155.

[Biedert et al., 2018] Biedert, T., Sohns, J.-T., Schröder, S., Amstutz, J., Wald, I., and Garth, C. (2018). Direct raytracing of particle-based fluid surfaces using anisotropic kernels. In *EGPGV*, pages 1–12.

[Chen et al., 2021] Chen, W., Litalien, J., Gao, J., Wang, Z., Fuji Tsang, C., Khamis, S., Litany, O., and Fidler, S. (2021). DIB-R++: Learning to predict lighting and material with a hybrid differentiable renderer. In *NeurIPS*.

[Chentanez and Müller, 2013] Chentanez, N. and Müller, M. (2013). Mass-conserving eulerian liquid simulation. *TVCG*, 20(1):17–29.

[Chorin, 1968] Chorin, A. J. (1968). Numerical solution of the Navier-Stokes equations. *Mathematics of computation*, 22(104):745–762.

[Community, 2018] Community, B. O. (2018). *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam.

[Fang et al., 2020] Fang, Y., Qu, Z., Li, M., Zhang, X., Zhu, Y., Aanjaneya, M., and Jiang, C. (2020). IQ-MPM: an interface quadrature material point method for non-sticky strongly two-way coupled nonlinear solids and fluids. *ACM Trans. Graph.*, 39(4):51–1.

[Han et al., 2020] Han, Z., Chen, C., Liu, Y.-S., and Zwicker, M. (2020). DRWR: A differentiable renderer without rendering for unsupervised 3D structure learning from silhouette images. In *ICML*.

[Holl et al., 2020] Holl, P., Koltun, V., and Thuerey, N. (2020). Learning to control pdes with differentiable physics. *arXiv preprint arXiv:2001.07457*.

[Hu et al., 2019] Hu, Y., Anderson, L., Li, T.-M., Sun, Q., Carr, N., Ragan-Kelley, J., and Durand, F. (2019). DiffTaichi: Differentiable programming for physical simulation. *arXiv preprint arXiv:1910.00935*.

[Jiang et al., 2015] Jiang, C., Schroeder, C., Selle, A., Teran, J., and Stomakhin, A. (2015). The affine particle-in-cell method. *ACM Trans. Graph.*, 34(4):1–10.

[Kajiya and Von Herzen, 1984] Kajiya, J. T. and Von Herzen, B. P. (1984). Ray tracing volume densities. *ACM SIGGRAPH computer graphics*, 18(3):165–174.

[Kato et al., 2018] Kato, H., Ushiku, Y., and Harada, T. (2018). Neural 3d mesh renderer. In *CVPR*, pages 3907–3916.

[Kim et al., 2019] Kim, B., Azevedo, V. C., Thuerey, N., Kim, T., Gross, M., and Solenthaler, B. (2019). Deep fluids: A generative network for parameterized fluid simulations. In *Comput Graph Forum*, pages 59–70.

[Ladickỳ et al., 2015] Ladickỳ, L., Jeong, S., Solenthaler, B., Pollefeys, M., and Gross, M. (2015). Data-driven fluid simulations using regression forests. *ACM Trans. Graph.*, 34(6):1–9.

[Li et al., 2018a] Li, T.-M., Aittala, M., Durand, F., and Lehtinen, J. (2018a). Differentiable monte carlo ray tracing through edge sampling. *ACM Trans. Graph.*, 37(6):1–11.

[Li et al., 2022] Li, Y., Li, S., Sitzmann, V., Agrawal, P., and Torralba, A. (2022). 3d neural scene representations for visuomotor control. In *CoRL*, pages 112–123.

[Li et al., 2020] Li, Y., Lin, T., Yi, K., Bear, D., Yamins, D., Wu, J., Tenenbaum, J., and Torralba, A. (2020). Visual grounding of learned physical models. In *ICML*, pages 5927–5936.

[Li et al., 2018b] Li, Y., Wu, J., Tedrake, R., Tenenbaum, J. B., and Torralba, A. (2018b). Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. In *ICLR*.

[Li et al., 2021] Li, Z., Niklaus, S., Snavely, N., and Wang, O. (2021). Neural scene flow fields for space-time view synthesis of dynamic scenes. In *CVPR*.

[Liu et al., 2021] Liu, L., Habermann, M., Rudnev, V., Sarkar, K., Gu, J., and Theobalt, C. (2021). Neural actor: Neural free-view synthesis of human actors with pose control. *ACM SIGGRAPH Asia*.

[Liu et al., 2019] Liu, S., Li, T., Chen, W., and Li, H. (2019). Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *ICCV*, pages 7708–7717.

[Loper and Black, 2014] Loper, M. M. and Black, M. J. (2014). OpenDR: An approximate differentiable renderer. In *ECCV*, pages 154–169.

[Mildenhall et al., 2020] Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. (2020). NeRF: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, pages 405–421. Springer.

[Monaghan, 1992] Monaghan, J. J. (1992). Smoothed particle hydrodynamics. *Annu. Rev. Astron. Astrophys.*, 30(1):543–574.

[Mrowca et al., 2018] Mrowca, D., Zhuang, C., Wang, E., Haber, N., Fei-Fei, L., Tenenbaum, J. B., and Yamins, D. L. (2018). Flexible neural representation for physics prediction. In *NeurIPS*, pages 8813–8824.

[Müller et al., 2003] Müller, M., Charypar, D., and Gross, M. H. (2003). Particle-based fluid simulation for interactive applications. In *SCA*, pages 154–159.

[Müller et al., 1999] Müller, S., Milano, M., and Koumoutsakos, P. (1999). Application of machine learning algorithms to flow modeling and optimization. *Annual Research Briefs*, pages 169–178.

[Nguyen-Phuoc et al., 2018] Nguyen-Phuoc, T., Li, C., Balaban, S., and Yang, Y.-L. (2018). RenderNet: A deep convolutional network for differentiable rendering from 3d shapes. In *NeurIPS*.

[Park et al., 2021] Park, K., Sinha, U., Barron, J. T., Bouaziz, S., Goldman, D. B., Seitz, S. M., and Martin-Brualla, R. (2021). Nerfies: Deformable neural radiance fields. In *ICCV*, pages 5865–5874.

[Peng et al., 2021] Peng, S., Dong, J., Wang, Q., Zhang, S., Shuai, Q., Zhou, X., and Bao, H. (2021). Animatable neural radiance fields for modeling dynamic human bodies. In *ICCV*, pages 14314–14323.

[Poloni et al., 2000] Poloni, C., Giurgevich, A., Onesti, L., and Pediroda, V. (2000). Hybridization of a multi-objective genetic algorithm, a neural network and a classical optimizer for a complex design problem in fluid dynamics. *Comput Methods Appl Mech Eng*, 186(2-4):403–420.

[Price, 2012] Price, D. J. (2012). Smoothed particle hydrodynamics and magnetohydrodynamics. *J. Comput. Phys.*, 231(3):759–794.

[Pumarola et al., 2021] Pumarola, A., Corona, E., Pons-Moll, G., and Moreno-Noguer, F. (2021). D-NeRF: Neural radiance fields for dynamic scenes. In *CVPR*, pages 10318–10327.

[Qi et al., 2017] Qi, C. R., Yi, L., Su, H., and Guibas, L. J. (2017). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *NeurIPS*, 30.

[Sanchez-Gonzalez et al., 2020] Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., and Battaglia, P. (2020). Learning to simulate complex physics with graph networks. In *ICML*, pages 8459–8468.

[Schenck and Fox, 2018] Schenck, C. and Fox, D. (2018). SPNets: Differentiable fluid dynamics for deep neural networks. In *CoRL*, pages 317–335.

[Stam, 1999] Stam, J. (1999). Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 121–128.

[Stomakhin et al., 2013] Stomakhin, A., Schroeder, C., Chai, L., Teran, J., and Selle, A. (2013). A material point method for snow simulation. *ACM Trans. Graph.*, 32(4):1–10.

[Su et al., 2021] Su, S.-Y., Yu, F., Zollhöfer, M., and Rhodin, H. (2021). A-NeRF: Articulated neural radiance fields for learning human shape, appearance, and pose. In *NeurIPS*.

[Sulsky et al., 1995] Sulsky, D., Zhou, S.-J., and Schreyer, H. L. (1995). Application of a particle-in-cell method to solid mechanics. *Comput Phys Commun*, 87(1-2):236–252.

[Ummenhofer et al., 2019] Ummenhofer, B., Prantl, L., Thuerey, N., and Koltun, V. (2019). Lagrangian fluid simulation with continuous convolutions. In *ICLR*.

[Wang et al., 2004] Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. *TIP*.

[Wu et al., 2017a] Wu, J., Lu, E., Kohli, P., Freeman, B., and Tenenbaum, J. (2017a). Learning to see physics via visual de-animation. In *NeurIPS*, volume 30, pages 153–164.

[Wu et al., 2017b] Wu, J., Wang, Y., Xue, T., Sun, X., Freeman, W. T., and Tenenbaum, J. B. (2017b). MarrNet: 3d shape reconstruction via 2.5d sketches. In *NeurIPS*.

[Zhang et al., 2018] Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. (2018). The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, pages 586–595.

[Zhu and Bridson, 2005] Zhu, Y. and Bridson, R. (2005). Animating sand as a fluid. *ACM Trans. Graph.*, 24(3):965–972.