






Efficient Generation of Multimodal Fluid Simulation Data

D. Baieri¹  and D. Crisostomi¹  and S. Esposito²  and F. Maggioli³  and E. Rodolà¹ 

¹Sapienza – University of Rome, Italy

²University of Tübingen, Germany

³University of Milano-Bicocca, Italy

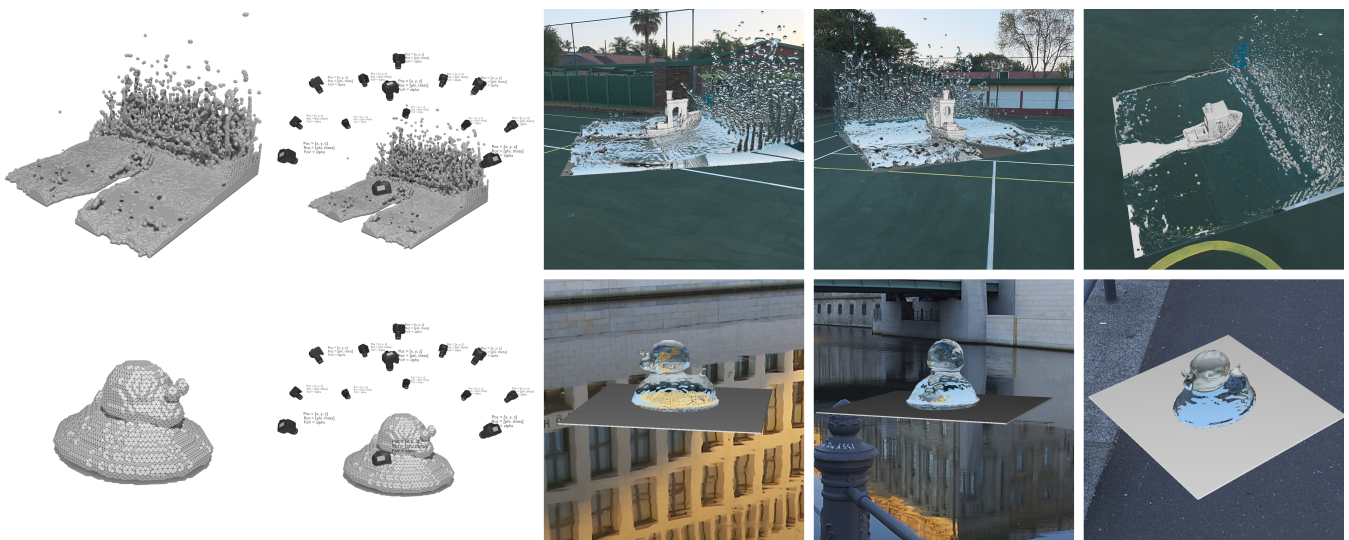


Figure 1: Our tool performs fluid simulations with the Lattice Boltzmann Method (LBM), exporting the geometry of the fluid and rendered frames from multiple viewpoints at each simulated step. Camera positions are sampled via Fibonacci sphere sampling [KISS15] to ensure uniform coverage, and exported as metadata.

Abstract

In this work, we introduce an efficient and intuitive framework to produce synthetic multi-modal datasets of fluid simulations. The proposed pipeline can reproduce the dynamics of fluid flows and allows for exploring and learning various properties of their complex behavior from distinct perspectives and modalities. We aim to exploit these properties to fulfill the community's need for standardized training data, fostering more reproducible and robust research. We employ our framework to generate a set of thoughtfully designed training datasets, which attempt to span specific fluid simulation scenarios meaningfully. We demonstrate the properties of our contributions by evaluating recently published algorithms for the neural fluid simulation and fluid inverse rendering tasks using our benchmark datasets.

CCS Concepts

• **Computing methodologies** → **Physical simulation**; **Computer graphics**;

1. Introduction

Applying the representational power of machine learning to the prediction of complex fluid dynamics has been a relevant subject of study for years and is regarded today as an established research field [LFBC23]. However, the amount of available fluid simula-

tion data does not match the notoriously high requirements of machine learning methods. Researchers have typically addressed this issue by generating their own datasets, as in the case of Pfaff *et al.* [PFSGB21] and Ummerhofer *et al.* [UPTK20], only to cite a few. The obvious consequence of this behavior is an heterogeneous

distribution of training data across different methods, which in turn prevents a consistent comparison of the proposed approaches.

Recently, the growing diffusion of datasets for animations has largely contributed to the enhancement of data-driven models, streamlining traditional computer graphics challenges and expanding digital artists' creative possibilities. Notably, skeleton-based animation datasets, supported by specialized data capture technology, have become prevalent due to their diverse applications. Prominent examples include DFAUST [BRPMB17] for human body animations and DeformingThings4D [LTT*21] for a wider range of subjects. One really successful application of human motion datasets was the creation of SMPL [LMR*15], a skinned multi-person linear model, which may be used for traditional skinning as well as in a generative fashion, to sample novel human poses and styles.

In this setting, the introduction of a large-scale dataset of fluid animations would be largely beneficial to the advancement of data-driven methods for the dynamics of fluids. However, capturing intricate fluid deformations contrasts sharply with the goal of producing a large-scale dataset, as the task necessitated of specialized, expensive, and labor-intensive tools [HHL*05, EUT19].

With this work, we introduce a novel pipeline for the generation of synthetic multi-modal fluid simulations datasets. By leveraging a GPU implementation and well-established efficient implementations of fluids dynamics solvers, our framework is efficient enough that no data needs to be exchanged between users, except for the configuration files required to reproduce the dataset. Furthermore, our procedure allows multiple modalities (*e.g.*, fluid geometry, photorealistic renderings) and is general enough for it to be applied to various tasks in data-driven fluid simulation. The dataset can be constructed using flexible templates, which can be customized through a configuration file by defining variables and constants such as initial fluid states, boundary conditions, and simulation parameters. The templates are extensible to incorporate extra features, like initial velocities or dynamic force fields. To enforce a better coverage and uniformity in the data, we generate multiple scenes from the same template by varying the configuration parameters. We simulate each scene using a highly efficient GPU-based Lattice Boltzmann simulator [Leh], enabling fast large-scale dataset creation. Our pipeline also exposed additional features and options, such as multi-view renderings, various file formats for the exported data, and support to both Eulerian and Lagrangian simulation data.

This work holds potential for various research areas. We demonstrate its utility in two key domains that have garnered considerable attention: advancing data-driven fluid simulation models [XZB24, UPTK20, WXCT19] and tackling inverse rendering/surface recovery [CLZ*22, LQC*23]. Overall, our work aims to bridge a long-standing gap in data-driven fluid simulation research, enhancing community contributions, reproducibility, and systematic evaluation. Our key contributions are: **a)** introducing an efficient framework for generating synthetic, multi-modal fluid simulation data, capturing a wide range of fluid dynamics (see Figure 1); **b)** using this framework to generate three training datasets for distinct fluid simulation scenarios, providing training data for consistent future research; **c)** demonstrating the effectiveness of our datasets by successfully training state-of-the-art models for fluid simulation and inverse rendering tasks.

2. Motivation and applications

As mentioned in the previous section, the increasing availability of large-scale skeleton-based animation datasets boosted the research on data-driven models for human and non-human animations. We believe that our dataset generation framework has the potential of contributing in a similar way to the field of data-driven fluid simulation, enhancing the reproducibility of methods, simplifying the distribution of codebases and data, and offering new research avenues.

Notably, our pipeline supports fluid super-resolution, where models learn to enhance low-resolution simulations into detailed, high-resolution outputs. GAN models have been notably effective in this area [WXCT19, XFCT18], improving super-resolution in scenarios like smoke flow upsampling and turbulence prediction [BLDL20, BWDL21]. Further research extends this approach to complex scenarios like multi-phase flows [LM22] and label-free super-resolution [GSW21]. This methodology also shows promise in simulating biological systems, indicating its broad applicability [FSD*20]. More in general, one could exploit neural networks to learn the entire simulation model using the current fluid state as input, adopting either Lagrangian or Eulerian perspectives. The Lagrangian approach focuses on particle-based fluid dynamics. Continuous convolution on point sets has been shown to yield efficient and robust models [UPTK20]. Enhancements include momentum conservation [PUKT22] and applying graph neural networks to particle-based models [LF22], improving performance without sacrificing quality. Conversely, the Eulerian viewpoint treats fluids and properties (like velocity) as functions over space. Innovations here include generative models for velocity fields [KAT*19] and using latent spaces in generative models to produce stable and controllable simulations [WKA*20, WBT19]. We refer to Vinuesa and Brunton [VB22] for a comprehensive review of learning paradigms in fluid simulation. More recent research explores NeRF's inverse rendering technique [MST*20] in fluid dynamics. Chu *et al.* [CLZ*22] first integrated Navier-Stokes principles into 4D NeRF training, developing a dual-network model enforcing physical constraints to accurately reconstruct smoke density with limited camera views. NeuroFluid [GDWY22] built on this, adopting a Lagrangian approach and incorporating physics constraints into volume rendering for model optimization. Li *et al.* [LQC*23] combined Eulerian and Lagrangian methods, achieving remarkable versatility across diverse materials. However, these approaches overlook real-world fluid material properties, treating color as a mere surface texture. NeReF [WYC*22] addressed this by formulating a NeRF that considers reflective and transmissive materials, enabling realistic light-fluid interactions in transparent fluids like water. Finally, Dent *et al.* [DYZ*23] proposed a dynamic simulation method for turbulent fluids represented as implicit neural fields.

A prevalent problem in the cited research is the absence of a standardized benchmark dataset, leading researchers to create new simulation datasets for each method evaluation. Our generation tool addresses this by enabling: a) export of fluids as both particles and density fields, b) multi-view renderings of simulations, and c) efficient dataset regeneration using a configuration file, eliminating the need to distribute massive volumes of data.

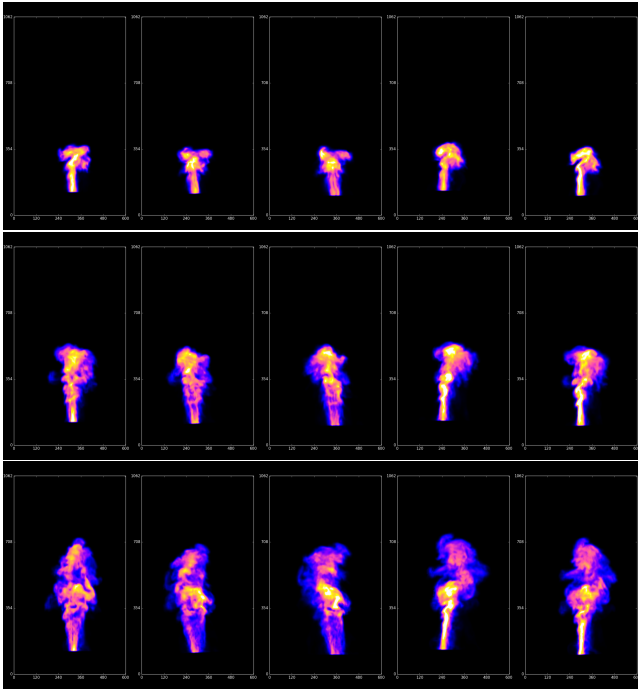


Figure 2: Multiple rendered views for three timesteps of a smoke plume simulation from the ScalarFlow [EUT19] dataset. The density field is reconstructed from real smoke captures.

3. Related work

While our dataset is not the first in fluid simulation data, it differs significantly from prior works like the Johns Hopkins Turbulence Databases (JHTD) [GKY*16, PBLM07, LPW*08]. JHTD offers extensive, high-resolution data on turbulent flows, targeting geophysical and engineering applications like meteorology and aerodynamics. However, its massive scale and detail, with thousands of frames and billions of voxels per frame, makes it less suitable for computer graphics and vision applications. In contrast, our contribution is designed with practicality and adaptability for CG/CV purposes as a primary focus.

Many datasets accompanying data-driven simulation studies are limited in terms of availability, data distribution, modality, and applicability. Pfaff *et al.* [PFSGB21]’s work, using graph neural networks [SGT*08] on triangle meshes, was confined to 2D due to planar mesh limitations in representing 3D volumes. EA-GLE dataset [JBN*23], also mesh-based, focused only on turbulent wind dynamics influenced by boundary shapes. Notably, very specific datasets like those provided by Ummenhofer *et al.* [UPTK20] were initially minor contributions, but later became foundational for training methods like those in Prant *et al.* [PUKT22]. Jakob *et al.* [JGG20] proposed a large dataset of 2D laminar and turbulent fluid parametrized by the Reynolds number for neural flow map interpolation. Stachenfeld *et al.* [SFK*22] introduced volumetric datasets across different dimensions, modeling real-world chaotic systems, but they are all limited to this type of representation. In contrast, our dataset generation framework is specifically tailored for computer graphics applications, offering additional visual data.

The work most similar in spirit to ours is ScalarFlow [EUT19], a collection of video captures of real-world smoke plumes dynamics (see Figure 2), coupled with density field reconstructions computed by an algorithm proposed by the authors. Despite ScalarFlow being a large scale dataset (ca. 26 billion voxels of data), its scope is limited to the real-world capture setting set up by the authors; when using real data over synthetic would provide no significant advantage, our generation framework could provide similar data in arbitrary quantities, without the additional cost of capture. Recent proposals by Toshev *et al.* [TA23, TGF*23] provided contributions in similar, albeit complementary, directions to our own: their LagrangeBench tool and data allow to systematically test pre-trained neural Lagrangian simulation models on small synthetic benchmark datasets consisting of well-known simulation settings. Such efforts further highlight the demand for such resources in the field. Combining these works with our framework for the efficient generation of large-scale datasets could create a full pipeline for training and evaluating neural Lagrangian simulation models.

Lastly, we mention that using large-scale simulation data for training data-driven models is not entirely new even outside the field of fluid dynamics. In particular, synthetic and simulation data are largely used in a wide range of research areas spanning garment animation [BME20, PLPM20, CPA*21], precision agriculture [MKH*23, KWP*24], and scene understanding [RRR*21, KOH*24].

4. Dataset

In this section, we outline our data generation process, providing high-level implementation details. Additional and more detailed documentation is available in our codebase, which will be released upon acceptance. We use this procedure to create three datasets in total: two mid-scale training datasets for data-driven fluid simulation and a set of scenes for fluid inverse rendering algorithms.

4.1. Data generation

Our C++ code library[†] is built on top of the FluidX3D Lattice-Boltzmann GPU simulator [Leh]. This was chosen considering its a) impressive efficiency on mid-range hardware, b) built-in ray-traced rendering capabilities, and c) wide compatibility with several operating systems and architectures due to the implementation of GPU kernels in OpenCL. We refactored the core implementation and improved the API to expose most functionalities via a newly introduced `Scene` class. By using this class as a wrapper for FluidX3D, we are not binding our framework to a specific implementation or solver, making it easy to replace it in the future. Through the `Scene` objects, we represent the parameters and the initial state of the simulation, facilitating a fine-grained configuration and enabling inheritance for extending base behavior. For instance, initial fluid shapes and boundary conditions can be provided through input mesh files, which are adjustable by means of rigid transformations and scaling. Enabling specialized features like point-wise external force fields or inflows/outflows can be easily accomplished by overriding some of the basic functionalities of the `Scene` class with new subclasses.

[†] Code at <https://github.com/daniele-baieri/FSD>

Listing 1: JSON configuration file for our generation procedure. Variables are sampled and aggregated in order to generate configuration files for individual instances in the dataset. Fluid bodies and solid obstacles can be specified both as constants and variables, and sampled according to specific properties of geometric primitives or meshes (e.g., center, scale, rotation).

```

"export_root": "path/to/out/dir",
"seed": 123456,
"constants": {
  "sim_params": {
    "Nx": 256, "Ny": 256, "Nz": 256,
    ...
  },
  "obstacles": {
    ...
  },
  ...
},
"variables": {
  "sim_params": {
    "nu": {
      "type": "linscale",
      "vmin": 0.0005, "vmax": 0.005, "steps": 20
    },
    "sigma": {
      "type": "normal",
      "mean": 0.001, "std": 0.0005, "nsamples": 20
    }
  },
  "fluids": [
    ...
  ]
}

```

The refactored implementation of FluidX3D serves as a library for dataset-specific `Scene` subclasses, which can be designed from the user and directly accessed from the simulation tool. The parameters of these subclasses are entirely configurable using a simple JSON file (see Listing 1), and the generation procedure is provided in form of a Python script that creates configuration files and feeds them to the simulation tool. The script takes various input parameters, including global generation parameters (e.g., random seed for reproducibility), as well as constants and variables for `Scene` attributes: constants are preserved in generated instances, while the variables are sampled based on their selected distribution, which includes linear intervals, uniform distributions, normal distributions and collections. Users can also set a maximum scene count, in which case scene instances are uniformly sampled from all possible combinations of variables. This configuration is easily supplied via a separate JSON file. During the simulation, specific frames can be extracted both as multi-view renderings and geometric information. For rendering efficiency, we again rely on the specialized FluidX3D implementation, as it balances speed over photorealism. Camera ray are bounced up to two times and use the mean of material BSDF, rather than sampling them; this way, the light interactions of the water material we use for fluids can still be displayed, while the rendering procedure remains really fast, being executed on the GPU (see Section 5.3). For geometric data extraction, we offer two options: a) exporting fluids as binary particle arrays, with the number of particles determined by the configuration file input and b) exporting the fluid’s density field as a memory-efficient 3D matrix in sparse coordinate format (COO), stored as a 1D binary array. Despite the

latter method being very memory-efficient (see Section 5.3), selecting a specific number of particles may be more convenient at high resolutions. We provide a Python loader to Numpy arrays for both file formats.

4.2. Coherent subsets of simulation space

In virtually every application, an important property for a dataset is to have a reasonably rich distribution of data representing “semantically similar” cases or features. Indeed, the data distribution must somehow encode that such case or feature has not a specific appearance.

The generation procedure we described in Section 4.1 is specifically designed with this property in mind. By defining a parametric `Scene` object, we can change the value of the parameters to generate collections of “similar” (in a fluid dynamics sense) simulations, regardless of the scale of such collections. In this regard, we used our framework to introduce two medium scale benchmark datasets based on well-known fluid dynamics settings. Throughout this section, we will refer to 3D coordinate systems assuming that the up-axis is the z direction.

Our first training dataset (`obstacles`) consists of simulations displaying a sphere of fluid colliding with varying boundary geometries under the effect of gravity. We define a cuboid boundary $B = (\mathbf{p}, \mathbf{q})$ enveloping the entire scene, where $\mathbf{p}_{\{x,y,z\}}$ and $\mathbf{q}_{\{x,y,z\}}$ are the min/max coordinate values for the vertices of B , respectively. We represent the initial fluid state by a sphere $S = (\mathbf{c}, r)$ and the solid obstacle as a triangle mesh $M = (V, F)$, which we can instantiate by sampling its center, rotation and size (\mathbf{o}, θ, s) . In order to ensure interaction between the two, we place the mass of fluid and the obstacle on the vertical axis passing through the center of the cuboid (namely, $\mathbf{c}_x = \mathbf{o}_x = (\mathbf{q}_x - \mathbf{p}_x)/2$ and $\mathbf{c}_y = \mathbf{o}_y = (\mathbf{q}_y - \mathbf{p}_y)/2$). We sample the radius $r \sim \mathcal{U}(r_{\min}, r_{\max})$ and mesh size $s \sim \mathcal{U}(s_{\min}, s_{\max})$ as the maximum dimension of its bounding box. Since we want the fluid above the obstacle to ensure interaction, we sample the two in the top and bottom halves of the simulation domain, respectively. Thus, we sample $\mathbf{c}_z \sim \mathcal{U}\left(\frac{\mathbf{q}_z - \mathbf{p}_z}{2} + r_{\max}, \mathbf{q}_z - r_{\max}\right)$ and $\mathbf{o}_z \sim \mathcal{U}\left(\mathbf{p}_z + \frac{s_{\max}}{2}, \frac{\mathbf{q}_z - \mathbf{p}_z}{2} - \frac{s_{\max}}{2}\right)$, where we choose $r_{\min}, r_{\max}, s_{\min}$, and s_{\max} to be, respectively, the 10%, 20%, 25% and 45% of the cuboid’s height $\mathbf{q}_z - \mathbf{p}_z$. Lastly, we randomly rotate M along x to augment the dataset with additional unique boundary interactions, i.e., $\theta_x \sim \mathcal{U}(0, \frac{\pi}{2})$. The geometry of the mesh M is chosen randomly from a small collection of 5 shapes with varying complexity.

The second dataset, which we refer to as `dam-break`, is a collection of instances of this well-known fluid dynamics scenario. In this setting, a cuboid of fluid is placed in contact with the lower face of a second cuboid boundary $B = (\mathbf{p}, \mathbf{q})$, and the scene is simulated under the effect of gravity. The only variable for our scene generation procedure is the initial placement of the dam inside the scene: if we represent the cuboid mass of fluid by its center and sides, i.e., $D = (\mathbf{c}, \mathbf{s})$, we set the y coordinates to span the entire y -axis ($\mathbf{s}_y = \mathbf{q}_y - \mathbf{p}_y, \mathbf{c}_y = \mathbf{s}_y/2$) and sample the x coordinates as $\mathbf{c}_x \sim \mathcal{U}(\mathbf{p}_x + \delta_x, \mathbf{q}_x - \delta_x)$ and $\mathbf{s}_x \sim \mathcal{U}(s_x^{\min}, \min\{\mathbf{c}_x, \mathbf{N}_x - \mathbf{c}_x\})$, where we set s_x^{\min} and δ_x to the 10% and the 25% of $\mathbf{q}_x - \mathbf{p}_x$, respectively. For the z coordinates, since we require the dam to touch

Table 1: Real-world scale of the obstacles and dam-break datasets. Legend: $\delta x, \delta m, \delta t$ \rightarrow space, mass and time units (respectively). v \rightarrow kinematic shear viscosity. σ \rightarrow surface tension. ρ \rightarrow density. g \rightarrow gravitational acceleration. Re_{max} \rightarrow upper bound for the Reynolds number of the simulations.

	Real	LBM
δx	$7.81 \cdot 10^{-3} [m]$	1
δm	$4.76 \cdot 10^{-4} [kg]$	1
δt	$4.45 \cdot 10^{-4} [s]$	1
v	$2.00 \cdot 10^{-3} [m^2/s]$	$1.46 \cdot 10^{-2}$
σ	$7.20 \cdot 10^{-2} [kg/s^2]$	$3.00 \cdot 10^{-4}$
ρ	$1.00 \cdot 10^3 [kg/m^3]$	1
g	$9.81 [m/s^2]$	$2.49 \cdot 10^{-4}$
Re_{max}	10119	10119

the ground, we only sample the center $\mathbf{c}_z \sim \mathcal{U}(\mathbf{p}_z + \delta_z, \frac{\mathbf{q}_z - \mathbf{p}_z}{2})$ and set $\mathbf{s}_z = 2(\mathbf{c}_z - \mathbf{p}_z)$, so that the mass of fluid is in contact with the ground. Here, δ_z is set to the 25% of $\mathbf{q}_z - \mathbf{p}_z$.

We choose physical properties of the fluid to achieve: a) reasonable space/time scale, b) water-like behavior, c) stability of the simulation. The resulting real world scale, coupled with the corresponding LBM units, is detailed in Table 1. We fix the simulation domain to have characteristic length $L = 2[m]$ in all dimensions, and select density and surface tension values to be identical to those of water. Since we are using a LBM, we need to fix the simulation resolution to obtain the remaining quantities; we choose $\mathbf{N} = (256, 256, 256)$, which we observed to yield a reasonable tradeoff between quality and efficiency. From the fixed data, we obtain $\delta x = L/\max\{\mathbf{N}\}$, and $\delta m = \rho \cdot \delta x^3$. To determine the timestep, we fix the surface tension in LBM units and obtain $\delta t = \sqrt{\sigma_{real}/\sigma_{LBM}} \cdot \delta m$. The value is empirically chosen to obtain a reasonable time scale. In this setting, using the real kinematic shear viscosity of water (10^{-6}) would dramatically increase the Reynolds number $Re_{max} = (\min\{\mathbf{N}\} \cdot \mathbf{u}_{max})/v_{LBM}$ (where $\mathbf{u}_{max} = 1/\sqrt{3}$ in LBM) of our simulations, yielding extremely turbulent flows which can make the simulation unstable. Therefore, we reduce the viscosity to keep the maximum Reynolds number around 10^4 . This value is large enough to capture a wide range of fluid behaviors, and we verified ex-post that it yields stable simulations. We give complete details on the generation of the two datasets in Table 2.

4.3. Hand-crafted multi-view scenes

Another desirable property for a dataset is that it must be easy to design, extend, and distribute. These requirements collide with the type and volume of data expected by a dataset of fluid dynamics, where the already large amount of data for representing a fluid in a 3D domain must be multiplied by a large number of frames and scenes.

Nonetheless, with our generation framework we are able to describe a scene by just using a small JSON file describing the scene configuration. Users can exchange these lightweight files and rely on the efficiency of FluidX3D for generating the data in short time. In this setting, even designing adding other scenes to the dataset becomes a simple task.

Table 2: Generation data for our neural fluid simulation datasets, exporting one frame every 0.02s of simulation (i.e., every 42 LBM steps with δt as in Table 1). The resulting throughput proves the efficiency of our procedure: allowing for a 10% decay in performance, we could be able to generate a large scale 1mln frames dataset in ~ 45 hours.

Dataset	dam-break	obstacles
Generation time (s)	15253	14654
Throughput (FPS)	6.55	6.82
Sampled scenes	200	400
t_{max} (s)	10.0	5.0
Frame count	100.000	100.000
Frame time (s)	0.02	0.02
Size (GB)	23.2	17.5
Particles	5000	3000

We have used our generation procedure to simulate a limited set of hand-crafted scenes, primarily intended for training inverse rendering and surface recovery models. These scenes are not designed to cover a wide spectrum of fluid behaviors but rather serve as challenging test cases to assess the accuracy of novel methods and their ability to generalize. Details about the ship scene’s data are summarized in Figure 1, and comprehensive information on real-world scale and simulation for each scene is provided in Table 3. We simulate all scenes with resolution $\mathbf{N} = (256, 256, 256)$ and export the results at 60FPS. 10 viewpoints are sampled in the upper hemisphere via Fibonacci sphere sampling [KISS15], and each output frame is rendered with resolution 1080×1080 twice: once with a water-like material and once with a Lambertian material. For each camera, a background-only render is also produced, and the camera parameters for the viewpoints are exported as JSON. The data totals 20.32GB, and we stress that they can be distributed by only providing a few MB of mesh files and configuration files.

5. Evaluation

To evaluate our framework, we show how a dataset of fluid dynamics data could be applicable across different applications and for testing state-of-the-art techniques. Furthermore, to prove that our pipeline allows for an easier distribution of the data, we analyze the performance of FluidX3D in generating the data presented in Sections 4.2 and 4.3.

5.1. Data-driven Fluid Simulation

Among the most attractive characteristic of data-driven models there is probably their ability to perform very well on a wide range of challenging tasks. State-of-the-art methods like the *Deep Langrangian Fluid* (DLF) model introduced by Ummenhofer *et al.* [UPTK20] proved that this data-driven paradigm can be successfully applied to the prediction of fluid dynamics. We train and evaluate the DLF model on the datasets described in Section 4.2 to prove that the ability to produce a large number of variations for the same scene can be beneficial for successfully training a model.

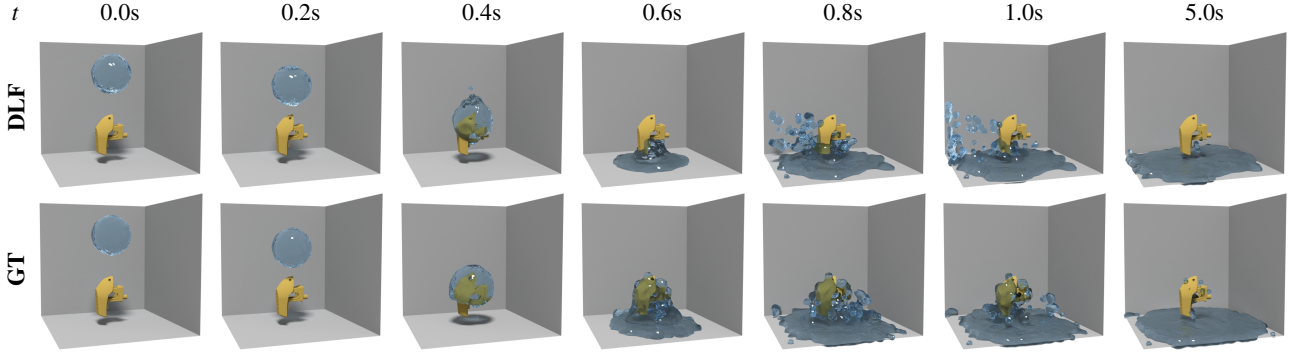


Figure 3: Samples from a 5 seconds rollout of a DLF [UPTK20] model trained on our *obstacles* dataset, with unseen initial conditions, compared to ground truth LBM simulation.

Table 3: Generation time and real-world scale for the scenes in our inverse rendering dataset. ρ and g are as in Table 1 for all scenes. We fix the reference real world $\sigma = 0.072 \text{ [kg/s}^2\text{]}$ and vary the LBM coefficient for surface tension.

Scene	t_{\max} (s)	Frames	L [m]	δx [m]	δm [kg]	δt [s]	v [m^2/s]	σ [LBM]	Re_{\max}
ball	1.5	90	2.0	$7.81 \cdot 10^{-3}$	$4.76 \cdot 10^{-4}$	$8.13 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	$1 \cdot 10^{-4}$	22170
dam	2.5	150	2.0	$7.81 \cdot 10^{-3}$	$4.76 \cdot 10^{-4}$	$2.57 \cdot 10^{-4}$	$2 \cdot 10^{-3}$	$1 \cdot 10^{-5}$	17527
duck	1.5	90	2.0	$7.81 \cdot 10^{-3}$	$4.76 \cdot 10^{-4}$	$8.13 \cdot 10^{-4}$	$5 \cdot 10^{-2}$	$1 \cdot 10^{-4}$	222
ship	2.5	150	2.0	$7.81 \cdot 10^{-3}$	$4.76 \cdot 10^{-4}$	$8.13 \cdot 10^{-4}$	$5 \cdot 10^{-3}$	$1 \cdot 10^{-4}$	2217

Model Ummenhofer *et al.* [UPTK20] aim to understand fluid mechanics by studying particle motion. In particular, a continuous convolutional network is fed with a collection of particles, each paired with its features. Each particle is associated with a feature vector that is a constant scalar of 1, paired with the particle’s velocity, denoted by \mathbf{v} , and its viscosity, ν . Therefore, at any timestep n , a particle p_i^n is represented by the tuple $(\mathbf{x}_i^n, [1, \mathbf{v}_i^n, \nu_i^n])$. To compute intermediate velocities and positions, and integrate external force information, the velocity is listed as an input feature. Using Heun’s method, the intermediate velocities \mathbf{v}_i^{n*} and positions \mathbf{x}_i^{n*} for timestep n are then computed as:

$$\mathbf{v}_i^{n*} = \mathbf{v}_i^n + \Delta t \mathbf{a}_{\text{ext}}, \quad \mathbf{x}_i^{n*} = \mathbf{x}_i^n + \Delta t \frac{\mathbf{v}_i^n + \mathbf{v}_i^{n*}}{2} \quad (1)$$

where \mathbf{a}_{ext} represents an acceleration vector, enabling the application of external forces like fluid control or gravity. These intermediate values are devoid of any particle or scene interactions; such interactions are incorporated using the ConvNet. For the network to manage collisions, another group of static particles s_j are introduced. These are sampled along scene boundaries and paired with normals \mathbf{n}_j as their feature vectors, expressed as $s_j = (\mathbf{x}_j, [\mathbf{n}_j])$. The network performs the function:

$$[\Delta \mathbf{x}_1, \dots, \Delta \mathbf{x}_N] = \text{ConvNet} \left(\{p_i^{n*}\}_{i=1}^N, \{s_j\}_{j=1}^M \right), \quad (2)$$

employing convolutions to merge features from both sets of particles. Here, $\Delta \mathbf{x}$ serves as a position correction, factoring in all particle interactions, including collisions with the scene boundary. The correction is finally used to update positions and velocities for timestep $n+1$ with:

$$\mathbf{x}_i^{n+1} = \mathbf{x}_i^{n*} + \Delta \mathbf{x}_i, \quad \mathbf{v}_i^{n+1} = \frac{\mathbf{x}_i^{n+1} - \mathbf{x}_i^n}{\Delta t} \quad (3)$$

Experiment and results We train DLF over *obstacles*, split as train and validation sets with a 90:10 ratio, which amounts to 360 training simulations (tot. 90k frames) and 40 validation simulations (tot. 10k frames). Figure 3 shows a qualitative comparison of the model prediction vs. the ground truth simulation, after training DLF to convergence (50k total iterations), given an initial state from the validation set. The model is able to consistently predict realistic dynamics throughout the simulation, and reach a stable state coherent with the one shown in the data, proving our data to be suitable for this application; the only shortcoming of the learned model is its inability to properly capture viscous behavior, probably due to the viscosity not being explicitly modeled in its formulation (eqs. (1) to (3)).

5.2. Inverse Rendering Fluid Simulations

The advent of NeRF-like models gave a massive boost to the research about inverse rendering tasks. Nonetheless, they mostly rely on multi-view images of the same scene, which can be quite expensive and time-consuming to obtain. Instead, simulating a single scene and rendering it from multiple points of view can effectively provide a large amount of data for testing and evaluating a model at a more reasonable cost and with less time.

Our method allows the generation multi-view video data, labeled with camera positions and parameters, which can be used for inverse rendering tasks. By defining a Δt and a set of randomly sampled cameras on the upper hemisphere (see Figure 1), we render the scene from these viewpoints during simulation. While our data can be used to train various NeRF-based models, both static and dynamic, we tested the dataset described in Section 4.3 on PAC-NeRF [LQC*23]. PAC-NeRF recovers explicit geometric representations and physical

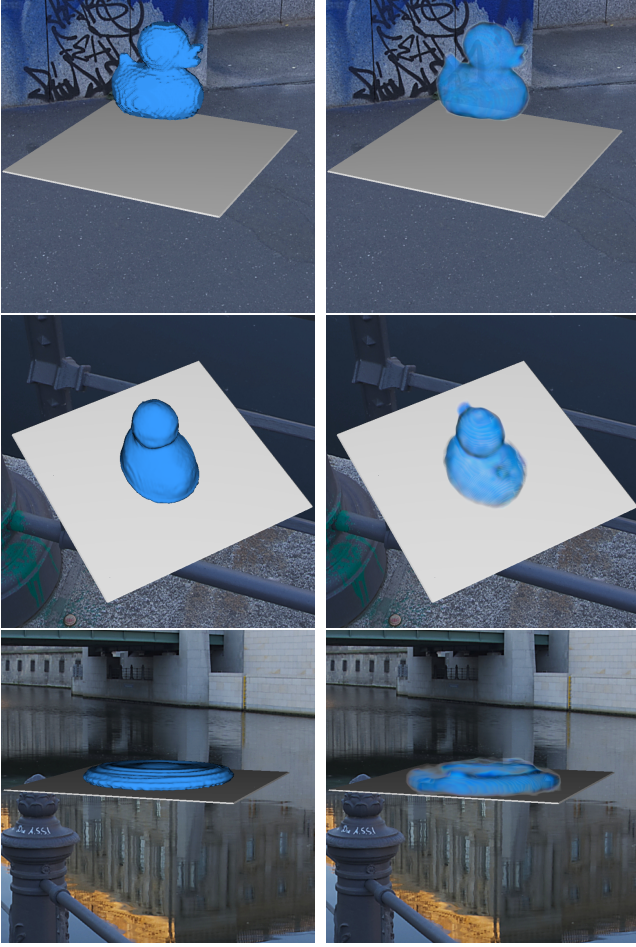


Figure 4: Left: a subset of the views in our *duck* scene training data. Right: PAC-NeRF [LQC*23] reconstruction and rendering for the same (unseen) views (the background is not learned by the model, so it was manually composited for this visualization).

properties of dynamic objects in scenes by combining neural scene representations with differentiable physics engines for continuum materials.

Model A dynamic NeRF comprises time-dependent volume density field $\sigma(\mathbf{x}, t)$ and a time-and-view-dependent appearance (color) field $\mathbf{c}(\mathbf{x}, \omega, t)$ for each point $\mathbf{x} \in \mathbb{R}^3$, and directions $\omega = (\theta, \phi) \in \mathbb{S}^2$ (spherical coordinates). The appearance $\mathbf{C}(\mathbf{r}, t)$ of a pixel specified by ray direction $\mathbf{C}(\mathbf{r}, t)$ ($s \in [s_{\min}, s_{\max}]$) is obtained by differentiable volume rendering [MST*20]:

$$\mathbf{C}(\mathbf{r}, t) = \mathbf{c}_{bg} T(s_f, t) + \int_{s_n}^{s_f} T(s, t) \sigma(\mathbf{r}(s), t) \mathbf{c}(\mathbf{r}(s), \omega, t) ds \quad (4)$$

$$T(s, t) = \exp\left(-\int_{s_n}^s \sigma(\mathbf{r}(\bar{s}), t) d\bar{s}\right) \quad (5)$$

The dynamic NeRF can therefore be trained to minimize the

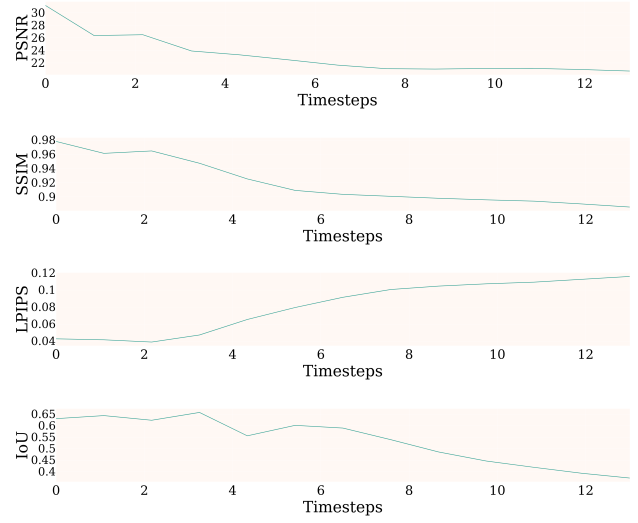


Figure 5: PAC-NeRF evaluation on *duck*. Both rendering quality and IoU over occupancy grids degrade as the simulation progresses because the neural radiance field reconstruction gets increasingly constrained by the physics-based losses. Metrics are only computed on the masked foreground object.

rendering loss:

$$\mathcal{L}_{\text{render}} = \frac{1}{N} \sum_{i=0}^{N-1} \frac{1}{|\mathcal{R}|} \sum_{\mathbf{r} \in \mathcal{R}} \|\mathbf{C}(\mathbf{r}, t_i) - \hat{\mathbf{C}}(\mathbf{r}, t_i)\|^2 \quad (6)$$

where N is the number of frames of videos, $\hat{\mathbf{C}}(\mathbf{r}, t_i)$ is the ground truth color observation.

PAC-NeRF first initializes an Eulerian voxel field over the first frame of the sequence. It then uses a grid-to-particle conversion method to obtain a Lagrangian particle field; this is advected, enforcing that the appearance and volume density fields admit conservation laws characterized by the velocity field \mathbf{v} of the underlying physical system:

$$\frac{D\sigma}{Dt} = 0, \quad \frac{D\mathbf{c}}{Dt} = 0 \quad (7)$$

with $\frac{D\phi}{Dt} = \frac{\partial\phi}{\partial t} + \mathbf{v} \cdot \nabla\phi$ being the material derivative of an arbitrary time-dependent field $\phi(\mathbf{x}, t)$. Moreover, the velocity field must obey momentum conservation for continuum materials:

$$\rho \frac{D\mathbf{v}}{Dt} = \nabla \cdot \mathbf{T} + \rho \mathbf{g} \quad (8)$$

where ρ is the physical density field, \mathbf{T} is the internal Cauchy stress tensor, and \mathbf{g} is the acceleration due to gravity and it is evolved using a differentiable Material Point Method (MPM) [HFG*18]. The advected field is then mapped back to the Eulerian domain using the particle-to-grid conversion and is used for collision handling and neural rendering. We refer to the original paper for more technical details.

Experiment and results We trained PAC-NeRF on *duck* by subsampling a 1s simulation of 1800 frames down to 13 frames with

Table 4: Analysis of resource consumption in terms of CPU memory, GPU memory, scene size and time, varying rendering and simulation resolution. Values are averaged over 20 repeated runs over the *dam* scene from the multi-view dataset. Rendering is disabled while gathering the simulation data. Geometry is exported in sparse density format.

	Rendering resolution				Simulation resolution			
	800 × 800	FHD	2K	4K	64 ³	128 ³	256 ³	512 ³
CPU Memory (MB)	374.4	374.4	374.4	374.4	5.6	46.8	374.4	2995.2
GPU Memory (MB)	1242.4	1244.4	1246.4	1248.4	148.6	270.8	1242.4	9015.2
Exported (MB)	780.35	861.44	912.94	983.42	11.12	83.12	653.2	4110.02
Time (s)	16.97	20.46	21.13	22.48	5.59	7.82	16.05	112.8

a Δt of $\approx 0.077s$. Figure 4 visually confirms that our model generates accurate renders of simulations, validating our multi-view data generation for use in inverse rendering techniques within fluid dynamics. PAC-NeRF applies stricter advection constraints to the NeRF model compared to other 4D NeRF methods. This enhances scene dynamics recognition and yields smoother, more realistic inter-frame reconstructions. However, it may compromise the supervised frame reconstruction quality over time, unlike other 4D NeRF methods which maintain stable reconstruction quality but may result in less physically accurate interpolations between simulation steps. This behaviour is depicted in Figure 5, where we show the degradation of performance over time for PAC-NeRF on the *duck* scene. The figure reports the variation over time of the Peak Signal-to-Noise Ratio (PSNR), the Structural Similarity Index Measure (SSIM) [WBSS04], the Learned Perceptual Image Patch Similarity (LPIPS) [ZIE*18], and the Intersection over Union (IoU) between the object masks.

5.3. Generation performance

A major benefit of our framework is the ability to describe a large-scale dataset by means of just a few MB worth of meshes and JSON configuration files. This not only makes easy to share and distribute the data, but also to extend the dataset with new scenes. However, for this to be a real advantage, producing the simulation data must be a faster procedure than sharing them.

We measure the runtime for generating the *obstacles* and *dam-break* datasets, and we summarize the results in Table 2. All scenes are rendered from a single viewpoint at low resolution (800 × 800), to offer visual support to the generated data. Our procedure reaches a throughput of ~ 6.5 frames per second, allowing to generate 100k frames of data in ~ 4 hours. We further analyze our procedure’s runtime and memory efficiency by collecting performance data over multiple runs, varying simulation and rendering resolutions. The results are reported in Table 4: exploiting GPU programming massively benefits both the simulation and rendering steps, so much so that rendering 150 frames even in 4K only accounts for 28% of the total computation time. The data also motivates our choice of sparse format for storing density fields: while 4GB per scene may seem a lot at simulation resolution 512, the dense counterpart would require about 80GB per scene (150 · 512³ floats), *i.e.*, only 5% of the disk memory.

All generations were run on medium-end consumer hardware, to further solidify the claim that our data can be re-generated locally,

without the need to share entire datasets. Our machine runs Windows 11 over 32GB of DDR4 RAM, an intel core i7 12700K CPU (3.6GHz), and a NVIDIA RTX4070Ti GPU (7680 CUDA cores). The data was generated on an SSD drive to minimize disk latency.

6. Conclusions

We introduce a novel framework for the generation large-scale multimodal fluid simulation data. Our pipeline relies on an efficient implementation of the Lattice-Boltzmann method and a GPU renderer, both encapsulated inside an intuitive, extensible, and simple interface that allows describing families of fluid dynamics scenes just by means of a simple configuration file. We have used our framework to introduce three benchmark datasets for research, and evaluated them on state-of-the-art methods for data-driven fluid simulation and inverse rendering tasks, demonstrating the potential value of our contribution. Finally, we tested the efficiency of our pipeline, and proved that it can be suitable for the fast generation of large-scale datasets, effectively avoiding the problem of sharing large volumes of data. In the future, we plan of further extending our work by introducing the possibility of simulating other types of fluids (*e.g.*, non-Newtonian fluids, ferrofluids), as well as expanding the set of pre-made scene templates and configurations.

References

- [BLDL20] BAI K., LI W., DESBRUN M., LIU X.: Dynamic upsampling of smoke through dictionary-based learning. *ACM Trans. Graph.* 40, 1 (sep 2020). URL: <https://doi.org/10.1145/3412360>, doi: 10.1145/3412360. 2
- [BME20] BERTICHE H., MADADI M., ESCALERA S.: Cloth3d: clothed 3d humans. In *European Conference on Computer Vision* (2020), Springer, pp. 344–359. 3
- [BRPMB17] BOGO F., ROMERO J., PONS-MOLL G., BLACK M. J.: Dynamic FAUST: Registering human bodies in motion. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (July 2017). 2
- [BWDL21] BAI K., WANG C., DESBRUN M., LIU X.: Predicting high-resolution turbulence details in space and time. *ACM Trans. Graph.* 40, 6 (dec 2021). URL: <https://doi.org/10.1145/3478513.3480492>, doi:10.1145/3478513.3480492. 2
- [CLZ*22] CHU M., LIU L., ZHENG Q., FRANZ E., SEIDEL H.-P., THEOBALT C., ZAYER R.: Physics informed neural fields for smoke reconstruction with sparse data. *ACM Transactions on Graphics* 41, 4 (aug 2022), 119:1–119:14. 2
- [CPA*21] CORONA E., PUMAROLA A., ALENYA G., PONS-MOLL G., MORENO-NOGUER F.: Smplicit: Topology-aware generative model for clothed people. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2021), pp. 11875–11885. 3

- [DYZ*23] DENG Y., YU H.-X., ZHANG D., WU J., ZHU B.: Fluid simulation on neural flow maps. *ACM Transactions on Graphics (TOG)* 42, 6 (2023), 1–21. 2
- [EUT19] ECKERT M.-L., UM K., THUEREY N.: Scalarflow: A large-scale volumetric data set of real-world scalar transport flows for computer animation and machine learning. *ACM Trans. Graph.* 38, 6 (nov 2019). URL: <https://doi.org/10.1145/3355089.3356545>, doi:10.1145/3355089.3356545. 2, 3
- [FSD*20] FERDIAN E., SUINESIAPUTRA A., DUBOWITZ D. J., ZHAO D., WANG A., COWAN B., YOUNG A. A.: 4dflownet: Super-resolution 4d flow mri using deep learning and computational fluid dynamics. *Frontiers in Physics* 8 (2020). URL: <https://www.frontiersin.org/articles/10.3389/fphy.2020.00138>, doi:10.3389/fphy.2020.00138. 2
- [GDWY22] GUAN S., DENG H., WANG Y., YANG X.: Neurofluid: Fluid dynamics grounding with particle-driven neural radiance fields. In *ICML* (2022). 2
- [GKY*16] GRAHAM J., KANOV K., YANG X. I. A., LEE M., MALAYA N., LALESCU C. C., BURNS R., EYINK G., SZALAY A., MOSER R. D., MENEVEAU C.: A web services accessible database of turbulent channel flow and its use for testing a new integral wall model for les. *Journal of Turbulence* 17, 2 (2016), 181–215. URL: <https://doi.org/10.1080/14685248.2015.1088656>, arXiv:<https://doi.org/10.1080/14685248.2015.1088656>, doi:10.1080/14685248.2015.1088656. 3
- [GSW21] GAO H., SUN L., WANG J.-X.: Super-resolution and denoising of fluid flow using physics-informed convolutional neural networks without high-resolution labels. *Physics of Fluids* 33, 7 (2021). 2
- [HFG*18] HU Y., FANG Y., GE Z., QU Z., ZHU Y., PRADHANA A., JIANG C.: A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Transactions on Graphics* 37, 4 (2018), 150. 7
- [HHL*05] HOYER K., HOLZNER M., LÜTHI B., GUALA M., LIBERZON A., KINZELBACH W.: 3d scanning particle tracking velocimetry. *Experiments in Fluids* 39 (2005), 923–934. 2
- [JBN*23] JANNY S., BÉNÉTEAU A., NADRI M., DIGNE J., THOME N., WOLF C.: EAGLE: Large-scale learning of turbulent fluid dynamics with mesh transformers. In *The Eleventh International Conference on Learning Representations* (2023). URL: <https://openreview.net/forum?id=mfIX4QpsARJ>. 3
- [JGG20] JAKOB J., GROSS M., GÜNTHER T.: A fluid flow data set for machine learning and its application to neural flow map interpolation. *IEEE Transactions on Visualization and Computer Graphics* 27, 2 (2020), 1279–1289. 3
- [KAT*19] KIM B., AZEVEDO V. C., THUEREY N., KIM T., GROSS M., SOLENTHALER B.: Deep fluids: A generative network for parameterized fluid simulations. *Computer Graphics Forum* 38, 2 (2019), 59–70. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13619>, arXiv:<https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13619>, doi:<https://doi.org/10.1111/cgf.13619>. 2
- [KISS15] KEINERT B., INNMANN M., SÄNGER M., STAMMINGER M.: Spherical fibonacci mapping. *ACM Transactions on Graphics (TOG)* 34, 6 (2015), 1–7. 1, 5
- [KOH*24] KE B., OBUKHOV A., HUANG S., METZGER N., DAUDT R. C., SCHINDLER K.: Repurposing diffusion-based image generators for monocular depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2024), pp. 9492–9502. 3
- [KWP*24] KLEIN J., WALLER R., PIRK S., PAŁUBICKI W., TESTER M., MICHELS D. L.: Synthetic data at scale: a development model to efficiently leverage machine learning in agriculture. *Frontiers in Plant Science* 15 (2024), 1360113. 3
- [Leh] LEHMANN M.: Fluidx3d. URL: <https://github.com/ProjectPhysX/FluidX3D>. 2, 3
- [LF22] LI Z., FARIMANI A. B.: Graph neural network-accelerated lagrangian fluid simulation. *Computers & Graphics* 103 (2022), 201–211. 2
- [LFBC23] LINO M., FOTIADIS S., BHARATH A. A., CANTWELL C. D.: Current and emerging deep-learning methods for the simulation of fluid dynamics. *Proceedings of the Royal Society A* 479, 2275 (2023), 20230058. 1
- [LM22] LI M., MCCOMB C.: Using physics-informed generative adversarial networks to perform super-resolution for multiphase fluid simulations. *Journal of Computing and Information Science in Engineering* 22, 4 (2022), 044501. 2
- [LMR*15] LOPER M., MAHMOOD N., ROMERO J., PONS-MOLL G., BLACK M. J.: SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)* 34, 6 (Oct. 2015), 248:1–248:16. 2
- [LPW*08] LI Y., PERLMAN E., WAN M., YANG Y., MENEVEAU C., BURNS R., CHEN S., SZALAY A., EYINK G.: A public turbulence database cluster and applications to study lagrangian evolution of velocity increments in turbulence. *Journal of Turbulence* 9, 31 (2008). 3
- [LQC*23] LI X., QIAO Y.-L., CHEN P. Y., JATAVALLABHULA K. M., LIN M., JIANG C., GAN C.: PAC-neRF: Physics augmented continuum neural radiance fields for geometry-agnostic system identification. In *The Eleventh International Conference on Learning Representations* (2023). URL: <https://openreview.net/forum?id=tVkrbkz42vc>. 2, 6, 7
- [LTT*21] LI Y., TAKEHARA H., TAKETOMI T., ZHENG B., NIESSNER M.: 4dcomplete: Non-rigid motion estimation beyond the observable surface. *IEEE International Conference on Computer Vision (ICCV)* (2021). 2
- [MKH*23] MAGGIOLI F., KLEIN J., HÄDRICH T., RODOLÀ E., PAŁUBICKI W., PIRK S., MICHELS D. L.: A physically-inspired approach to the simulation of plant wilting. In *SIGGRAPH Asia 2023 Conference Papers* (2023), pp. 1–8. 3
- [MST*20] MILDENHALL B., SRINIVASAN P. P., TANCIK M., BARRON J. T., RAMAMOORTHY R., NG R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV* (2020). 2, 7
- [PBLM07] PERLMAN E., BURNS R., LI Y., MENEVEAU C.: Data exploration of turbulence simulations using a database cluster. In *SC '07: Proceedings of the 2007 ACM/IEEE Conference on Supercomputing* (2007), Association for Computing Machinery. 3
- [PFSGB21] PFAFF T., FORTUNATO M., SANCHEZ-GONZALEZ A., BATTAGLIA P.: Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations* (2021). URL: https://openreview.net/forum?id=roNqYL0_XP. 1, 3
- [PLPM20] PATEL C., LIAO Z., PONS-MOLL G.: Tailormet: Predicting clothing in 3d as a function of human pose, shape and garment style. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (jun 2020), IEEE. 3
- [PUKT22] PRANTL L., UMMENHOFER B., KOLTUN V., THUEREY N.: Guaranteed conservation of momentum for learning particle-based fluid dynamics. In *Conference on Neural Information Processing Systems* (2022). 2, 3
- [RRR*21] ROBERTS M., RAMAPURAM J., RANJAN A., KUMAR A., BAUTISTA M. A., PACZAN N., WEBB R., SUSSKIND J. M.: Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *Proceedings of the IEEE/CVF international conference on computer vision* (2021), pp. 10912–10922. 3
- [SFK*22] STACHENFELD K., FIELDING D. B., KOCHKOV D., CRANMER M., PFAFF T., GODWIN J., CUI C., HO S., BATTAGLIA P., SANCHEZ-GONZALEZ A.: Learned simulators for turbulence. In *International Conference on Learning Representations* (2022). URL: <https://openreview.net/forum?id=msRBojTz-Nh>. 3
- [SGT*08] SCARSELLI F., GORI M., TSOI A. C., HAGENBUCHNER M., MONFARDINI G.: The graph neural network model. *IEEE transactions on neural networks* 20, 1 (2008), 61–80. 3

- [TA23] TOSHEV A. P., ADAMS N. A.: Lagrangebench datasets, Oct. 2023. URL: <https://zenodo.org/doi/10.5281/zenodo.10021925>, doi:10.5281/zenodo.10021925. 3
- [TGF*23] TOSHEV A. P., GALLETTI G., FRITZ F., ADAMI S., ADAMS N. A.: Lagrangebench: A lagrangian fluid mechanics benchmarking suite. In *37th Conference on Neural Information Processing Systems (NeurIPS 2023) Track on Datasets and Benchmarks* (2023). URL: <https://arxiv.org/abs/2309.16342>. 3
- [UPTK20] UMMENHOFER B., PRANTL L., THUEREY N., KOLTUN V.: Lagrangian fluid simulation with continuous convolutions. In *International Conference on Learning Representations* (2020). 1, 2, 3, 5, 6
- [VB22] VINUESA R., BRUNTON S. L.: Enhancing computational fluid dynamics with machine learning. *Nature Computational Science* 2, 6 (2022), 358–366. 2
- [WBSS04] WANG Z., BOVIK A. C., SHEIKH H. R., SIMONCELLI E. P.: Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing* 13, 4 (2004), 600–612. 8
- [WBT19] WIEWEL S., BECHER M., THUEREY N.: Latent space physics: Towards learning the temporal evolution of fluid flow. In *Computer graphics forum* (2019), vol. 38, Wiley Online Library, pp. 71–82. 2
- [WKA*20] WIEWEL S., KIM B., AZEVEDO V., SOLENTHALER B., THUEREY N.: Latent space subdivision: Stable and controllable time predictions for fluid flow (2020). *arXiv preprint arXiv:2003.08723* (2020). 2
- [WXCT19] WERHAHN M., XIE Y., CHU M., THUEREY N.: A multi-pass gan for fluid flow super-resolution. *Proc. ACM Comput. Graph. Interact. Tech.* 2, 2 (jul 2019). URL: <https://doi.org/10.1145/3340251>, doi:10.1145/3340251. 2
- [WYC*22] WANG Z., YANG W., CAO J., XU L., MING YU J., YU J.: Neref: Neural refractive field for fluid surface reconstruction and implicit representation. *ArXiv abs/2203.04130* (2022). URL: <https://api.semanticscholar.org/CorpusID:247315295>. 2
- [XFCT18] XIE Y., FRANZ E., CHU M., THUEREY N.: tempogan: A temporally coherent, volumetric gan for super-resolution fluid flow. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 95. 2
- [XZB24] XU B., ZHOU Y., BIAN X.: Self-supervised learning based on transformer for flow reconstruction and prediction. *Physics of Fluids* 36, 2 (2024). 2
- [ZIE*18] ZHANG R., ISOLA P., EFROS A. A., SHECHTMAN E., WANG O.: The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), pp. 586–595. 8