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

Shanyan Guan†, Huayu Deng†, Yunbo Wang†, Xiaokang Yang†

Abstract

Deep learning has shown great potential for modeling the physical dynamics of complex particle systems such as fluids. 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.

1. Introduction

Intuitive physics or intuitive physical inference is a research area that attracts 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 understand the underlying physics, the key is to solve the inverse problem of synthesizing the visual scenes. A typical forward modeling process includes two steps: fluid simulation and dynamic scene rendering.

Despite recent progress in learning particle-based fluid simulators (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 this problem, in which a key component is the particle-driven neural renderer named PhysNeRF.

![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.](image-url)
NeuroFluid: Fluid Dynamics Grounding with Particle-Driven Neural Radiance Fields

We solve the problem of grounding the physical dynamics of particle-based fluid systems from a sequence of visual observations \( \{ I^d_{t=1:T} \} \) collected from a sparse set of views \( \{ d \} \). The dynamics can be represented by a particle state transition model \( T_\theta \). Given the initial particles state \( s_0 \), \( T_\theta \) estimates the transitions of particle states from \( s_t \) to \( s_{t+1} \) with probability \( p(s_{t+1}|s_t; \theta) \). Unlike existing learning-based fluid simulators (Li et al., 2018b; Mrowca et al., 2018; Sanchez-Gonzalez et al., 2020; Li et al., 2020; Schenck & Fox, 2018; Ummenhofer et al., 2019; Kim et al., 2019), in our problem setup, the parameters \( \theta \) CANNOT be optimized with ground-truth particle states.

Instead, we receive a new observation \( I^d_{t+1} \) 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 \( R_\phi \) to synthesize them. The forward modeling process of fluid dynamics grounding can be formulated as:

\[
s_{t+1} \leftarrow T_\theta(s_t), \\
\hat{I}^d_{t+1} \leftarrow R_\phi(s_{t+1}, d),
\]

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

\[
\arg \max_{\theta, \phi} \sum_{t,d} \log \left( p(I^d_{t+1}|s_{t+1},d; \phi) p(s_{t+1}|s_t; \theta) \right). \tag{2}
\]

We evaluate \( T_\theta \) by (i) measuring the distance between the optimized particle positions \( \{ P_t \}_{1:T} \) and the true positions \( \{ P^*_t \}_{1:T} \). Further, it can be partly assessed through forward modeling of (ii) novel view synthesis and (iii) rolling-out \( T_\theta \) 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 \( \theta \) and \( \phi \) 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 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

Why particle-based representations? Particle-based representations facilitate fluid dynamics grounding. Fluid representation techniques have been fully explored in previous literature (Ummenhofer et al., 2019; Li et al., 2019), 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 \( (T_\theta) \) receive the gradients generated by the error of the rendering results, allowing \( T_\theta \) to gradually approach the real fluid dynamics.
In practice, we explicitly specify the particle states $s_t$ in Eq. (1) and Eq. (2) as the particle positions $P_t$ and velocities $V_t$ over the entire particle set.

**Particle transition model.** In forward modeling, the particle transition model $T_\theta$ learns the movement and interactions of particles between consecutive time steps. Starting with the initial particle states $(P_0, V_0)$, which are known during training and testing phases, it rolls-out over time to estimate a sequence of particle states $\{(P_t, V_t)\}_{1:T}$. At each time step, $(P_t, V_t)$ 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 network 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 rendering function (Kajiya & Von Herzen, 1984):

$$\text{RGB value } c \text{ and the view-dependent emitted color } e \text{ trained to map the physical scene. Next, a multilayer perceptron (MLP) is used to predict how to encode the neighboring particles in small neighborhoods of 3D space, therefore, we parameterize the physical dynamics of the fluid. This appropriate way, in our case, is the particle-driven neural rendering fields.}

**3.2. Particle-Driven Neural Radiance Fields**

**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 $o$ along the view direction $d$, which can be denoted by $r = o + nd$. Then, $N$ points with 3D coordinates $(x_i)_{1:N}$ are sampled along $r$ between the near and far bounds of the physical scene. Next, a multilayer perceptron (MLP) is trained to map $(x_i, d)$ to the intrinsic volume density $\sigma(x_i)$ and the view-dependent emitted color $c(x_i, d)$. Finally, the RGB value $C(r)$ is decided using a classic volume rendering function (Kajiya & Von Herzen, 1984):

$$C(r) = \sum_{i=1}^{N} T_i \left( 1 - \exp \left( - \sigma(x_i) \delta_j \right) \right) c(x_i, d),$$

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

**PhysNeRF: Linking 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 $x$, PhysNeRF first conducts ball query (Qi et al., 2017) to search its neighboring fluid particles $P^x \in P_t$:

$$P^x = S(\|p_i - x\|_2, r_s),$$

where $p_i$ refers to the $i$-th particle in $P_t$, produced by $T_\theta$ 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 $P^x$ in its 3D neighborhood. Therefore, we parameterize $P^x$ to view-independent encoding $e_x$ and view-dependent encoding $e_d$ that can be written as follows:

$$(e_x, e_d) = \mathcal{E}(P^x, x).$$

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 $(e_x, e_d)$ and $(x, d)$ to volume density and emitted color. The rendering mechanism of PhysNeRF can be formulated as follows:

$$(\sigma(x, e_x), c(x, e_x, d, e_d)) = \mathcal{R}_\phi(x, d, e_x, e_d).$$

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 $P_t$. It is the unique property of PhysNeRF that makes it different from all existing NeRF-based neural renderers.

**Neighborhood encoding in PhysNeRF.** We here introduce how to encode the neighboring particles $P^x$ to $(e_x, e_d)$. As shown in Figure 2, we first define $p_c$ as the weighted average position over $P^x$:

$$p_c = \frac{1}{K} \sum_i w_i p_i, \quad p_i \in P^x,$$
Neighborhood Encoding

Figure 2. Overview of PhysNeRF, a key component in NeuroFluid. Left: In PhysNeRF, 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 $e_x$ and view-dependent particle encoding $e_d$ to estimate the volume density $\sigma$ and the emitted color $c$.

3.3. Optimizing NeuroFluid

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

$$\mathcal{L}(\widehat{I}_d^t, I_d^t) = \sum_{u,v} \left\| \widehat{I}_d^t(u,v) - I_d^t(u,v) \right\|^2_2,$$  \quad (11)

where $I_d^t(u,v)$ is typically represented as $C(r)$ 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 $T_0$ and $R_0$ from a cold start makes the optimization prone to collapse to some undesirable local minima. Therefore, we first warm-up PhysNeRF on the initial fluid scene with a sparse set of $N_d$ views conditioned on $\{P_0, V_0, \theta, \phi\}$ to obtain an effective initialization of the renderer.

The objective function in the warm-up phase is

$$\min_{\phi} \sum_d \mathcal{L}(\widehat{I}_d^t, I_d^t; P_0, V_0, \theta, \phi).$$  \quad (12)

After warming-up PhysNeRF, it roughly learns the correlation between view-dependent observations and particle geometries, we then jointly train $T_0$ and $R_0$ on $\{I_d^t\}_{t=1:T}$, collected from a static camera with a fixed direction of $d$. The objective function in this end-to-end training phase is

$$\min_{\theta, \phi} \sum_{t=1}^{T} \mathcal{L}(\widehat{I}_d^t, I_d^t; P_0, V_0, \theta, \phi).$$  \quad (13)

Implementation and training details. The model architecture of PhysNeRF is the same as NeRF, except that it takes as input $(e_x, e_d)$. We adopt the hierarchical sampling strategy to sample 64 points coarsely and then sample 128 points finely along each ray. The positional encoding parameters for $e_x$ and $e_d$ are the same as those for $x$ and $d$. NeuroFluid are trained with the Adam optimizer (Kingma & Ba, 2015). In the warm-up phase of $R_0$, it is trained for 100k steps with an initial learning rate of $5e^{-4}$, which
is then exponentially decayed with $\gamma = 0.1$. In the joint training phase, the entire model is trained for 500k steps. The initial learning rate of $T_\theta$ and $R_\phi$ are set to $1e^{-6}$ and $5e^{-4}$ respectively. The learning rate of $T_\theta$ is decayed by 0.5 after 10k, 30k, 50k, 100k, and 300k steps. That of $R_\phi$ is decayed by 0.5 after 10k, 75k, and 150k steps. To ease the convergence, we initialize $T_\theta$ with the released DLF model that is pre-trained on other benchmarks. The resolution of the observed image is $400 \times 400$.

4. Experiments

4.1. Experimental Setup

The effect of fluid dynamics grounding is evaluated from three aspects: distances between the grounded and true particle positions (Section 4.2), distances between the predicted particles and the ground truth in a future time horizon (Section 4.3), and the quality of novel view synthesis (Section 4.4).

Benchmarks. We train NeuroFluid on visual observations of fluids generated by DFSPH (Bender & Koschier, 2015) and Blender (Community, 2018). DFSPH is a particle-based physics engine that can simulate complex fluid dynamics. Blender is used to render the simulated particles with different materials to produce high-fidelity and realistic images. We generate four benchmarks named “HoneyCone”, “WaterCube”, “WaterSphere”, and “WaterBunny”. We simulate the fluid dynamics falling inside a cube box for 60 time steps. As shown in Table 1, different benchmarks vary in the initial shape, material, viscosity, and density of the fluid. For HoneyCone, WaterCube, and WaterSphere, we use the visual observations during the first 50 time steps as the training data. In particular, we use WaterBunny to evaluate the generalization ability of PhysNeRF, which is pre-trained on the first three benchmarks and evaluated on novel particle distributions of StanfordBunny for image synthesis.

Evaluation metrics. Following existing fluid simulation approaches (Ummenhofer et al., 2019), we use the Euclidean distance to measure the performance of particle grounding and future prediction, which is calculated by $d = \frac{1}{N} \sum_{i=1}^{N} \min_{\tilde{p}_i \in P_i} ||p_i - \tilde{p}_i||_2$, where $P_i$ is the set of estimated particles at a specific time step, and $\tilde{p}_i$ is the ground-truth particle position for $p_i$. To evaluate the quality of novel view synthesis, we follow the original NeRF (Mildenhall et al., 2020) to use PSNR, SSIM (Wang et al., 2004), and LPIPS (Zhang et al., 2018) as the metrics. Higher PSNR and SSIM values and a lower LPIPS indicate better rendering of fluid scenes.

Compared methods for particle grounding and future prediction. Since NeuroFluid is a pilot study for inferring fluid dynamics from visual observations alone, it is difficult to make a completely fair comparison with existing approaches. Therefore, we make a compromise by comparing it with two baseline models for fluid simulation. Like the particle transition model in NeuroFluid, they follow the same DLF architecture (Ummenhofer et al., 2019). Given the initial particle positions and velocities, DLF uses a continuous convolution network that is performed in 3D space to simulate the particle transitions. However, these two baseline models are pre-trained in different scenarios:

- **DLF**: This model has the same network parameters as the initialization of the particle transition model in NeuroFluid. It is well-trained on the benchmarks used in (Ummenhofer et al., 2019), supervised with true particle positions. Although DLF has shown the generalization ability to a wide variety of fluid shapes, materials, and environments, in NeuroFluid, it can be further improved by learning from visual observations of the fluid surface.

- **DLF**: To provide DLF with stronger supervisions, we finetune the above DLF model with true particle states on the evaluation benchmarks in Table 1. The finetuning stage lasts for 50 epochs with a learning rate of $1e^{-6}$.

Note that although VGPL (Li et al., 2020) can also infer particles positions from visual observation, it requires a well-trained and fixed particle transition model. In other words, unlike NeuroFluid, the goal of VGPL is not to learn the underlying dynamics from the observed fluid scenes.

Compared methods for novel view synthesis. To demonstrate that NeuroFluid obtains excellent rendering effects for fluid scenes by explicitly considering the particle distributions, we compare it with the NeRF-based approaches, including D-NeRF (Pumarola et al., 2021), NeRF-T, and the 3D-aware fluid renderer from Li et al. (2022):
Table 2. Quantitative results on the errors of fluid dynamics grounding \((t < 50)\) and prediction \((50 \leq t < 60)\), which are calculated between the grounded/predicted particle positions and the ground truth provided by the fluid simulator (lower is better). For DLF\(^†\), the transition model is finetuned on the testing benchmarks in a fully supervised way, that is, using true particle positions at \(t < 50\).

| METHOD  | WATERCUBE |  | WATERSPHERE |  | HONEYCONE |  |
|---------|-----------|---|-------------|---|-----------|---|
|         | GROUNDING | PREDICTION | GROUNDING | PREDICTION | GROUNDING | PREDICTION |
|         | \(d_{t<50}\) | \(dt=49\) | \(d_{t>50}\) | \(dt=49\) | \(d_{t<50}\) | \(dt=49\) |
| DLF     | 32.3      | 48.3      | 47.4      | 46.2      | 32.2      | 47.6      |
| NEUROFLUID | 28.8      | 34.9      | 35.5      | 36.7      | 31.1      | 31.5      |
| DLF\(^†\) | 30.0      | 28.5      | 30.0      | 31.8      | 34.3      | 66.1      |

Figure 3. Qualitative results of fluid dynamics grounding the future dynamics prediction. From the top to bottom, we visualize ground-truth particles, particles predicted by DLF, and particles grounded/predicted by NeuroFluid.

- **NeRF:** It represents a static scene as 5D radiance fields of locations and view directions.
- **D-NeRF:** It extends NeRF to dynamic scenes with a deformation network that estimates 3D transitions \(\Psi : (x, t) \rightarrow \Delta x\) in a canonical space. Same as their setting, we randomly select one view from four views at every time step. The candidate views are the same as those used in the warm-up training stage.
- **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 \((x, d, t)\).
- **3D-aware fluid renderer:** Li et al. (2022) used an image encoder to learn latent fluid states, and fed the latent states to a NeRF to render the fluid surface. Unlike NeuroFluid, it is not based on the particles. Besides, it learns the dynamics model and the NeRF-based renderer separately.

We take 4 views for the training data, which are also involved in the warm-up training stage of PhysNeRF.

4.2. Performance on Fluid Dynamics Grounding

We compare NeuroFluid for particle dynamics grounding with DLF and DLF\(^†\), which have the same architecture as the transition model in NeuroFluid but are trained under the supervision of ground-truth particles. We evaluate: (i) the averaged Euclidean distance of the estimated particles to ground truth during the observation period \((i.e., d_{t<50})\), (ii) the end-point particle distance at \(t = 49\). All models are fed with true particle positions and velocities at \(t = 0\).

From Table 2, we can see that NeuroFluid consistently performs better than DLF on all benchmarks, indicating that it effectively exploits visual observations to improve the accuracy of fluid dynamics grounding. Besides, since
NeuroFluid: Fluid Dynamics Grounding with Particle-Driven Neural Radiance Fields

4.3. Performance on Future Prediction

To study whether the transition model has learned intrinsic physical transition principles, we conduct forward prediction for the next 10 time steps. Same as above, in Table 2, we compute the averaged Euclidean distance to future true particle positions \(d_{t=50}^{AVG}\) and the end-point distance \(d_{t=59}\). Overall, the predicted fluid dynamics of NeuroFluid is more accurate than DLF. On WaterCube and WaterSphere, DLF\(†\) performs slightly better than NeuroFluid. On HoneyCone, however, the predicted particles of DLF\(†\) diverge the most from the ground truth. A possible reason is that DLF\(†\) is prone to overfit the training particle dynamics in the observation period, being supervised by true particle states.

4.4. Performance on Novel View Synthesis

In Figure 4, we study PhysNeRF for novel view synthesis results at the observed time steps. Being fed with time-independent input, NeRF produces images that are static and blurry. Other compared methods can synthesize images with time-varying texture by simply encoding time index as input (NeRF-T), learning spatial deformation field (D-NeRF), or learning temporal latent representations (Li et al., 2022). However, the artifacts of blurry textures and 3D geometry still exist. In contrast, NeuroFluid synthesizes photo-realistic images in novel views, as the neighborhood encoding scheme introduces particle-based physical guidance. The rendered images show fine details of the fluid.
Table 4. Quality of image rendering for the next 10 time steps \((t \in [50, 59])\).

| METHOD    | WATERCUBE | WATERSPHERE | HONEYCONE |
|-----------|-----------|-------------|-----------|
|           | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ |
| NeRF      | 21.46 | 0.90 | 0.30 | 18.75 | 0.80 | 0.39 | 20.65 | 0.91 | 0.23 |
| NeRF-T    | 24.06 | 0.89 | 0.27 | 20.67 | 0.87 | 0.40 | 21.37 | 0.90 | 0.23 |
| D-Nerf    | 27.28 | 0.92 | 0.17 | 26.95 | 0.90 | 0.22 | 23.25 | 0.92 | 0.16 |
| LI et al. | 26.96 | 0.93 | 0.15 | 21.22 | 0.89 | 0.35 | 16.09 | 0.87 | 0.35 |
| NeuroFluid | 28.50 | 0.93 | 0.13 | 28.69 | 0.93 | 0.14 | 29.69 | 0.96 | 0.09 |

Table 5. Experiments on WaterCube with unknown initial particle states and ablation studies of neighborhood encoding (Rows 3-6).

| MODEL | GROUNDING | PREDICTION | NOVEL VIEW SYNTHESIS |
|-------|-----------|------------|----------------------|
|       | \(d_{t=50} \leq 50\) | \(d_{t=49} \geq 50\) | \(d_{t=50} \geq 50\) | PSNR↑ | SSIM↑ | LPIPS↓ |
| FULL MODEL | 28.8 | 34.9 | 35.5 | 36.7 | 30.76 | 0.95 | 0.09 |
| UNKNOWN INITIAL PARTICLE POSITIONS | 35.6 | 27.2 | 26.6 | 26.3 | 29.21 | 0.94 | 0.12 |
| w/o FICTITIOUS PARTICLE CENTER \((p_c)\) | 37.2 | 40.7 | 41.3 | 42.9 | 28.41 | 0.94 | 0.12 |
| w/o SPHERE DENSITY \((\sigma_p)\) | 31.2 | 37.9 | 39.3 | 39.4 | 29.65 | 0.95 | 0.10 |
| w/o DEFORMATION VECTOR \((v_D)\) | 33.0 | 38.1 | 40.5 | 42.1 | 28.91 | 0.95 | 0.11 |
| w/o PARTICLE-RELATIVE DIRECTION \((d_c)\) | 32.2 | 39.8 | 43.9 | 47.0 | 29.56 | 0.95 | 0.10 |

Figure 5. Results of PhysNeRF conditioned on true particle positions of novel scenes (WaterBunny), showing effective rendering of details in lighting and fluid geometry.

unknown initial particle positions. In previous experiments, we use ground-truth particle states at the initial time step \((t = 0)\). Here we investigate what if NeuroFluid is fed with estimated initial positions of the fluid particles. Specifically, we estimate the initial positions as follows. First, we train an original NeRF model at the initial time step using the sparse views in the warm-up stage. Second, we extract meshes from the results using the Marching Cube method. Third, we sample particles in the volume surrounded by the meshes as the initial particles of NeuroFluid. From Table 5, we can observe that even without the ground-truth initial particle positions, NeuroFluid can produce decent results in dynamics grounding, prediction, and novel view synthesis.

Ablation studies on neighborhood encoding. To verify the effectiveness of the neighborhood encoding scheme, we experiment with different variants of NeuroFluid by removing each component in PhysNeRF. In Table 5 (Rows 3-6), we report the performance of fluid dynamics grounding, future prediction, and novel view synthesis on WaterCube. As shown, the full neighborhood encoding scheme outperforms all these baseline models, especially in dynamics grounding and prediction. These results illustrate that the proposed representations of fictitious particle center \((p_c)\), sphere density \((\sigma_p)\), deformation \((v_D)\), and particle-relative direction \((d_c)\) are all essential for NeuroFluid.

Rendering novel fluid scenes. One contribution of PhysNeRF is to use neighborhood encoding to correlate the particles with the neural radiance field. To verify that PhysNeRF can effectively respond to different particle-based geometries, we use a pre-trained PhysNeRF to render a novel fluid scene with a new initial shape of water, i.e., Stanford Bunny. It is worth noting that the PhysNeRF is trained on other benchmarks of water scenes with different initial shapes of cubes, spheres, and cones. As shown in Figure 5, when fed with the ground-truth particles of WaterBunny, PhysNeRF
renders the water surface with high fidelity, showing the generalization ability to unseen fluid shapes. Because of this, PhysNeRF can effectively drive the entire model to reason about fluid dynamics in end-to-end optimization.

The resolution of observed images. The proposed NeuroFluid receives $400 \times 400$px images in the training stage. In Table 6, we show the influence of changing the size of visual observations. It is important to note that the higher resolution can slightly improve the results of dynamics grounding and future prediction, while we can still obtain reasonable results using lower resolution images.

| IMAGE RES. | GROUNDING $d_t<50$ | GROUNDING $d_t=49$ | PREDICTION $d_t<50$ | PREDICTION $d_t=49$ |
|------------|-------------------|-------------------|-------------------|-------------------|
| 200 $\times$ 200 PX | 29.2 | 35.9 | 37.2 | 37.0 |
| 400 $\times$ 400 PX | 29.0 | 35.1 | 36.2 | 36.0 |
| 800 $\times$ 800 PX | 29.0 | 34.5 | 36.0 | 36.0 |

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;b; Nguyen-Phuoc et al., 2018; Han et al., 2020; Chen et al., 2021). The most relevant work to fluid dynamics grounding is (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 computer graphics and related scientific areas (Chorin, 1968; Stam, 1999; Chentanez & Müller, 2013; Sulsky et al., 1995; Zhu & Bridson, 2005; Stomakhin et al., 2013; Jiang et al., 2015; Fang et al., 2020). Particle-based methods represent fluid dynamics through the 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; Ladicky 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 & 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 Neural Radiance Fields (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 & 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.

NeuroFluid still has some limitations that need to be further addressed in future work. One is that it requires initial particle velocities. Further discussion is needed on how to estimate initial velocities or find a way to bypass this requirement. Another issue is how to reduce the cumulative error of particle transitions, which may require defining effective physics-informed constraints in particle space.

Acknowledgements

This work was supported by the Natural Science Foundation of China (U19B2035, 62106144), Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102), and Shanghai Sailing Program (21ZS10202133).
References

Amada, T., Imura, M., Yasumuro, Y., Manabe, Y., and Chihara, K. Particle-based fluid simulation on gpu. In GPGPU, volume 41, pp. 42, 2004.

Battaglia, P. W., Hamrick, J. B., and Tenenbaum, J. B. Simulation as an engine of physical scene understanding. Proceedings of the National Academy of Sciences, 110 (45):18327–18332, 2013.

Bender, J. and Koschier, D. Divergence-free smoothed particle hydrodynamics. In SCA, pp. 147–155, 2015.

Biedert, T., Sohns, J.-T., Schröder, S., Amstutz, J., Wald, I., and Garth, C. Direct raytracing of particle-based fluid surfaces using anisotropic kernels. In EGPGV, pp. 1–12, 2018.

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

Chentanez, N. and Müller, M. Mass-conserving eulerian liquid simulation. IEEE Transactions on Visualization and Computer Graphics, 20(1):17–29, 2013.

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

Community, B. O. Blender - a 3D modelling and rendering package. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. URL http://www.blender.org.

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

Han, Z., Chen, C., Liu, Y.-S., and Zwicker, M. DRWR: A differentiable renderer without rendering for unsupervised 3D structure learning from silhouette images. In ICLR, 2020.

Holl, P., Koltun, V., and Thuerey, N. Learning to control pdes with differentiable physics. arXiv preprint arXiv:2001.07457, 2020.

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

Jiang, C., Schroeder, C., Selle, A., Teran, J., and Stomakhin, A. The affine particle-in-cell method. ACM Transactions on Graphics, 34(4):1–10, 2015.

Kajiya, J. T. and Von Herzen, B. P. Ray tracing volume densities. In SIGGRAPH, volume 18, pp. 165–174, 1984.

Kato, H., Ushiku, Y., and Harada, T. Neural 3D mesh renderer. In CVPR, pp. 3907–3916, 2018.

Kim, B., Azevedo, V. C., Thuerey, N., Kim, T., Gross, M., and Solenthaler, B. Deep fluids: A generative network for parameterized fluid simulations. In Computer Graphics Forum, volume 38, pp. 59–70, 2019.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In ICLR, 2015.

Ladický, L., Jeong, S., Solenthaler, B., Pollefeys, M., and Gross, M. Data-driven fluid simulations using regression forests. ACM Transactions on Graphics, 34(6):1–9, 2015.

Li, T.-M., Aittala, M., Durand, F., and Lehtinen, J. Differentiable monte carlo ray tracing through edge sampling. ACM Transactions on Graphics, 37(6):1–11, 2018a.

Li, Y., Wu, J., Tedrake, R., Tenenbaum, J. B., and Torralba, A. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. In ICLR, 2018b.

Li, Y., Wu, J., Tedrake, R., Tenenbaum, J. B., and Torralba, A. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. In ICLR, 2019.

Li, Y., Lin, T., Yi, K., Bear, D., Yamins, D., Wu, J., Tenenbaum, J., and Torralba, A. Visual grounding of learned physical models. In ICML, pp. 5927–5936, 2020.

Li, Y., Li, S., Sitzmann, V., Agrawal, P., and Torralba, A. 3D neural scene representations for visuomotor control. In CoRL, pp. 112–123, 2022.

Li, Z., Niklaus, S., Snavely, N., and Wang, O. Neural scene flow fields for space-time view synthesis of dynamic scenes. In CVPR, 2021.

Liu, L., Habermann, M., Rudnev, V., Sarkar, K., Gu, J., and Theobalt, C. Neural actor: Neural free-view synthesis of human actors with pose control. ACM Transactions on Graphics, 40(6):1–16, 2021.

Liu, S., Li, T., Chen, W., and Li, H. Soft rasterizer: A differentiable renderer for image-based 3D reasoning. In ICCV, pp. 7708–7717, 2019.

Loper, M. M. and Black, M. J. OpenDR: An approximate differentiable renderer. In ECCV, pp. 154–169, 2014.

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. NeRF: Representing scenes as neural radiance fields for view synthesis. In ECCV, pp. 405–421, 2020.
NeuroFluid: Fluid Dynamics Grounding with Particle-Driven Neural Radiance Fields

Monaghan, J. J. Smoothed particle hydrodynamics. *Annual Review of Astronomy and Astrophysics*, 30:543–574, 1992.

Mrowca, D., Zhuang, C., Wang, E., Haber, N., Fei-Fei, L., Tenenbaum, J. B., and Yamins, D. L. Flexible neural representation for physics prediction. In *NeurIPS*, pp. 8813–8824, 2018.

Müller, M., Charypar, D., and Gross, M. H. Particle-based fluid simulation for interactive applications. In *SCA*, pp. 154–159, 2003.

Müller, S., Milano, M., and Kounoutsakos, P. Application of machine learning algorithms to flow modeling and optimization. *Annual Research Briefs*, pp. 169–178, 1999.

Nguyen-Phuoc, T., Li, C., Balaban, S., and Yang, Y.-L. RenderNet: A deep convolutional network for differentiable rendering from 3D shapes. In *NeurIPS*, 2018.

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

Peng, S., Dong, J., Wang, Q., Zhang, S., Shuai, Q., Zhou, X., and Bao, H. Animatable neural radiance fields for modeling dynamic human bodies. In *ICCV*, pp. 14314–14323, 2021.

Poloni, C., Giurjevich, A., Onesti, L., and Pediroda, V. Hybridization of a multi-objective genetic algorithm, a neural network and a classical optimizer for a complex design problem in fluid dynamics. *Computer Methods in Applied Mechanics and Engineering*, 186(2-4):403–420, 2000.

Price, D. J. Smoothed particle hydrodynamics and magnetohydrodynamics. *Journal of Computational Physics*, 231(3):759–794, 2012.

Pumarola, A., Corona, E., Pons-Moll, G., and Moreno-Noguer, F. D-NeRF: Neural radiance fields for dynamic scenes. In *CVPR*, pp. 10318–10327, 2021.

Qi, C. R., Yi, L., Su, H., and Guibas, L. J. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017.

Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., and Battaglia, P. Learning to simulate complex physics with graph networks. In *ICML*, pp. 8459–8468, 2020.

Schenck, C. and Fox, D. SPNets: Differentiable fluid dynamics for deep neural networks. In *CoRL*, pp. 317–335, 2018.

Stam, J. Stable fluids. In *SIGGRAPH*, pp. 121–128, 1999.

Stomakhin, A., Schroeder, C., Chai, L., Teran, J., and Selle, A. A material point method for snow simulation. *ACM Transactions on Graphics*, 32(4):1–10, 2013.

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

Sulsky, D., Zhou, S.-J., and Schreyer, H. L. Application of a particle-in-cell method to solid mechanics. *Computer Physics Communications*, 87(1-2):236–252, 1995.

Ummenhofer, B., Prantl, L., Thuerey, N., and Koltun, V. Lagrangian fluid simulation with continuous convolutions. In *ICLR*, 2019.

Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

Wu, J., Lu, E., Kohli, P., Freeman, B., and Tenenbaum, J. Learning to see physics via visual de-animation. In *NeurIPS*, volume 30, pp. 153–164, 2017a.

Wu, J., Wang, Y., Xue, T., Sun, X., Freeman, W. T., and Tenenbaum, J. B. MarrNet: 3D shape reconstruction via 2.5D sketches. In *NeurIPS*, 2017b.

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, pp. 586–595, 2018.

Zhu, Y. and Bridson, R. Animating sand as a fluid. *ACM Transactions on Graphics*, 24(3):965–972, 2005.