DOI: 10.1111/cgf.13522
ACM SIGGRAPH / Eurographics Symposium on Computer Animation 2018
T. Beeler and N. Thuerey
(Guest Editors)

Volume 37 (2018), Number 8

Liquid Splash Modeling with Neural Networks

Kiwon Um, Xiangyu Hu, and Nils Thuerey

Technical University of Munich, Germany

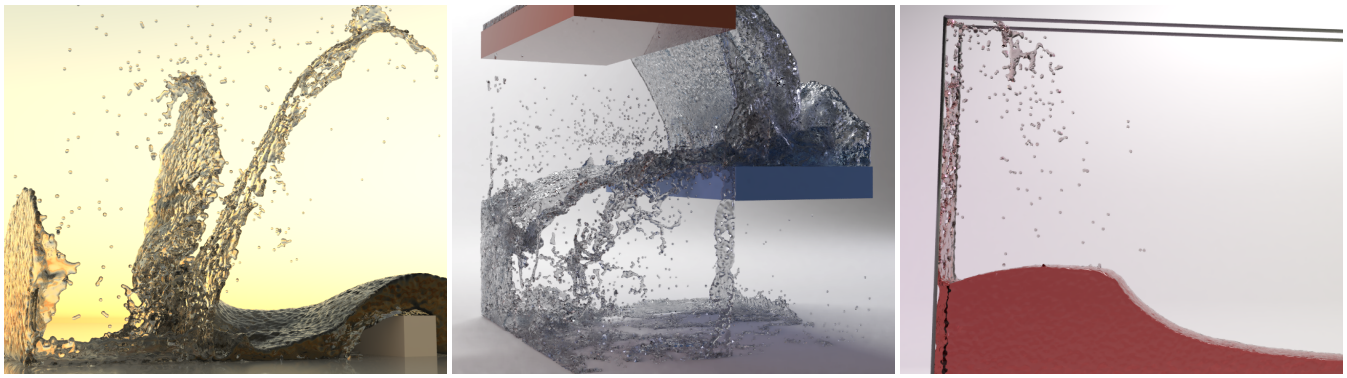

Figure 1: *Our data-driven splash model improves the visual fidelity of FLIP simulations as shown here with three different simulation setups. Our model learns to infer the probability and velocity changes of under-resolved droplet formation effects with the trained neural networks.*

Abstract

This paper proposes a new data-driven approach to model detailed splashes for liquid simulations with neural networks. Our model learns to generate small-scale splash detail for the fluid-implicit-particle method using training data acquired from physically parametrized, high resolution simulations. We use neural networks to model the regression of splash formation using a classifier together with a velocity modifier. For the velocity modification, we employ a heteroscedastic model. We evaluate our method for different spatial scales, simulation setups, and solvers. Our simulation results demonstrate that our model significantly improves visual fidelity with a large amount of realistic droplet formation and yields splash detail much more efficiently than finer discretizations.

CCS Concepts

•***Computing methodologies*** → ***Physical simulation;*** *Supervised learning by regression;*

1. Introduction

For large-scale liquid simulations, it is crucial for visual fidelity that a numerical simulation can produce a sufficient number of very small droplets [GB13, UHT17]. However, it is difficult to capture such splashes in practical simulations due to their complex small-scale surface geometry and dynamics. A typical remedy for this difficulty is to use additional representations that are physically or heuristically inspired to improve the visual realism of the underlying simulations [TFK*03, KCC*06, IAAT12]. As an alternative, small drops can be also tracked as part of the underlying simulation method [GSLF05]. However, accurately resolving the effects of drop formation typically requires the use of very fine spatial discretization, which in turn leads to high computational cost. Thus, it is often challenging to generate vivid splashes in liquid simulations as they require resolving the small-scale dispersive motions that lead to droplets forming and being ejected from the bulk volume.

This paper proposes a new data-driven splash generation approach that improves the visual fidelity of hybrid particle-grid liquid simulations. By learning the formation of small-scale splashes from physically accurate simulations, our model effectively approximates the sub-grid scale effects that lead to droplet generations. This enables us to generate realistic splashes in coarse sim-

ulations without the need for manually tweaking parameters or increased computational costs induced by high resolution simulations. We realize our model using machine learning techniques with neural networks (NNs) and integrate the model into the fluid-implicit-particle (FLIP) algorithm [ZB05]. Within this environment, we investigate the generality of our design by employing different simulation methods for generating the training data. Moreover, we show an extension of our model to generate secondary particles, which are independent from the underlying simulation. Using this extension, we also investigate controlling the number of generated splashes and the ability of our model to learn from multi-scale data. Figure 1 shows three examples of results that our model can generate. In the following, we will refer to our model as *MLFLIP*, which indicates a combination of machine learning and FLIP.

2. Related Work

The behavior of liquids is typically modeled with the *Navier-Stokes* (NS) equations:

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = \mathbf{g} - \frac{1}{\rho} \nabla P + \nu \nabla^2 \mathbf{u} \quad \text{and} \quad \nabla \cdot \mathbf{u} = 0, \qquad (1)$$

where $\mathbf{u}$ is the velocity, $\mathbf{g}$ is the gravity, ρ is the density, P is the pressure, and ν is the viscosity coefficient. There exist many numerical methods for solving these equations, which are commonly categorized as Eulerian and Lagrangian approaches [Bri15, IOS*14].

FLIP is a particularly popular method for liquid simulations [ZB05], and it is widely used in movie visual effects at the moment. Its effective combination of Lagrangian and Eulerian properties enables the efficient solve of liquid motions. While FLIP has become a practical solution for liquid simulations, the core method has been extended to various ways in order to improve its simulation quality and efficiency. For example, different position correction methods improved the distribution of particles [ATT12, UBH14], and Ferstl *et al.* [FAW*16] proposed a narrow band method that improves the efficiency by sampling the volume with particles only near the surface. In addition, FLIP has been widely adopted to various problems of fluid simulations such as viscous free surfaces [BB08] and solid-fluid coupling [BBB07]. Two-phase fluids have been also simulated via an extension of FLIP [BB12]. Moreover, the affine particle-in-cell (APIC) method [JSS*15] as a variant effectively addressed the stability issues of FLIP and the dissipation of the particle-in-cell method, and APIC has been further generalized to polynomial representations [FGG*17].

A central goal of our model is to improve the visual fidelity of liquid simulations with small-scale details. Apart from FLIP, the smoothed particle hydrodynamics (SPH) approach is a popular alternative in graphics [MCG03, SP09, AAT13, BK17]. Due to the nature of its computational mechanism based on interparticle interactions, SPH is able to simulate dispersive motions and droplets. This behavior led to a combination of SPH and the particle level set approach [EMF02]; this simulates the diffuse regions via SPH [LTKF08]. Our model infers such small-scale interactions in a data-driven way. In addition, we employ an efficient hybrid particle-grid solver (i.e., FLIP). Thus, our method does not require potentially expensive particle neighborhood information. A possible alternative approach to achieve this goal in FLIP is to add details using extra representations for diffuse materials [YLHQ14, YLX*15]. Likewise, Ihmsen *et al.* [IAAT12] proposed a flexible secondary particle model for such diffuse materials in SPH simulations. With enough manual tuning and elaborate coupling processes, these extra representation approaches can yield realistic results, but in contrast to their work, we focus on an automated approach that captures splash effects for physically-parametrized real world scales. Our model does not require any parameter tuning on the user side and sophisticated integration. At the same time, one of our goals is to demonstrate that NNs are suitable to detect and generate these splashes.

A method that shares our goal to enable splashes with FLIP is the unilateral incompressibility (UIC) solver [GB13]. The UIC solver allows positive divergence in fluid cells such that it can create larger-scale splashes. However, it leads to a very different visual behavior as the grid based velocities lead to a formation of sheets and filaments rather than detaching droplets. In addition, the UIC approach requires two solves of the linear complementarity problem. Our approach targets a very different direction. Instead of modifying the pressure solve, we incorporate a statistical model with the help of machine learning. We note that our approach is orthogonal to the choice of Poisson solver and thus could also integrate into the UIC solver in a single simulation to increase the small-scale splash effects.

Machine learning: Machine learning is a field that recently received substantial attention due to the success of so-called deep neural networks. Here, especially the seminal image classification work of Krizhevsky *et al.* [KSH12] has to be mentioned, but other areas such as object detection [GDDM14] and control of complex systems based on visual inputs [MKS*13] have likewise seen impressive advances. As our splash model utilizes NNs, we briefly review their concepts and give an overview of previous work on NNs for physics simulations. In general, the learning process aims for the approximation of a general function f using a given data set (i.e., input $\mathbf{x}$ and output $\mathbf{y}$) in the form of $\mathbf{y} = f(\mathbf{x}, \mathbf{w})$ where $\mathbf{w}$ is the set of weights and biases to be trained. Using NNs, the general function f is modeled by networks of multiple layers where each layer contains multiple nodes (i.e., artificial neurons). These networks consist of layers with connected nodes. The output vector $\mathbf{y}_L$ from a layer L is typically computed with $\mathbf{y}_L = \Phi(\mathbf{w}_L \mathbf{y}_{L-1} + \mathbf{b}_L)$ where $\Phi(\cdot)$ is the activation function that is applied to each component, $\mathbf{w}_L$ is the weight matrix of the layer, and $\mathbf{b}_L$ is the bias vector of the layer. Here, the activation function Φ enables the networks to capture non-linearities in the approximated function. We will demonstrate that NNs, which so far have rarely been used for fluid simulations, can be used for realization of our data-driven splash model.

NNs were previously used to compute local pressure approximations [YYX16] while others employed networks with convolutional layers to regress the whole pressure field [TSSP17]. Moreover, an approach using regression forests, which represent an alternative to NNs, was proposed to efficiently compute forces in SPH [LJS*15]. More recently, NNs were also successfully employed for patch-based smoke simulations [CT17], super-resolution with temporal coherence [XFCT18], and fast generation of liquid animations with

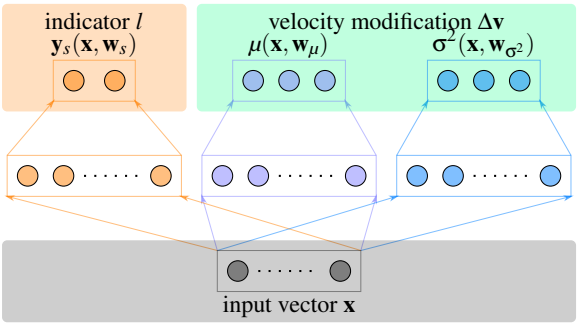

Figure 2: *The overall structure of our neural networks.*

space-time deformations [BPT17]. In the engineering community, approximating effects smaller than the discretization resolution is known as *coarse-grained modeling* [HEVEDB09], but this idea has not been used to model splash formation. We propose to use machine learning techniques to represent accurate and high resolution data in order to approximate complex small-scale effects with high efficiency.

3. Data-driven Splash Modeling

The following section details our data-driven approach for generating splashes. The principal idea of our approach is to infer statistics about splash formation based on data from simulations that are parametrized to capture the droplet formation in nature. Our definition of a *splash* is a small disconnected region of liquid material that is not globally coupled with the main liquid volume. The key novelty of our approach is that it does not require manually chosen parameters, such as velocity or curvature thresholds, to generate the splashes. Rather, we use a statistical model and data extracted from a series of highly detailed and pre-computed simulations.

Our approach consists of two components: a *detachment classification* and a *velocity modification*. Based on a feature descriptor consisting of localized flow information, the classifier predicts whether a certain volume of liquid forms a detached droplet within a chosen duration Δt (typically, on the order of a single frame of animation). For droplets that are classified as such, our modifier predicts its future velocity based on a probability distribution for velocity modifications. We use NNs to represent both components, and the following sections describe our statistical model and the corresponding machine learning approach.

3.1. Neural Network Model

The input to our model is a feature descriptor $\mathbf{x} \in \mathbb{R}^M$ that encapsulates enough information of the flow such that a machine learning model can infer whether the region in question will turn into a splash. For this binary decision "*splash* or *non-splash*", we will use an indicator value $l \in \{1,0\}$ in the following. Each descriptor is typically generated for a position $\mathbf{p}$. The M individual components of $\mathbf{x}$ consist of flow motion and geometry in the vicinity of $\mathbf{p}$. In practice, we use 3^3 samples of the velocity and level set. The discussion of this choice will be given in Section 3.3 in more detail.

We train our models with a given data set that consists of feature vectors $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$ and corresponding detachment indicator values $\mathbf{L} = \{l_1, l_2, \cdots, l_N\}$; they are generated during a preprocessing phase at locations $\{\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_N\}$. Then, our classifier aims for inferring the probability P_s that a feature vector $\mathbf{x}_i$ is in the class indicated by l_i. Considering a probability distribution function $\mathbf{y}_s$ that P_s follows, we will approximate the function $\mathbf{y}_s$ from the given data. For this task, we can follow established procedures from the machine learning field [Bis06].

The probability distribution $\mathbf{y}_s(\mathbf{x}_i, \mathbf{w}_s)$ is the target function that is represented by the weights $\mathbf{w}_s$. The weights are the actual degrees of freedom that will be adjusted in accordance to the data during the learning phase. We can express P_s in terms of $\mathbf{y}_s$ as:

$$P_s(l_i|\mathbf{x}_i) \sim P\left(l_i|\mathbf{y}_s(\mathbf{x}_i, \mathbf{w}_s)\right), \tag{2}$$

which yields the following likelihood function that we wish to maximize:

$$L_d(\mathbf{L}|\mathbf{X}) = \prod_{i=1}^{N} P\left(l_i|\mathbf{y}_s(\mathbf{x}_i, \mathbf{w}_s)\right). \tag{3}$$

In order to maximize this likelihood, we use the well-established *softmax* (i.e., normalized exponential function) for the loss of our classification networks. It will successfully encode the given data set and train the model for $\mathbf{y}_s$, and then we can evaluate with new feature vectors at any position in a flow to predict whether the region will turn into a detached droplet within the time frame Δt.

Let $\Delta\mathbf{v}$ be an instance of a velocity change for a splash with respect to the motion of the bulk liquid in its vicinity. We will afterward consider this velocity change of a droplet relative to the bulk as the *velocity modification* of a particle in our simulation. Similar to the classifier above, our goal is to infer the set of velocity modifications $\Delta\mathbf{V} = \{\Delta\mathbf{v}_1, \Delta\mathbf{v}_2, \cdots, \Delta\mathbf{v}_N\}$ based on the corresponding set of feature vectors $\mathbf{X}$. From the statistics of our training data, we found that it is reasonable to assume that the velocity modifications follow a normal distribution relative to the mean flow direction. Accordingly, we model the modifier as a modification function $f_m(\Delta\mathbf{v}_i|\mathbf{x}_i)$ following a normal distribution with mean μ and variance σ^2:

$$f_m(\Delta\mathbf{v}_i|\mathbf{x}_i) \sim \mathcal{N}\left(\Delta\mathbf{v}_i|\mu(\mathbf{x}_i, \mathbf{w}_\mu), \sigma^2(\mathbf{x}_i, \mathbf{w}_{\sigma^2})\right), \tag{4}$$

thus

$$f_m(\Delta\mathbf{v}_i|\mathbf{x}_i) \sim \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(\frac{-(\Delta\mathbf{v}_i - \mu_i)^2}{2\sigma_i^2}\right), \tag{5}$$

where, for the sake of simplicity, μ_i and σ_i^2 denote $\mu(\mathbf{x}_i, \mathbf{w}_\mu)$ and $\sigma^2(\mathbf{x}_i, \mathbf{w}_{\sigma^2})$, respectively. Then, the negative log likelihood function L_m (also known as loss function) for the given data is defined as follows:

$$L_m(\Delta\mathbf{V}|\mathbf{X}) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{d}\left[\frac{(\Delta\mathbf{v}_{i,j} - \mu_{i,j})^2}{\sigma_{i,j}^2} + \ln\sigma_{i,j}^2\right], \tag{6}$$

where j denotes the spatial index.

As Eq. (6) indicates, we model the modifier as a mean variance estimation (MVE) problem [NW94, KNCA11]. Instead of directly estimating the mean of targets, the MVE problem assumes

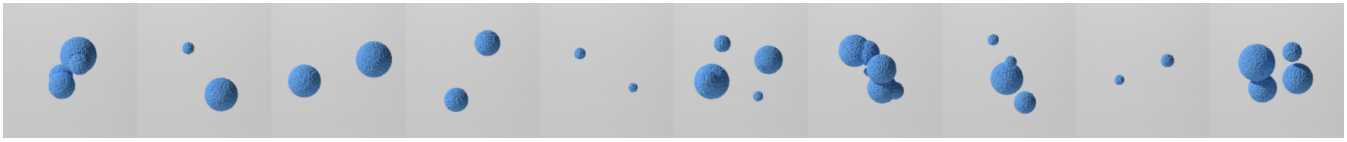

Figure 3: *Selected close-up snapshots of ten randomized initial conditions to generate the droplet formation data for training.*

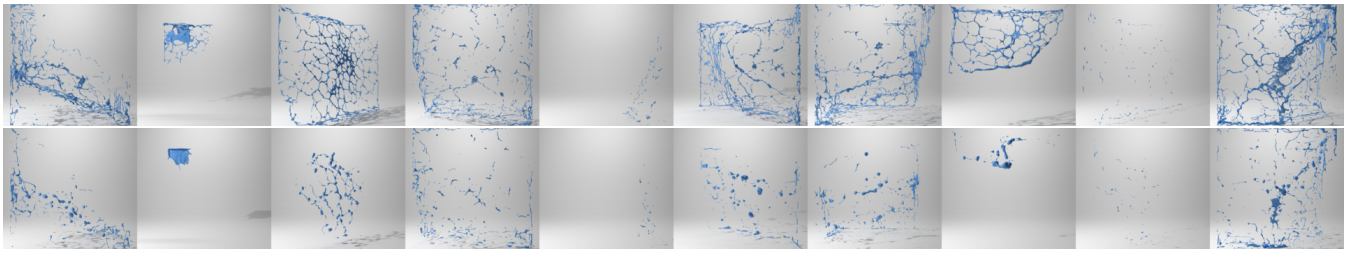

Figure 4: *Example frames of (top)* 5mm *and (bottom) 1.5*mm *scale simulations for training data generation.*

that errors are normally distributed around the mean and estimates both the mean and *heteroscedastic* variance. Note that $\mu(\mathbf{x}_i, \mathbf{w}_\mu)$ and $\sigma^2(\mathbf{x}_i, \mathbf{w}_{\sigma^2})$ are the target functions that are approximated with each set of weights $\mathbf{w}_\mu$ and $\mathbf{w}_{\sigma^2}$.

In our approach, the mean and variance functions are represented by NNs and approximated by estimating the two sets of weights such that they minimize the loss function L_m for the given data $\{\mathbf{X}, \Delta\mathbf{V}\}$. We want to point out that several machine learning algorithms are available to solve this problem [Bis06], but NNs have proven themselves to be robust for this problem, and we found that they work very well in our tests.

The overall structure of our NNs is illustrated in Figure 2. The NNs learn for two separate components: classifier and modifier. Sharing the input vector $\mathbf{x}$, both components are represented as separate two-layer NNs. The size of output from the first layer is double the input vector's, and the output is fully connected to the final output. All outputs from each layer are activated using the hyperbolic tangent function. Note that there is a large variety of different network layouts that could tried here, but we found that this simple structure worked sufficiently well in our tests.

3.2. FLIP Integration

Our NN-based splash generation model easily integrates into an existing FLIP simulation pipeline. After the pressure projection step, we run classification on all particles in a narrow band of one cell around the surface. For those that generate a positive outcome, we evaluate our velocity modification network to compute component-wise mean and variance. Then, we generate random numbers from correspondingly parametrized normal distributions to update the velocity of the new splash particle. All splashes are treated as ballistic particles and do not participate in the FLIP simulation until they impact the main volume again. Thus, we treat individual splashing droplets as particles that only experience gravitational acceleration but no other NS forces. This modeling is in line with the secondary effects often employed in movies [LB12].

Our model evaluates the probability of forming a splash in regard to a chosen duration Δt, which is typically larger than the simulation time-step size because of the stability constraint. Unless this difference is carefully considered, the time-step size will affect the inference results. As we use smaller time-step size so evaluate more frequently, we will get more splashes. To avoid this, we evaluate our inference using a stochastic process [PP02], i.e., random walk, scaling the expectation by time to match the desired duration. For a series of simulation steps Δt_k, which proceed for the chosen duration Δt (i.e., $\sum_k \Delta t_k = \Delta t$), we compute particle i's expectation E_i using two finite outcomes 1 (splash) and -1 (non-splash) with probabilities $\mathbf{y}_s$ that are evaluated from our trained model:

$$E_i = \sum_k \frac{1}{\sqrt{\Delta t / \Delta t_k}} \mathbf{y}_s(\mathbf{x}_{i,k}, \mathbf{w}_s) \cdot [1, -1]. \tag{7}$$

If the expectations of particles are positive after the random walks during Δt, we treat them as splashes and thus evaluate our velocity modification.

FLIP can use different numbers of simulation particles for the same grid resolution, and this will affect the number of inferred splashes since our model evaluates splashes for each simulation particle. In order to make the model robust to such a variety, we can normalize the inference process in space. While the expectations are evaluated for each particle, we can limit the maximum number of splashes per unit volume using the expected values E_i (e.g., one particle whose expectation is the largest value in a cell). We found that our approach generates a consistent number of splashes regardless of the size of the time-step and the number of simulation particles per grid cell.

Look-ahead correction: While our splash generation algorithm reliably works in our tests, we noticed a small chance of misclassification. This can happen, for instance, when the side of an obstacle is just outside the region of our input feature vector. To minimize the influence of such misclassifications, we implemented a look-ahead step that reverts the classification of individual splashes if the droplets do not manage to form the expected splashes. For this look-ahead check, we advance all bulk volume particles, i.e., those

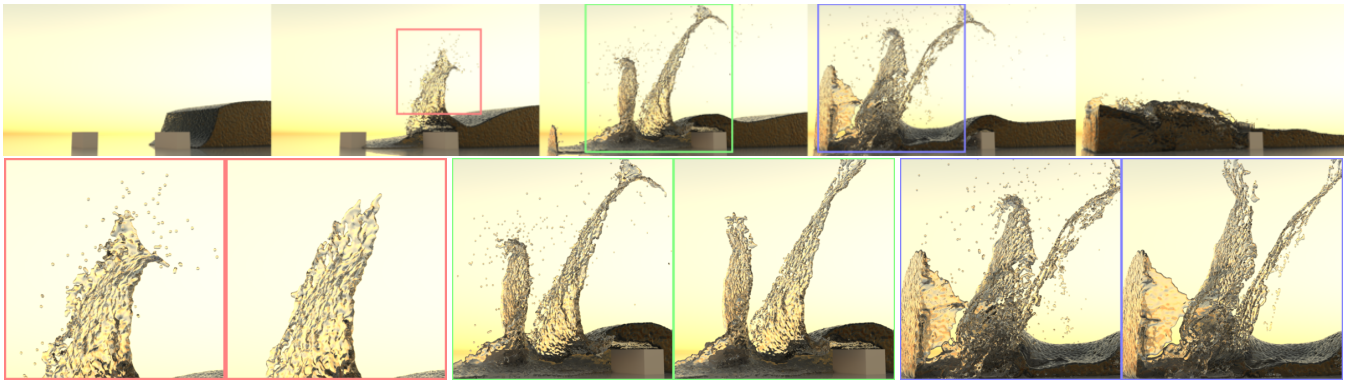

Figure 5: *Example frames of dam simulations with MLFLIP. The bottom row compares (left) MLFLIP with (right) FLIP side-by-side in the selected areas of three frames.*

that were not classified as splashes, by Δt using the current grid velocities. We separately integrate positions of the new splash particles for a time interval of Δt with their updated velocities. If a new splash particle ends up inside of the bulk liquid or an obstacle, we revert its classification and modification.

3.3. Training Data

Our NNs require a large set of input feature vectors with target outputs for training. In our model, the latter consists of the classification result l indicating whether a flow region forms a splash and the velocity modification $\Delta\mathbf{v}$ predicting the trajectory of a splash. We generate the training data from a series of simulations with randomized initial conditions designed to incite droplet formation. The randomized initial conditions are the number of droplets and their initial positions and velocities. We choose the ranges of each condition such that they yield sufficient variability for data generation. The snapshots of the randomized example simulations are shown in Figure 3. Note that at this stage any available "accurate" NS solver could be employed. However, we demonstrate that FLIP can *bootstrap* itself by using high resolution simulations with correctly parametrized surface tension.

For the training data simulations, it is crucial that they resolve important sub-grid effects that are not well resolved on coarse resolutions to which our model will be applied later. We test our model with two physical parametrizations where the surface tension is dominant, and thus many droplets are generated. Our training simulations are performed using FLIP with the ghost fluid method [HK05] for surface tension effects. The two scales use a grid spacing of 5mm and 1.5mm, and they are parametrized with the surface tension of water, i.e., 0.073 N/m. Both scales use a simulation grid of 100^3. Several example frames for both scales are shown in Figure 4.

Note that the simulation data for training will be used to encode the desired sub-grid effects for much larger scales afterwards. When applying our model in new simulation setups, we typically have scales that are much coarser than those used for generating the training data. Thus, the feature descriptor (i.e., $\mathbf{x}_i$) needs to be defined at this coarse simulation scale, and our networks need to infer their outputs (i.e., l_i and $\Delta\mathbf{v}_i$) based only on this coarse input. For this purpose, we make use of a coarse grid at data generation time. This coarse grid represents the scale to which the model will be applied afterward. For every time step, we down-sample the necessary high resolution fields from the data generation simulation to this lower resolution and extract the feature descriptors for training.

As indicated in Section 3.1, we define a feature descriptor $\mathbf{x}_i$ using 3^3 samples of the velocity and level set values interpolated with a sampling spacing of h; i.e., the feature vector consists of 108 components containing $3^3 \times 3$ velocity values and $3^3 \times 1$ level set values. Because the splash formation mostly relies on the local flow physics near the liquid surface, we focus on the localized flow information and the surface region where the splash is likely initiated. From pilot experiments, we found that the improvement is negligible when more samples or more features such as obstacle information are used for the feature vector.

In order to extract the splash indicator value l, we analyze the particle's spatial motion for a chosen duration Δt (i.e., a single frame of animation in our experiments). Using an auxiliary grid, the separate volumes are recognized by computing the isolated liquid regions from the level set field or cell flags. We then identify the splashing particles (i.e., l=1) as those ending up in a new disconnected component that falls below a given volume threshold at time $t + \Delta t$. In our case, if a disconnected component consists of less than 8 cells, the volume is marked as droplet volume. All particles in such a droplet are marked as splashes if the droplet did not exist as a disconnected component at time t.

The velocity modifier of our model predicts the trajectory for a splash. We evaluate this prediction after updating the velocity of particle from the divergence-free velocity at the target resolution. Thus, the new velocity $\mathbf{v}_m^{t+\Delta t}$ for a splash particle is defined as $\mathbf{v}_m^{t+\Delta t} = \mathbf{v}^{t+\Delta t} + \Delta\mathbf{v}$, and we compute the velocity modification $\Delta\mathbf{v}$ for training:

$$\Delta\mathbf{v} = \frac{\mathbf{p}^{t+\Delta t} - \mathbf{p}^t}{\Delta t} - \mathbf{v}^{t+\Delta t} \tag{8}$$

where, intuitively, the first term on the right side represents the velocity of the training resolution, and the second term represents the down-sampled velocity evaluated at the target resolution.

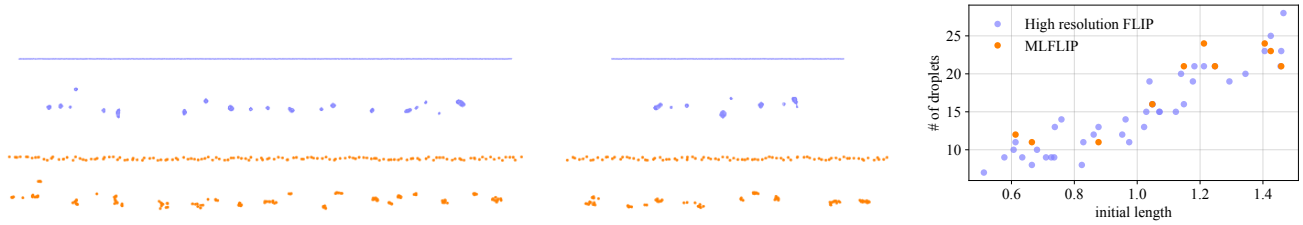

Figure 6: *Number of droplets formed after 1.5 seconds with different initial lengths. (a) Two selected high resolution FLIP simulations (top row in blue) show the surface tension driven dynamics. Two examples of our MLFLIP simulations (bottom row in orange) successfully recover the break-up in a low resolution simulation. Droplet count statistics are shown in (b). Our simulations (orange dots) very accurately match the full resolution simulation results (blue dots).*

As the splashes are initiated near the liquid surface in general, we extract the training data only from the surface particles. The surface particles are recognized by slightly expanding the area of empty (i.e., air) cells. Note that we use the data from splashing as well as non-splashing particles for training. It is crucial for training that the networks see sufficiently large numbers of samples from both classes.

For training, we used 1.3M samples of the 5mm scale and 441K samples of the 1.5mm scale. They were extracted from sixteen training simulations per scale. The data of each scale contain the same number of both splashing and non-splashing samples. Note that the same set of initial conditions (Figure 3) generates different physics in different scales (Figure 4). The smaller scale incites less splashes because the surface tension forces are more prominent. This resulted in a reduced number of training samples for the smaller scale of 1.5mm.

3.4. Training Networks

We randomly split the training samples into 75% for the *training set* and 25% for the *test set*. The graphs in Figure 7 illustrate the progress of the learning phase in our experiments. The training performed for 60K iterations with 5K samples as a training batch. When the data sets were fully used, we randomly shuffled the samples and then continued with the training. In order to train our MVE model, the first 30K iterations trained the mean, keeping the variance constant, afterwards the remaining 30K iterations trained both the mean and variance simultaneously [NW94]. Thus, as shown in the right of Figure 7, we could observe two learning phases: the first

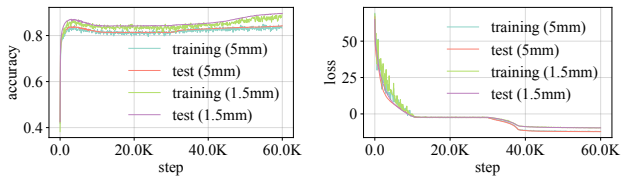

Figure 7: *Learning graphs for both the training and test sets in two scales. The left graph shows the classification accuracy; the right graph shows the loss L_m.*

intermediate convergence of the mean value after ca. 15K iterations and the final convergence approximately after 40K iterations.

To realize our NNs, we employed the *TensorFlow* framework [ten16] with its ADAM optimizer [KB14]. Here, the learning rate for training was set to 10^{-4}. We also employed weight decay and dropout (both with strength 10^{-1}) to avoid over-fitting. Additionally, we found that the batch normalization technique [IS15] significantly improves the learning rate and accuracy.

4. Model Evaluation

Our data-driven splash model (i.e., MLFLIP) incites the formation of splashes by inferring the likelihood and velocity modification of particle. As outlined in Section 3.3, our approach can employ any available NS solver to generate the training data. In order to demonstrate that our approach is agnostic to the choice of solver, we trained our model with two sets of data that were generated using two different simulation methods: FLIP and SPH. The FLIP data were generated using randomized initial conditions (Figure 18-(c)) similar to the three-dimensional example (Figure 3) with surface tension effects to incite splashes. On the other hand, SPH often results in a significant number of splashes due to its direct inter-particle interactions. Our SPH training data were generated using randomized breaking dam simulations as shown in Figure 17. Figure 9 compares the results of MLFLIP trained using these two data sets. As shown in the comparison, both MLFLIP models produce very comparable splashes while demonstrating that our model successfully learns from both solvers. In addition, we believe that other NS solvers could likewise be employed to generate training data.

4.1. Physical Evaluation

Next, we will evaluate the ability of our approach to capture realistic physical behavior. Due to the turbulent nature of splashing liquids, we rely on a two-dimensional setup similar to the well known Plateau-Rayleigh instability for 3D flows [dGBWQ04]. Plateau-Rayleigh instabilities explain the break-up of tubes of liquid due to surface tension, and it is one of the well known phenomena leading to droplet formation.

We consider strings of liquid with different initial lengths sampling each with perturbed FLIP particles. These setups were simulated with surface tension and viscosity of water with zero gravity.

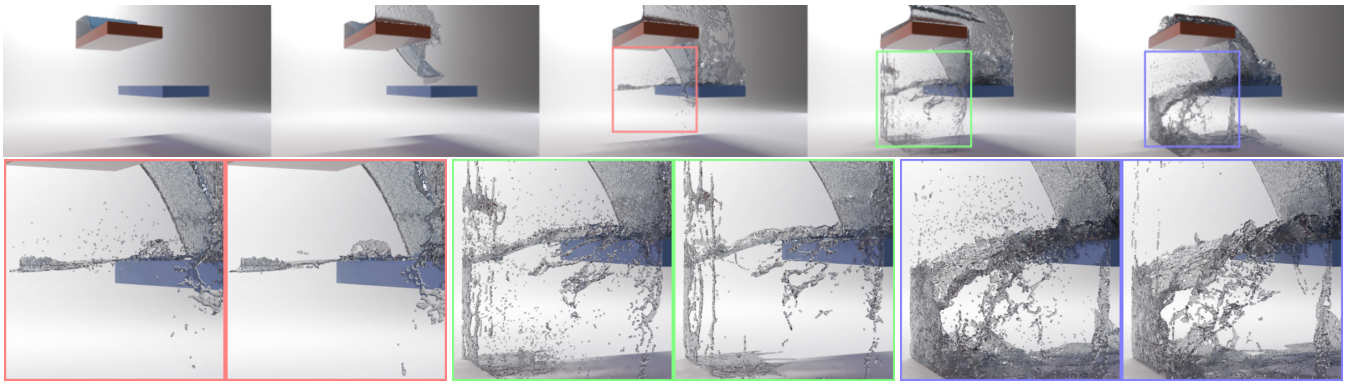

Figure 8: *Example frames of stairs simulations with MLFLIP. The bottom row compares (left) MLFLIP with (right) FLIP side-by-side in the selected areas of three frames.*

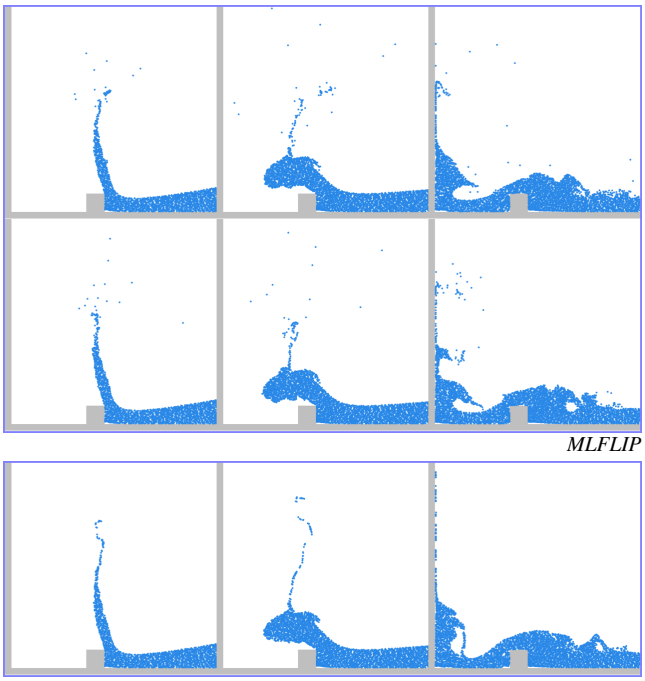

Figure 9: *Comparison among (top) MLFLIP trained using FLIP, (middle) MLFLIP trained using SPH, and (bottom) FLIP.*

As evaluation metric, we calculate the number of formed droplets after 1.5 seconds. To generate training data, we simulated 40 different simulations with a resolution of 1000×100 cells, which parametrized for a spatial scale of 2.5mm per cell. As shown in blue in Figure 6-(a), these simulations accurately resolve the surface tension effects and lead to the droplet counts shown with blue dots as reference data in Figure 6-(b).

We train our model on this data using a four times smaller base resolution of 250×25, for which we completely omit both surface tension and viscosity. With this resolution, the string is only ca. 1 cell thick, and hence the surface tension driven instabilities could not be properly simulated at this scale. Without surface tension, these simulations would not lead to any droplets forming. However, when enabling our model, we can accurately simulate the droplet formation for the liquid string setups. For these tests, we disable the inference of velocity variance in order to make the results independent of randomness. The results of our model are shown in orange in Figure 6-(a), and the corresponding droplet counts can be found in Figure 6-(b). As this graph shows, our model captures the ground truth numbers of droplets very well across the full range of different lengths. It is also worth pointing out here that simpler models for generating secondary particles would not be able to re-create this behavior. As such, this test case successfully demonstrates that our model learns to represent the underlying physics of droplet formation faithfully.

4.2. Secondary Particle Approach

While our approach described so far couples particles and bulk motion, we can modify our algorithm to produce secondary particles. To this end, when our model classifies simulation particles as splashes, we seed secondary particles at the same positions as the simulation particles and copy their velocities to the newly generated secondary particles. After that, the velocity modification is applied using random numbers. The secondary particle is simulated individually and removed when it ends up inside of the liquid volume. Figure 11 shows the example frames of the secondary particle model. Here, the splashes were enriched on the pre-simulated FLIP frames. Unlike the work of Ihmsen *et al.* [IAAT12], our model does not require any manual adjustment of parameters. While secondary splash models from previous work could potentially produce results similar to ours with enough manual adjustment of the parameters, the formation of splashes in our model purely relies on the generated training data. Despite this automated process, we demonstrate, in the following, that the results can be easily controlled by a user.

By virtue of the NN-based classification model, we can readily adopt the output classification values in order to control the number (i.e., likelihood) of droplet generation. Rather than classifying splash particles directly via the output probabilities of $\mathbf{y}_s$, we define a threshold ($\in [0,1]$) and compare the output probability of forming a splash with this threshold. Intuitively, a lower threshold

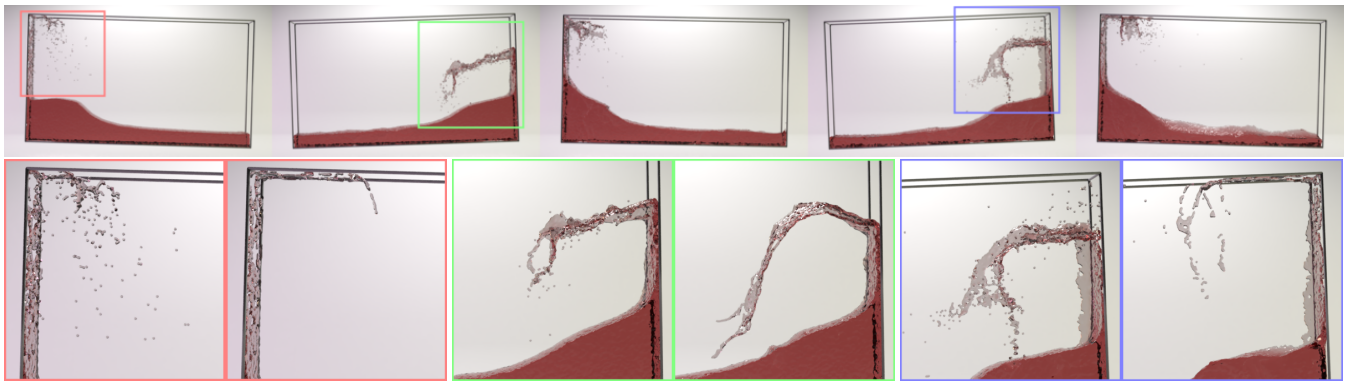

Figure 10: *Example frames of wave tank simulations with MLFLIP. The bottom row compares (left) MLFLIP with (right) FLIP side-by-side in the selected areas of three frames.*

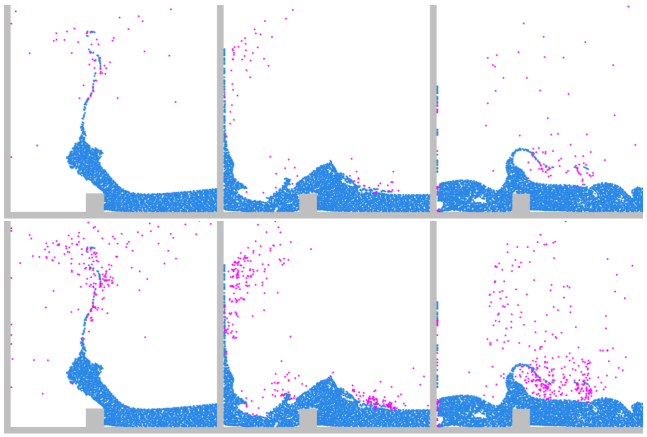

Figure 11: *Example frames of MLFLIP with the secondary particle approach. We colored secondary particles magenta. Each row uses two different numbers of samples (i.e., (top) one and (bottom) four) when we seed secondary particles per simulation particle.*

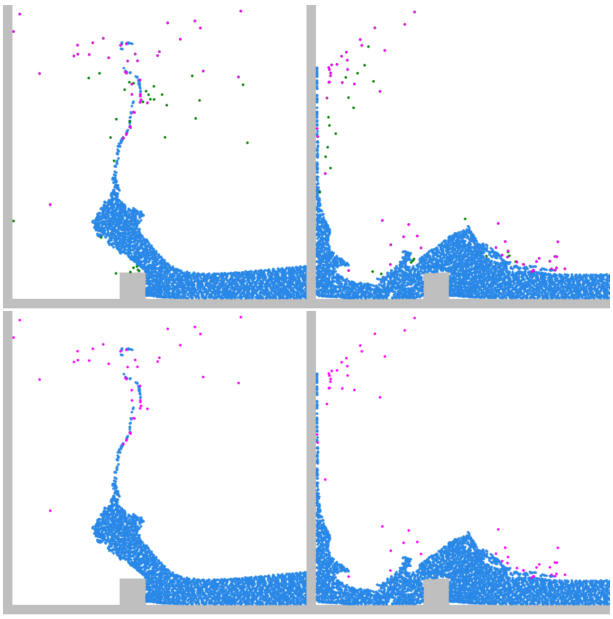

Figure 12: *Different numbers of splashes with two control thresholds: (top) 0.2 and (bottom) 0.8. The difference is highlighted in green at the top; i.e., the green color represents the additional splashes that are generated when the lower threshold of 0.2 is used.*

classifies the particle with a less likely value as a splash such that more splashes are generated, and vice versa. Figure 12 shows the example frames of two thresholds of 0.2 and 0.8. Note that here we adopted the secondary particle approach to assure the underlying simulations are comparable. The left side of Figure 15 plots the graph of the number of splash decisions with respect to the threshold value. We found that the number of splashes changes smoothly with respect to the chosen threshold $\mathbf{y}_s$. Note that positive splash decisions do not necessarily lead to the generation of visible droplets as some of these decisions can be reverted as outlined in Section 3.2.

One goal of our approach is to make the model robust to a wide range of spatial scales. Because of its data-driven nature, the reliable coverage of the model is often limited to the scale that the training data represent. Rather than training the model for a certain scale, we extend the model's coverage to a wider range using *heterogeneous* training data. To this end, we first generated the data in three scales: 2.5mm, 5mm, and 1cm in grid spacing (Figure 18).

Then, we used the grid spacing value of the target resolution as an input feature such that the model can correctly infer the right output values. Figure 13 shows the results of the extended model for similar surface positions and compares it with the three models that were individually trained using each set of data for each scale. We observed that the extended model produces splashes that are comparable to the three individual models. We also experimented with two intermediate scales, which are not directly represented by the training data, and found that the model works robustly as shown in Figure 14 and is able to generalize to a wider range of spatial scales. The simulations of Figure 13 and 14 all used the secondary particle approach.

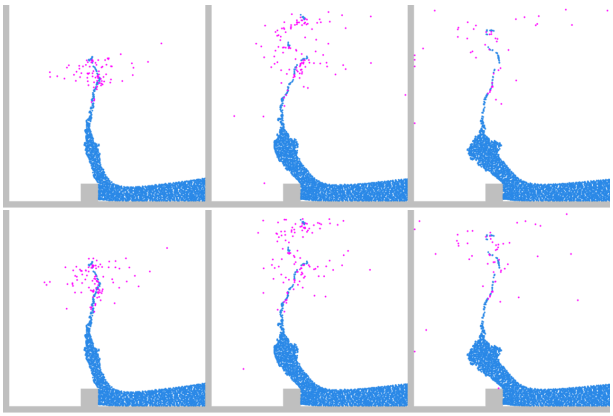

Figure 13: *Example frames of MLFLIP in three spatial scales: (left)* 5mm*, (middle)* 1cm*, and (right)* 2cm *in grid spacing. The top row shows the model that is trained using the three scales, whereas the bottom row shows the three models that are individually trained for each scale.*

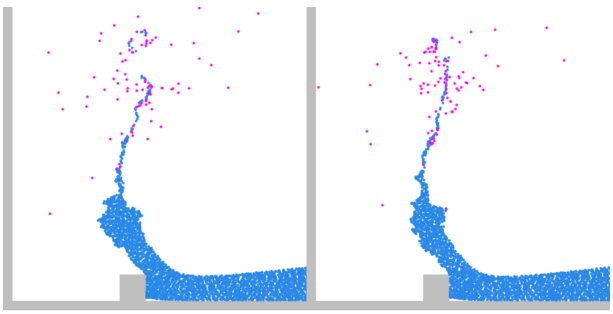

Figure 14: *Example frames of the three-scale MLFLIP model from Figure 13 for two intermediate scales: (left) 7.5*mm *and (right) 1.5*cm *that are not part of the training data.*

Convergence: In order to investigate if the learning process converges, we additionally trained our model with different numbers of samples and monitored the number of splashes generated using the model. Here, we also adopted the secondary particle approach in order to keep the same conditions for inference. The right side of Figure 15 shows the graph of the number of splash decisions with respect to the number of training samples. This graph indicates that the results of our model stabilize for more than ca. 40K training samples.

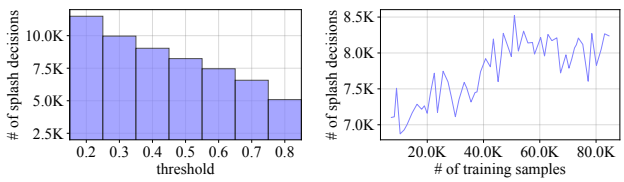

Figure 15: *The number of splash decisions with respect to (left) the threshold and (right) the number of training samples.*

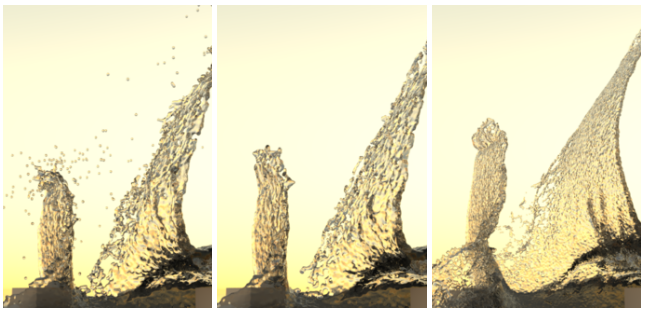

Figure 16: *Comparisons among the three simulations: (left) MLFLIP, (middle) FLIP, and (right) FLIP with the high resolution.*

5. Three-dimensional Results

This section demonstrates that our model improves the visual fidelity of liquid simulations in three examples: a *dam*, *stairs*, and a *wave tank*. The former two examples represent larger scales than the wave tank example in length. Hence, we use the 5mm splash model for both (i.e., dam and stairs) and 1.5mm splash model for the wave tank. Figure 5 and 8 show the comparisons between FLIP and MLFLIP for the dam and stairs examples. Our model leads to a significant increase in violent and detailed splashes for this large scale flow. Despite the large number of splashes, the plausibility of the overall flow is preserved. Likewise, our model robustly introduces splashes also in the smaller wave tank example as shown in Figure 10. The smaller real world size in conjunction with the smaller velocities leads to fewer splashes for this setup.

Our model requires additional calculations for the generation of splashes, and consequently, this results in a slightly increased runtime. In the dam example, the average computation time per simulation step was 0.52s for FLIP, while it was 0.60s for MLFLIP. Both simulations used the same grid resolution of $160\times150\times50$, where the grid spacing was 2cm. In the stairs example, the runtime was 0.62s for FLIP and 0.78s for MLFLIP, and their grid resolutions were both $100\times200\times100$. In the wave tank example, the runtime was 0.12s for FLIP and 0.14s for MLFLIP. In this case, the grid resolution was $150\times84\times10$, where the grid spacing was 0.6cm.

We observed that the splashes of our MLFLIP simulation are very difficult to resolve with regular FLIP simulations even with high resolutions. For the dam example, Figure 16 shows a visual comparison of three simulations: FLIP and MLFLIP with the same resolution and FLIP with a doubled resolution of $320\times300\times100$. Despite taking nine times longer per simulation step (i.e., 5.43s for FLIP and 0.60s for MLFLIP), this high resolution simulation fails to resolve the droplets of our MLFLIP simulation. Our model successfully generates splash details from a low resolution simulation, while the high resolution simulation barely improves the number of splashes despite its high computational cost.

6. Discussion and Limitations

Our work shares its goal with the well-known secondary particle approaches. Many of these approaches have proposed physically inspired heuristics utilizing a combination of measures, such as cur-

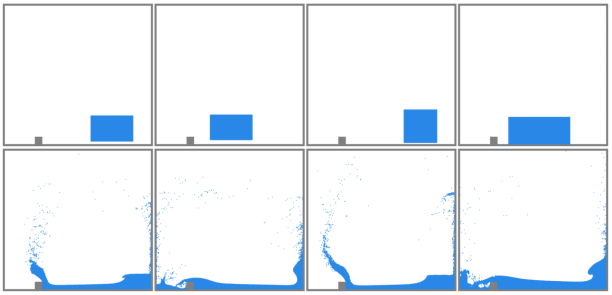

Figure 17: *Example frames of SPH simulations for training data generation. The top row shows four selected initial conditions; the bottom row shows their snapshots after one second.*

vature, velocity, vorticity, or deformation gradient to generate secondary particles. This naturally requires a manual tuning process to find a set of proper parameter values for the different components. In contrast, our model aims for an automated data-driven algorithm that does not require any tuning but instead learns a probabilistic model from reference data. However, with enough manual tuning for each simulation setup, simpler algorithms for secondary particles could lead to results that are very similar to the output of our algorithm.

Our model directly works with the regular representation of the fluid, i.e., the simulation particles, and does not require additional helper data structures. In this way, our model can be easily two-way-coupled with a FLIP simulation allowing for interactions of the splash effects with the bulk liquid. As a consequence, our model not only yields improved small scale details but can also capture realistic droplet formation effects as illustrated in Figure 6. This effect would be very difficult to re-create with previous algorithms. Interestingly, our model learns this from a set of examples instead of requiring a hand crafted analytical expression. Thus, our algorithm requires additional work for data generation and training but, in this way, arrives at a more general framework for the data-driven modeling of droplet effects.

In our data generation step, we typically place several droplets to incite splashes and randomize the initial velocities of these droplets in a pre-defined range. With our current model, this limits the inference capabilities, for instance, when simulations exhibit much larger velocities than the ones present in the training data. However, it would be possible to detect such cases in order to trigger generating additional training data. Apart from the data generation via numerical simulations, we envision that extracting data from real world experiments [GKN07] will be an interesting alternative for future work.

Because our model uses the simulation particles to represent droplets, the details of generated splashes depend on the particle resolution of the underlying FLIP simulation. This limits the scale of droplets to the size of a single simulation particle. Our model could be extended to a scale variant droplet model by adopting an adaptive approach [ATW13] and learning the formation of different scale droplets from physical parameters. The Ohnesorge number [Lef11] to characterize the dispersion behavior of liquids would be a good candidate here.

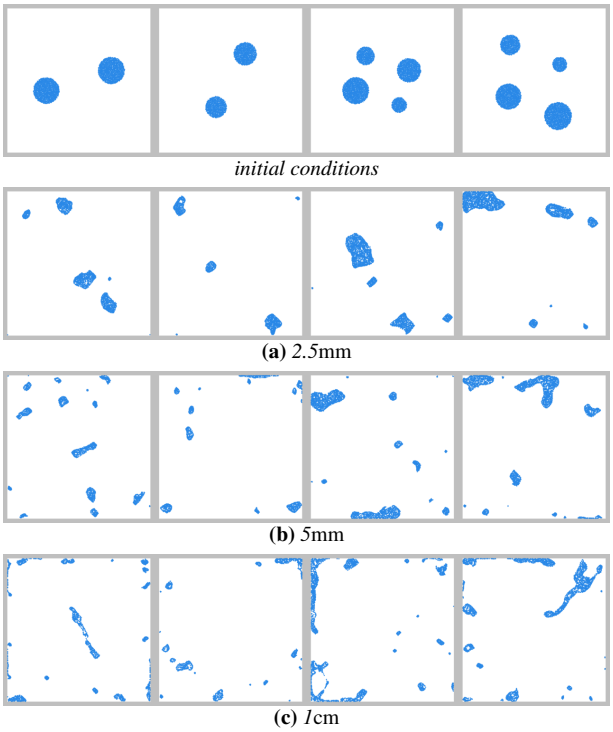

Figure 18: *Example frames of FLIP simulations with surface tension for training data generation for three spatial scales: (a) 2.5*mm*, (b) 5*mm*, and (c) 1*cm *in grid spacing. The initial conditions for each simulation are randomly configured similar to the three-dimensional examples (Figure 3).*

Although the splashes incited by our model transfer momentum when they merge with the liquid volume, our model currently does not compute interactions among droplets [JS17]. Finally, we only experimented with the formation of splash droplets. However, we envision that our model could be extended to other complex subgrid effects such as bubbles, capillary waves, and foam, which are highly expensive to compute with regular free-surface liquid simulations. We believe that droplet interactions and additional small-scale phenomena are important for believable simulations; thus, it will be very interesting to explore how our NN-based model can extend to these directions in future work.

7. Conclusions

This paper introduced a new data-driven splash generation model that utilizes machine learning techniques with NNs. We demonstrated that our models successfully learn the formation of splashes at different physical scales from the training data with the training process. Our model leads to improved splashing liquids and successfully learns to recognize the relevant mechanisms for droplet formation from the pre-computed data. Moreover, we demonstrated an extension of our model for secondary particles, and we evaluated the learned models with a variety of complex tests. Overall, our experiments highlight that our approach provides a very good basis for learning a wide range of realistic physical effects.

Acknowledgments

This work is supported by the ERC Starting Grant 637014.

References

[AAT13] AKINCI N., AKINCI G., TESCHNER M.: Versatile surface tension and adhesion for SPH fluids. *ACM Trans. Graph. 32*, 6 (Nov. 2013), 182:1–182:8. doi:10.1145/2508363.2508395. 2

[ATT12] ANDO R., THUREY N., TSURUNO R.: Preserving fluid sheets with adaptively sampled anisotropic particles. *IEEE Transactions on Visualization and Computer Graphics 18*, 8 (2012), 1202–1214. doi:10.1109/TVCG.2012.87. 2

[ATW13] ANDO R., THÜREY N., WOJTAN C.: Highly adaptive liquid simulations on tetrahedral meshes. *ACM Trans. Graph. 32*, 4 (July 2013), 103:1–103:10. doi:10.1145/2461912.2461982. 10

[BB08] BATTY C., BRIDSON R.: Accurate viscous free surfaces for buckling, coiling, and rotating liquids. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2008), SCA '08, Eurographics Association, pp. 219–228. 2

[BB12] BOYD L., BRIDSON R.: MultiFLIP for energetic two-phase fluid simulation. *ACM Trans. Graph. 31*, 2 (Apr. 2012), 16:1–16:12. doi:10.1145/2159516.2159522. 2

[BBB07] BATTY C., BERTAILS F., BRIDSON R.: A fast variational framework for accurate solid-fluid coupling. *ACM Trans. Graph. 26*, 3 (July 2007), 100:1–100:7. doi:10.1145/1276377.1276502. 2

[Bis06] BISHOP C. M.: *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. 3, 4

[BK17] BENDER J., KOSCHIER D.: Divergence-free SPH for incompressible and viscous fluids. *IEEE Transactions on Visualization and Computer Graphics 23*, 3 (Mar. 2017), 1193–1206. doi:10.1109/TVCG.2016.2578335. 2

[BPT17] BONEV B., PRANTL L., THUEREY N.: Pre-computed liquid spaces with generative neural networks and optical flow. *arXiv:1704.07854* (2017). arXiv:1704.07854. 3

[Bri15] BRIDSON R.: *Fluid Simulation for Computer Graphics*. CRC Press, 2015. 2

[CT17] CHU M., THUEREY N.: Data-driven synthesis of smoke flows with CNN-based feature descriptors. *ACM Trans. Graph. 36*, 4 (July 2017), 69:1–69:14. doi:10.1145/3072959.3073643. 2

[dGBWQ04] DE GENNES P.-G., BROCHARD-WYART F., QUERE D.: *Capillarity and Wetting Phenomena*, first ed. Springer-Verlag New York, 2004. 6

[EMF02] ENRIGHT D., MARSCHNER S., FEDKIW R.: Animation and rendering of complex water surfaces. *ACM Trans. Graph. 21*, 3 (July 2002), 736–744. doi:10.1145/566654.566645. 2

[FAW*16] FERSTL F., ANDO R., WOJTAN C., WESTERMANN R., THUEREY N.: Narrow band FLIP for liquid simulations. *Computer Graphics Forum 35*, 2 (2016), 225–232. 2

[FGG*17] FU C., GUO Q., GAST T., JIANG C., TERAN J.: A polynomial particle-in-cell method. *ACM Trans. Graph. 36*, 6 (Nov. 2017), 222:1–222:12. doi:10.1145/3130800.3130878. 2

[GB13] GERSZEWSKI D., BARGTEIL A. W.: Physics-based animation of large-scale splashing liquids. *ACM Trans. Graph. 32*, 6 (Nov. 2013), 185:1–185:6. doi:10.1145/2508363.2508430. 1, 2

[GDDM14] GIRSHICK R., DONAHUE J., DARRELL T., MALIK J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proc. Conference on Computer Vision and Pattern Recognition* (2014), IEEE, pp. 580–587. 2

[GKN07] GARG K., KRISHNAN G., NAYAR S. K.: Material based splashing of water drops. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques* (Aire-la-Ville, Switzerland, Switzerland, 2007), EGSR'07, Eurographics Association, pp. 171–182. doi:10.2312/EGWR/EGSR07/171-182. 10

[GSLF05] GUENDELMAN E., SELLE A., LOSASSO F., FEDKIW R.: Coupling water and smoke to thin deformable and rigid shells. In *ACM SIGGRAPH 2005 Papers* (New York, NY, USA, 2005), SIGGRAPH '05, ACM, pp. 973–981. doi:10.1145/1186822.1073299. 1

[HEVEDB09] HIJÓN C., ESPAÑOL P., VANDEN-EIJNDEN E., DELGADO-BUSCALIONI R.: Mori-zwanzig formalism as a practical computational tool. *Faraday Discussions 144* (Oct. 2009), 301–322. doi:10.1039/B902479B. 3

[HK05] HONG J.-M., KIM C.-H.: Discontinuous fluids. *ACM Trans. Graph. 24*, 3 (July 2005), 915–920. doi:10.1145/1073204.1073283. 5

[IAAT12] IHMSEN M., AKINCI N., AKINCI G., TESCHNER M.: Unified spray, foam and air bubbles for particle-based fluids. *The Visual Computer 28*, 6-8 (2012), 669–677. 1, 2, 7

[IOS*14] IHMSEN M., ORTHMANN J., SOLENTHALER B., KOLB A., TESCHNER M.: SPH fluids in computer graphics. In *Eurographics 2014 - State of the Art Reports* (Strasbourg, France, 2014), Eurographics Association, pp. 21–42. doi:10.2312/egst.20141034. 2

[IS15] IOFFE S., SZEGEDY C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv:1502.03167* (Feb. 2015). arXiv:1502.03167. 6

[JS17] JONES R., SOUTHERN R.: Physically-based droplet interaction. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2017), SCA '17, ACM, pp. 5:1–5:10. doi:10.1145/3099564.3099573. 10

[JSS*15] JIANG C., SCHROEDER C., SELLE A., TERAN J., STOMAKHIN A.: The affine particle-in-cell method. *ACM Trans. Graph. 34*, 4 (July 2015), 51:1–51:10. doi:10.1145/2766996. 2

[KB14] KINGMA D., BA J.: Adam: A method for stochastic optimization. *arXiv:1412.6980 [cs]* (Dec. 2014). arXiv:1412.6980. 6

[KCC*06] KIM J., CHA D., CHANG B., KOO B., IHM I.: Practical animation of turbulent splashing water. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2006), SCA '06, Eurographics Association, pp. 335–344. 1

[KNCA11] KHOSRAVI A., NAHAVANDI S., CREIGHTON D., ATIYA A. F.: Comprehensive review of neural network-based prediction intervals and new advances. *IEEE Transactions on Neural Networks 22*, 9 (Sept. 2011), 1341–1356. doi:10.1109/TNN.2011.2162110. 3

[KSH12] KRIZHEVSKY A., SUTSKEVER I., HINTON G. E.: Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems* (2012), NIPS, pp. 1097–1105. 2

[LB12] LOSURE J. B. M., BAER K. M. M.: Liquids in the croods. *SIGGRAPH Talks* (2012). 4

[Lef11] LEFEBVRE A.: *Atomization and Sprays*. CRC Press, 2011. 10

[LJS*15] LADICKÝ L., JEONG S., SOLENTHALER B., POLLEFEYS M., GROSS M.: Data-driven fluid simulations using regression forests. *ACM Trans. Graph. 34*, 6 (Oct. 2015), 199:1–199:9. doi:10.1145/2816795.2818129. 2

[LTKF08] LOSASSO F., TALTON J., KWATRA N., FEDKIW R.: Two-way coupled SPH and particle level set fluid simulation. *IEEE Transactions on Visualization and Computer Graphics 14*, 4 (2008), 797–804. doi:10.1109/TVCG.2008.37. 2

[MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2003), SCA '03, Eurographics Association, pp. 154–159. 2

[MKS*13] MNIH V., KAVUKCUOGLU K., SILVER D., GRAVES A., ANTONOGLOU I., WIERSTRA D., RIEDMILLER M.: Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013). 2

[NW94] NIX D. A., WEIGEND A. S.: Estimating the mean and variance of the target probability distribution. In *IEEE International Conference on Neural Networks, 1994. IEEE World Congress on Computational Intelligence* (June 1994), vol. 1, pp. 55–60. doi:10.1109/ICNN.1994.374138. 3, 6

[PP02] PAPOULIS A., PILLAI S. U.: *Probability, Random Variables, and Stochastic Processes*, fourth ed. McGraw Hill, 2002. 4

[SP09] SOLENTHALER B., PAJAROLA R.: Predictive-corrective incompressible SPH. *ACM Trans. Graph. 28*, 3 (July 2009), 40:1–40:6. doi:10.1145/1531326.1531346. 2

[ten16] TensorFlow, 2016. URL: http://tensorflow.org/. 6

[TFK*03] TAKAHASHI T., FUJII H., KUNIMATSU A., HIWADA K., SAITO T., TANAKA K., UEKI H.: Realistic animation of fluid with splash and foam. *Computer Graphics Forum 22*, 3 (2003), 391–400. doi:10.1111/1467-8659.00686. 1

[TSSP17] TOMPSON J., SCHLACHTER K., SPRECHMANN P., PERLIN K.: Accelerating eulerian fluid simulation with convolutional networks. In *Proceedings of Machine Learning Research* (July 2017), pp. 3424–3433. 2

[UBH14] UM K., BAEK S., HAN J.: Advanced hybrid particle-grid method with sub-grid particle correction. *Computer Graphics Forum 33*, 7 (Oct. 2014), 209–218. doi:10.1111/cgf.12489. 2

[UHT17] UM K., HU X., THUEREY N.: Perceptual evaluation of liquid simulation methods. *ACM Trans. Graph. 36*, 4 (July 2017), 143:1–143:12. doi:10.1145/3072959.3073633. 1

[XFCT18] XIE Y., FRANZ E., CHU M., THUEREY N.: tempoGAN: A temporally coherent, volumetric gan for super-resolution fluid flow. *arXiv:1801.09710 [cs]* (Jan. 2018). arXiv:1801.09710. 2

[YLHQ14] YANG L., LI S., HAO A., QIN H.: Hybrid particle-grid modeling for multi-scale droplet/spray simulation. *Computer Graphics Forum 33*, 7 (Oct. 2014), 199–208. doi:10.1111/cgf.12488. 2

[YLX*15] YANG L., LI S., XIA Q., QIN H., HAO A.: A novel integrated analysis-and-simulation approach for detail enhancement in FLIP fluid interaction. In *Proceedings of the 21st ACM Symposium on Virtual Reality Software and Technology* (New York, NY, USA, 2015), VRST '15, ACM, pp. 103–112. doi:10.1145/2821592.2821598. 2

[YYX16] YANG C., YANG X., XIAO X.: Data-driven projection method in fluid simulation. *Computer Animation and Virtual Worlds* (Jan. 2016), 415–424. doi:10.1002/cav.1695. 2

[ZB05] ZHU Y., BRIDSON R.: Animating sand as a fluid. *ACM Trans. Graph. 24*, 3 (July 2005), 965–972. doi:10.1145/1073204.1073298. 2