DOI: 10.1111/cgf.14010

COMPUTER GRAPHICS *forum*
Volume 00 (2020), number 00 pp. 1–12

Accelerating Liquid Simulation With an Improved Data-Driven Method

Yang Gao,[1,2] Quancheng Zhang,[1] Shuai Li,[1,3] Aimin Hao[1,2,3] and Hong Qin[4]

[1]State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing, China
gaoyang2963@163.com, zqcpengfei@gmail.com, lishuaiouc@126.com, ham_buaa@163.com
[2]Research Institute for Frontier Science, Beihang University, Beijing, China
[3]Peng Cheng Laboratory, Shenzhen, China
[4]Department of Computer Science, Stony Brook University (SUNY), Stony Brook, NY, USA
qin@cs.stonybrook.edu

Abstract

In physics-based liquid simulation for graphics applications, pressure projection consumes a significant amount of computational time and is frequently the bottleneck of the computational efficiency. How to rapidly apply the pressure projection and at the same time how to accurately capture the liquid geometry are always among the most popular topics in the current research trend in liquid simulations. In this paper, we incorporate an artificial neural network into the simulation pipeline for handling the tricky projection step for liquid animation. Compared with the previous neural-network-based works for gas flows, this paper advocates new advances in the composition of representative features as well as the loss functions in order to facilitate fluid simulation with free-surface boundary. Specifically, we choose both the velocity and the level-set function as the additional representation of the fluid states, which allows not only the motion but also the boundary position to be considered in the neural network solver. Meanwhile, we use the divergence error in the loss function to further emulate the lifelike behaviours of liquid. With these arrangements, our method could greatly accelerate the pressure projection step in liquid simulation, while maintaining fairly convincing visual results. Additionally, our neutral network performs well when being applied to new scene synthesis even with varied boundaries or scales.

Keywords: physically based animation, animation

ACM CCS: • Computing methodologies → Physical simulation; Neural networks

1. Introduction and Motivation

Fluid behaviours are widespread in nature and fluid simulation has always been a research hotspot in the field of industrial fluid modelling, computer graphics, film/game industry, etc. The dynamic behaviour of fluid is governed by the Navier–Stokes equations (N-S equations) which usually need iterative numerical solvers that is time-consuming in computation. To improve the efficiency of the simulators has always been a popular topic in researches. For instance, some researchers proposed adaptive-scale particles [APKG07, ZHQH], hybrid grids [MCPN08, CM11] and dynamic particle partitioning [ZGL*18] to reduce the computational burden. Using the preconditioned conjugate gradient (PCG) method [Bri15] instead of the general numerical iterative method, such as Jacobi iteration or Gauss–Seidel iteration, could obtain a significantly faster convergence rate. Although these methods alleviate the problem to some extent, it is still impossible to avoid the solving of the N-S equations, which is the bottleneck of efficiency.

In recent years, apart from the traditional physics-based solvers, data-driven approaches attract more and more attention of researchers. To name a few, Treuille *et al.* [TLP06] proposed a dimensionality reduction method that can conduct real-time simulation of flows. Kim *et al.* [KD13] improved the subspace scheme to support semi-Lagrangian advection. Zhai *et al.* [ZHQH17] later combined the subspace technique with Empirical Mode Decomposition (EMD) to control the style of flow animation. Jeong *et al.* [JSP*15] integrated random forests into the smoothed-particle hydrodynamics (SPH) algorithm to achieve

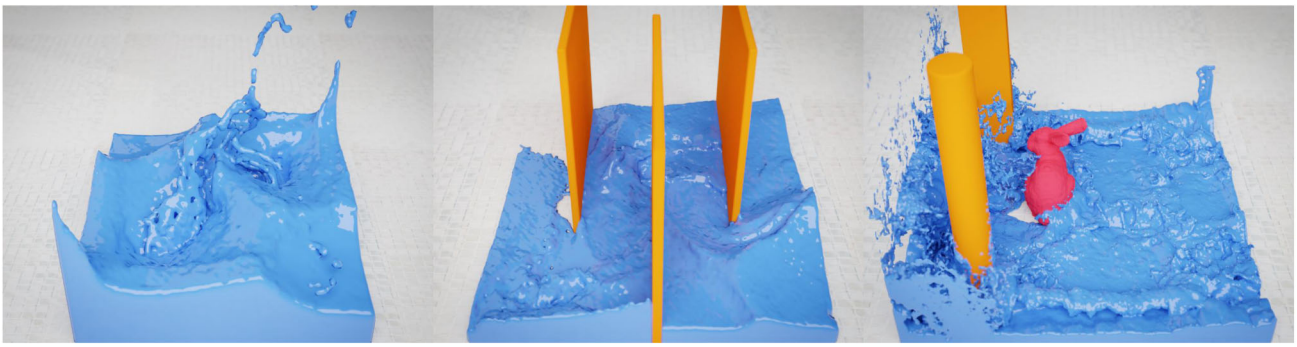

Figure 1: *Generated results using our method with varying resolutions. Left (80^3): Two water balls fall free; Middle (128^3): Water shuttles among three boards; Right (240^3): A breaking dam impacts a bunny and two cylinders. Our method speeds up about 10–100 times over PCG when the resolution increases and generates realistic visual effects.*

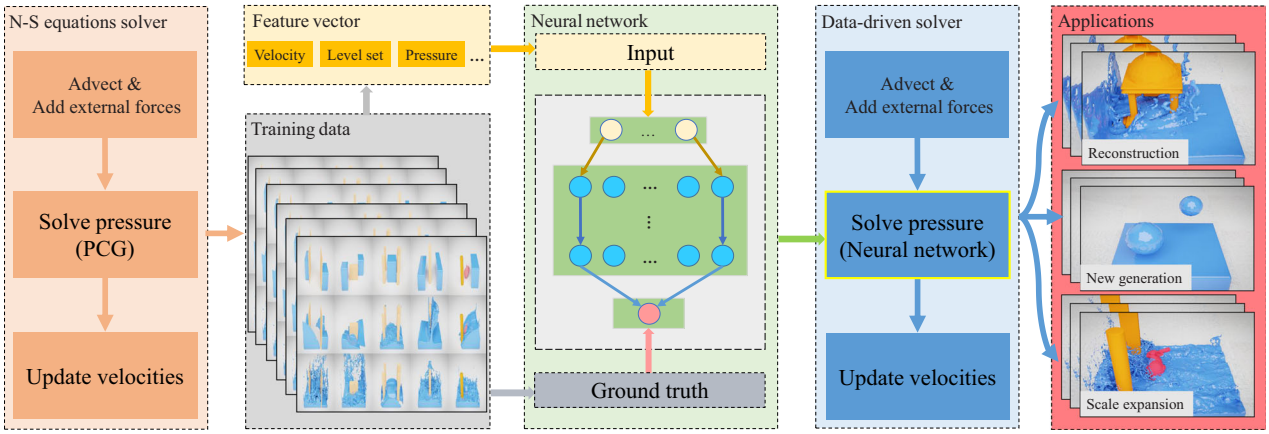

Figure 2: *A brief illustration of our work. We generate training data with PCG solver and train our networks with the special designed feature vector for liquid. We integrate the trained network into the simulation framework for more efficiently solving pressure projection than PCG. And our method can support three main applications involving reconstructing liquid simulations, generating new simulations and expanding liquid scenes in a very fast way.*

the end-to-end fluid reproduction. In recent years, with the rapid development of deep learning, many researchers seeked to treat physics-based simulation as learning applications. For example, there was a latent-space physics [WBT18] that also adapted the end-to-end idea. Yang *et al.* [YYX16] and Tompson *et al.* [TSSP17] used a neural network (NN) as pressure solver for smoke modelling instead of the conventional PCG solver. However, while these methods have significantly improved efficiency, few of them applied for liquids since there are significant differences between liquid and smoke, such as surface, density, viscosity, etc.

To address the computational efficiency problem in liquid simulation, this paper presents a data-driven approach to quickly solve the pressures and to authentically reconstruct the motions. More specifically, instead of solving the large sparse linear system, we establish our fast liquid pressure inference based on the existing NN gas simulators [YYX16, TSSP17]. To accommodate liquid circumstance, we take advantage of level set and velocity by putting them into the input feature vector so that the network perceives the surroundings more accurately. Besides, a divergence term is included into the loss function to predict more realistic results. We use the fluid-implicit-particle (FLIP) model as the basic framework in this paper for straightforward implementation, but our network solution can be easily migrated to other models as well (refer to Figure 1). The training dataset is produced from a high-precision PCG solver, which ensures the effective inference of pressures. The brief pipeline of our work is shown in Figure 2.

Specifically, the salient contributions of this paper include:

- We present a data-driven method to solve pressure projection for liquid simulation, which greatly improves the efficiency and our method can be easily migrated to other liquid models.
- We propose a feature vector that includes representative characteristics such as level set and velocity which can accurately describe the liquid state during simulation.
- We develop an FLIP pressure solver that uses only local information for liquid generation, which is capable of handling varying boundaries and spatial scales.

2. Related Works

This work is closely related to the Lagrangian methods, the grid-based fluid solvers and machine-learning-based fluid simulation. We briefly review them in the following.

Lagrangian methods discretize fluid into particles to tackle the implied dynamics. A typical representative is the smooth particle hydrodynamics (SPH) method which was first introduced by Monaghan *et al.* [Mon92] for fluid simulation. To improve the computational efficiency of SPH, Desbrun *et al.* [DC99] used a space–time adaptive method with smaller time step and particle radius in key areas, and reduced computing overhead by concentrating computing resources. Adams *et al.* [APKG07] proposed an approach using small radius particles in areas with complex boundaries and fast flow rates. Ihmsen *et al.* [ICS*14] developed IISPH of a formula for projection based on PCISPH [SP09] that allows for larger time step and a faster convergence rate. Takahashi *et al.* [TDF*15] calculated viscosity implicitly to improve the stability and efficiency of viscous integration.

Grid-based methods are also very popular apart from the Lagrangian methods. Since Foster *et al.* [FM96] applied Euler method to the field of computer vision, it has attracted attentions of many researchers [EMF02, SKK07]. The detail level of Euler solvers depends on the resolution of grids so that it shall be at the expense of computational efficiency to make finer grids with rich details. For grid-based methods, the pressure projection needs to solve a large linear equation system whose scale is related to the number of grids, which takes up most of the simulation time. Enright *et al.* [ENGF03] proposed the ghost particle method to improve the accuracy and efficiency of pressure projection on the surface of liquid. Foster *et al.* [FF01] and Bridson *et al.* [Bri15] used the PCG method to solve linear equations, achieving relatively high efficiency. Ando *et al.* [ATW13] leveraged an alternative Eulerian tetrahedral mesh discretization to significantly reduce the complexity of the pressure solver while increasing the robustness. Molemaker *et al.* [MCPN08] and Chentanez *et al.* [CM11] used multi-grid scheme to solve the pressure projection to realistically represent the details of complex areas, while ensuring stability under large time step settings. Among the grid based solvers, FLIP is the most popular method which describes fluid with particles and solve equations in grids. It combines the advantages of Lagrangian and Euler to effectively simulate complex fluids. Gao *et al.* [GLQH17, GLQ*19] introduced unified solvers for different materials to improve the efficiency and to solve the interactions within FLIP model. Ferstl *et al.* [FAW*16] proposed the narrow-band FLIP method, which sampled particles only near the surface of the liquid. It ensures rich visual details while reducing the computational overhead overall.

Machine learning has been applied into fluid simulation in recent years. To avoid solving the time-consuming equations during the simulation, Jeong *et al.* [JSP*15] proposed a random forests method integrated with SPH. In the particle system, the current particle states were encoded and transferred into a trained random forest, and the particles states in the next frame were directly obtained. Wiewel *et al.* [WBT18] trained a set of automatic encoder-decoder, encoding data of each frame as the input, with a long-short-term memory network as the predictor of the next frame and used decoder after obtaining the predicted frame sequence. These end-to-end approaches abandoned the traditional idea of solving N-S equations. In addition to using machine learning directly for fluid simulation, Zheng *et al.* [ZGL*18] proposed an unsupervised PP-SPH method for high-speed fluids acceleration. They introduced a k-means clustering method into the SPH to partition particles into two disjoint groups, and used a two-scale time step scheme for these two types of particles. Some researchers also used machine learning to improve the visual effects of simulation results. Chu *et al.* [CT17] and Xie *et al.* [XFCT18] used convolutional NN to generate superresolution results from low-resolution simulation scenarios. Um *et al.* [UHT18] proposed a deep NN to add a large number of splashed droplet details to a low-resolution simulation scene. Moreover, Yang *et al.* [YYX16] and Tompson *et al.* [TSSP17] accelerated the solving of Poisson equation by training an NN as the pressure solver. However, their works only reproduced the dynamics of smoke but could not be directly used for liquid. There is always a clear surface between liquid and air domains which has very different motions on its two sides. To deal with this partition of space is the main challenge of our work.

3. FLIP Model and Data-Driven Method

3.1. Basic FLIP model

In this paper, we use FLIP, which expresses material movement by particles and solves pressure projection with grids, as the basic simulating model. The training set is acquired with a high-precision FLIP simulator as well.

Physics-based fluid simulation is governed by the N-S equations and needs to solve the Poisson equation in the projection step to ensure incompressibility. The calculation model for a non-viscous, incompressible fluid can be seen in the following equations:

$$\frac{\partial \boldsymbol{u}}{\partial t} = -\boldsymbol{u} \cdot \nabla \boldsymbol{u} - \nabla p/\rho + \boldsymbol{f}, \tag{1}$$

$$\nabla \cdot \boldsymbol{u} = 0, \tag{2}$$

where $\boldsymbol{u}$ is the velocity, p is the pressure, ρ is the density of liquid and $\boldsymbol{f}$ is external force. Equation (1) is the momentum equation which controls the evolution of the fluid in space and time. Equation (2) requires that the divergence of the velocity field be zero everywhere and ensures that the fluid is not compressible. To solve these equations, it is necessary to discretize them in space and solve them by operator decomposition.

When performing pressure projection, we need to solve the Poisson equation:

$$\nabla \cdot \boldsymbol{u}^* = \frac{\Delta t}{\rho} \nabla^2 p, \tag{3}$$

where $\boldsymbol{u}^*$ is the temporary velocity before projection and Δt is time step. After discretizing the equations on the MAC grid, a large linear equations will be obtained by rearranging the resulting equations. The traditional solution of linear equations through numerical iterations, such as Gauss–Seidel iteration, PCG, etc., is often the most computationally expensive process in each time step. The whole algorithm framework is shown in Algorithm 1.

Algorithm 1. Algorithm Framework

Input:
Velocity of n_{th} frame $\mathbf{u}_n$
External force $\boldsymbol{f}$
Grid occupation O_n
Output:
Velocity of $(n+1)_{th}$ frame $\boldsymbol{u}_{n+1}$
for each particle **do**
Advect particle with $\boldsymbol{u}_n$
end for
Map particles to grid
Add external force $\boldsymbol{f}$
Pressure projection:
    Solve Poisson equation with PCG **or**
    [NN]Solve Poisson equation with NN
    Update grid velocity
Update particles velocity
return $\boldsymbol{u}_{n+1}$

3.2. NN for smoke simulation

For smoke simulation, in order to solve Equation (3) faster, some researchers have tried to train the NN as a pressure solver. Yang *et al.* [YYX16] adopted the local solver, which determines the pressure value based on velocity, pressure, etc., of a local region. They designed the content of the feature vector β by taking into account the discrete form of the Poisson equation with all the variables that may affect result, including pressure, divergence and obstacle occupation for the grid cell and its six adjacent grids:

$$\beta = \left\{ p_0, \bigcup_{i=1}^{6} \{p_i, \nabla \cdot \boldsymbol{u}_i, O_i\} \right\}, \tag{4}$$

where p_i $(i = 0, \ldots, 6)$ represents the pressure of the grid cell and its adjacent six grids, respectively, and it is the same for $\boldsymbol{u}_i$ $(i = 0, \ldots, 6)$ and O_i $(i = 0, \ldots, 6)$.

Instead of local solver, Tompson *et al.* [TSSP17] developed a global solver, which uses information from the global region to solve the pressure problem. They transformed the learning task into unsupervised learning by setting the training objective to reduce the global velocity divergence. Here is the loss function:

$$L_d = \left(\nabla \cdot \left(\boldsymbol{u}^* - \nabla p'\right)\right)^2, \tag{5}$$

where p' is the predicted pressure. These works provide us algorithmic and theoretical basis for the data-driven applications of fluid simulation.

4. Data-Driven Model for Liquid Simulation

Traditional fluid simulation methods typically use rigorous mathematical methods to solve N-S equations, which can obtain accurate results but are very time-consuming. Although some researchers use NNs to speed up the solution in smoke simulation, our purpose is to simulate liquid behaviours with effective NN. In this section, we will introduce how to apply NN to liquid simulation.

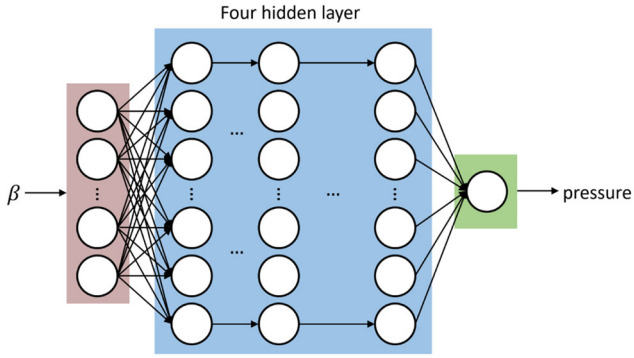

Figure 3: *The structure of the network.*

4.1. Learning liquid dynamics

The main goal of this paper is to make use of the function fitting ability of NN and regard it as a regression tool to solve the pressure projection. We adapt the local solver so that we can handle various boundaries flexibly. Also, it can be easily integrated into the methods of spatially discretizing N-S equations over grids. Our NN has a fully connected architecture with five layers. The number of neurons in the input layer is equal to the dimension of the feature vector which will be discussed later (Section 4.1.1). The number of neurons in the hidden layer is twice that of the input layer. The output layer has only one neuron, which represents the pressure. By evaluating the linear activation functions and multiple non-linear activation functions, we adopts the ReLU function as our activation function. Figure 3 illustrates the structure of the network. In this section, we will detail the design process of our feature vector and loss function.

4.1.1. *Feature vector construction*

Feature vector is used as the input of NN and represents all the information that the regression network can obtain. To make the network predict result as accurately as possible, the feature vector must be designed reasonably. According to the discrete form of Poisson's equation, Yang *et al.* [YYX16] designed the eigenvector describing the smoke state. This form may be feasible when describing smoke. However, both physical properties and motion patterns of the liquid are quite different from smoke, so we still need to find more characteristics to describe liquid.

Buoyancy serving as the external force, smoke flows often drift around and there is no obvious boundary between smoke and empty grids. Unlike smoke, there is a clear interface between the liquid and air, so for liquid, it is easy to track the interface via the level set method which describes the minimum distance Φ from each grid cell to the liquid surface. Moreover, the pressure inside the fluid is also closely related to the distance from the surface, especially in the vicinity of the liquid surface. So, the characteristic of level set should be considered as one of the liquid features.

In addition to level set, we also refer to the definition of fluid pressure in computational fluid dynamics to accurately extract the characteristics that may influence fluid pressure. We consider velocity as

one of the characteristics, and for non-viscous incompressible ideal fluid, the Bernoulli equation gives a close relationship between velocity and pressure. It should be pointed out that we are looking for an empirical approach to improving the accuracy of our method. Although Bernoulli's equation describes an irrotational and static flow, we do not need to pursue rigorous formula derivation. Bernoulli's equation indicates that the total pressure p at a point is divided into dynamic pressure p_d and static pressure p_s:

$$p = p_d + p_s = \frac{1}{2}\rho|\boldsymbol{u}|^2 + p_s, \tag{6}$$

where static pressure is only affected by depth in static fluid, while the density ρ is a constant. The dynamic pressure represents the kinetic energy possessed by a unit volume of liquid and measures the amount of pressure energy that converts to kinetic energy after stationary liquid is disturbed. Inspired by this formula, we hypothesize that velocity still affects pressure in turbulence to certain extent.

Through the above analysis, we have extracted characteristics such as level set and velocity to describe the characteristics of liquid. In order to better verify the effects of these characteristics on pressure prediction, we make many experiments about the effects of adding them to $\boldsymbol{\beta}$. These specific forms will be discussed in Section 5.2.

4.1.2. *Loss function design*

The feature vector is used to describe the surroundings in which the grid unit is located. It determines the information that the NN can perceive. The loss function determines convergence direction of the network. In the regression problem, the most simple and common loss function is the L-2 norm, which indicates the difference between the predicted value and the ground truth. In this paper, we use L_p to represent the distance:

$$L_p = \left(p - p'\right)^2, \tag{7}$$

where p' is the predicted pressure and p is the ground truth. L_p is a very strong convergence requirement, which requires the prediction results to completely converge to the true value. Yang *et al.* [YYX16] said that there was a stable relationship between input and output of the pressure projection. However, the performance of the flow field depends not only on the accuracy of the local pressure, but more importantly on the global results. Although we are studying a local solution, we still want to add reasonable global pressure feedback to the network. Therefore, just loss function L_p that applied to smoke simulation does not satisfy our requirement. We need to continue exploring other forms that can introduce global feedback.

The purpose of the pressure projection is to smooth out the divergence generated in the flow field after advection and the external force computation, ensuring the passivity of the flow field. Therefore, in the pressure prediction NN, enabling the prediction results to minimize the divergence of the flow field could also become our target. According to this idea, Tompson *et al.* [TSSP17] organized the loss function of the pressure prediction network into an unsupervised form. After the Laplacian is expanded on the MAC grid, Equation (5) becomes:

$$L_d = \left(\nabla \cdot \boldsymbol{u}^* - \left(\sum_{i=1}^{6} p_i' - 6 \cdot p_0'\right)\right)^2, \tag{8}$$

where we use the predicted results of other grids when doing backpropagation based on the prediction of one grid. Under this mutual influence, if we only use L_d as the loss function, the loss will oscillate and not converge with a high probability. Therefore, we combine L_d with L_p, expecting L_d to correct the convergence direction of the L_p. At the same time, we hope that L_p can point out a direction for L_d so that it will move in a converging direction. Note that in order to make every backpropagation advance in the direction that the flow divergence decreases, we construct each batch using a full frame of data. Thus, our network can accept more accurate feedback. Now we get the final form of the loss function f_{loss}:

$$f_{\text{loss}} = \sum (L_p + w \cdot L_d). \tag{9}$$

Here, f_{loss} needs to be summed over all the grids to get the complete information for a frame. Among them, w is a weight used to adjust different units of loss terms. In our experiments, we have tried different values of w such as 0.1, 1, 10, etc., and we have realized that $w = 1$ is a good and simple choice. We will demonstrate the effectiveness of our changes through experiments later.

4.2. Integration of data-driven solver

Our data-driven simulation framework can be easily realized by replacing the numerical iteration solution with our trained NN for solving Poisson equation. After adding the external forces to grid velocity, we calculate the divergence of $\boldsymbol{u}_n^*$ and build the input vector of NN with the information of current frame. Then we run a forward propagation to get the predicted pressure. To initialize the pressure field for NN, the first few frames are computed with the PCG pressure solver to get the initial pressure values. In our experiments, we calculate eight frames. The choice of this number is not so strict, we find that in the range of 5–10, the experimental results do not change significantly.

Algorithm 2 shows the procedure for solving the Poisson equation using our NN. To achieve the goal of quickly solving pressures, we use Algorithm 2 to replace the step in line 7 of Algorithm 1.

5. Experiments

In this section, we will detail the process of obtaining training data and in addition, we will determine the best form of the feature vector and loss function discussed in the previous section through a series of experiments and quantitative analysis. We use *mantaflow* [TP18] as our offline simulator.

5.1. Data acquisition

We use FLIP model to generate data for training. Our training data mainly include two types of scenarios. The first is the free fall movements of liquid, and the second type is breaking dams impact obstacles. To make the distribution of training data more extensive, the

Algorithm 2. Solve Poisson Equation with NN

Input:
Velocity of n_{th} frame $\boldsymbol{u}_n$
Temporary velocity of n_{th} frame $\boldsymbol{u}_n^*$
Levelset of n_{th} frame Φ_n
Pressure of n_{th} frame p_n
Grid occupation O_n
Output:
Pressure of $(n+1)_{th}$ frame p_{n+1}
1: **for** each grid **do**
2: Calculate divergence
3: Build input vector
4: Neural network forward propagation
5: Update grid pressure
6: **end for**
7: **return** p_{n+1}

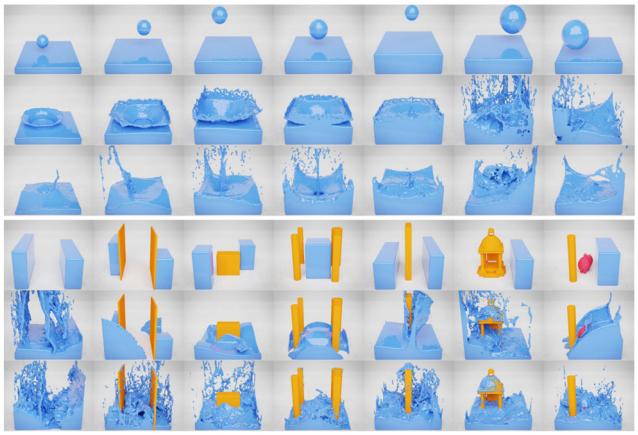

Figure 4: *Scenes in training data: TOP: Free fall droplet; Bottom: Breaking dam impacts different obstacles. We randomly select the size and locations to make our training data as rich as possible. All training data are generated by scenes with resolution of* 80^3.

geometric parameters of liquids and obstacles in each scene are different to cover the conditions encountered in conventional liquid simulations as much as possible. For example, for the first type of scenarios, we vary the position and radius of the water balls, as well as the height of the liquid surface. For the second type, we change the position and height of the dam, as well as the shape and position of the obstacles. Figure 4 shows part of the simulation scenes to collect the training set.

In our experiments, we run seven simulations for each type of scenarios with five simulations as the training set and the rest as the test set. Each simulation has an operation of 900 frames, and we group data of one frame every eight frames to make one batch according to the analysis in Section 4.1.2. All the simulation we generated (which will serve as training data) were performed at a grid resolution of 80^3, this low-resolution simulation allowed us to get data quickly. From the simulation scenarios, we have obtained about 50 million training samples.

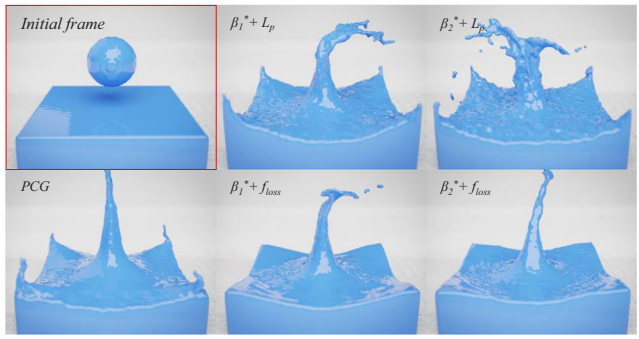

Figure 5: *A water ball hits the surface (*80^3*). We tried different combinations of feature vector and loss function. Visually, the neural network trained only by* L_p *as the loss function produces obvious artefacts. After adding* L_d *to the loss function, the results become better.*

5.2. Training neural network

We implement our NN using the *PyTorch* framework and the *Adam* optimizer. In previous section, we have discussed the possible components of the feature vector and loss function that have influences to the training model. To find the best design, we have trained our network with different combinations. In the following, we will illustrate and discuss the simulation results generated by different combinations.

We have discussed the difference in physical form between smoke and liquid in the previous section, so we add a level set to the feature vector to describe the surface of the liquid. Inspired by the Bernoulli equation, we find that velocity also has a large effect on the pressure of the fluid; therefore, we also consider the effect of velocity. However, we cannot obtain the exact form of velocity difference before pressure solving, so we test two feature vectors $\boldsymbol{\beta}_1^*$ and $\boldsymbol{\beta}_2^*$, as shown in Equation (10),

$$\begin{aligned}\boldsymbol{\beta}_1^* &= \left\{\boldsymbol{\beta}, \nabla\cdot\boldsymbol{u}_0, \bigcup_{i=0}^{6}\{\Phi_i\}, \boldsymbol{u}_0\right\},\\ \boldsymbol{\beta}_2^* &= \left\{\boldsymbol{\beta}, \nabla\cdot\boldsymbol{u}_0, \bigcup_{i=0}^{6}\{\Phi_i\}, \Delta|\boldsymbol{u}_0|^2\right\}.\end{aligned} \tag{10}$$

Among them, $\boldsymbol{u}_0$ represents the intermediate velocity of the grid, and $\Delta|\boldsymbol{u}_0|^2$ represents the difference between the square of the intermediate velocity and the initial velocity of the grid. To prove the effectiveness of adding a divergence item to the loss function, we train our NN with L_p and f_{loss}, respectively. Figure 5 compares the results of our network trained with different loss functions. The results generated by network with L_p show significant visual artefacts. However, after we add L_d into it, the fluid surface becomes more stable and smooth. It proves that the divergence term can effectively improve the simulation results. Therefore, in the following experiments, all the loss function is with the addition of L_d.

After determining the loss function, we will discuss the effect of the two feature vectors on the result. We use two NNs trained with $\boldsymbol{\beta}_1^*$ and $\boldsymbol{\beta}_2^*$, respectively, and compare their results with the PCG solution in the same initialization case. Figure 6 shows the results

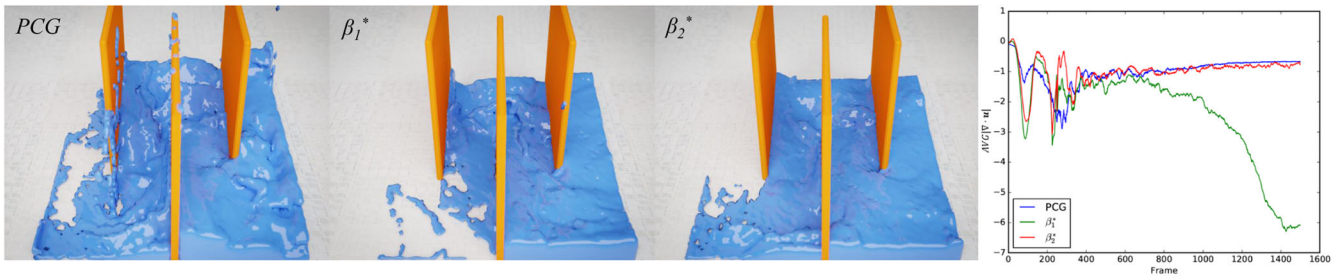

Figure 6: *The results (80^3) produced from different neural networks trained with β_1^* and β_2^* compared with PCG. As can be seen from the quantitative statistics of divergence, the network trained with β_2^* is better than that with β_1^*. Although the divergence is large at the beginning, it is acceptable compared to PCG.*

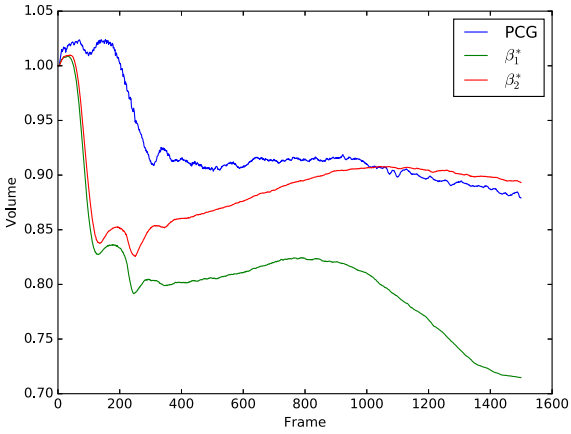

Figure 7: *The volume curves obtained by β_1^* and β_2^* compared with PCG. As can be seen from the figure, the trend of volume change is similar to that of divergence change. In the stage of severe impact, all three methods have the problem of volume loss. But after the motion slows down, the volume changes very gently. β_2^*, by contrast, performed better than β_1^*.*

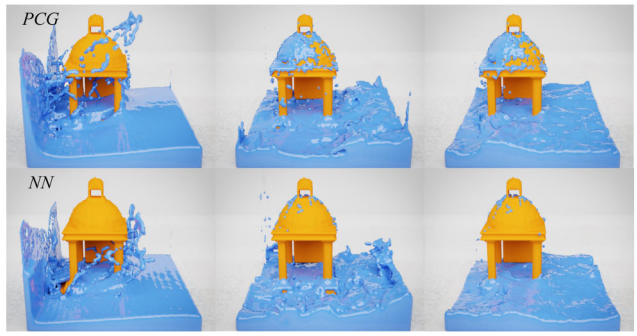

Figure 8: *Comparison between PCG and our network. Top: Simulation based on PCG solver; Bottom: Simulation based on our neural network. For scenes that appear in the training set, our approach produces subtle differences, but the overall results are similar. And our method only take one-seventh of the time. These two scenes have same resolution of 80^3.*

produced by the three different pressure solvers at frame 450. To quantify the influences, we calculate the changes in average absolute divergence over time, as shown in Figure 6 (far right). It can be seen from the statistics that whether it is an NN or PCG solver, the divergence is relatively large at the beginning since liquid is undergoing a vigorous impact movement, our experiment results are acceptable when compared with the PCG solution. After 300 frames, the result of β_2^* is obviously better than that of β_1^*, and it is close to PCG. In addition, we also calculated the volumetric change diagrams of the three results, as shown in Figure 7. It can be seen that all three results have the problem of volume loss. In comparison, β_2^* is better than β_1^*. So, we would like to choose β_2^* as the input vector of our network. That means that the input layer of our NN is a 28-dimensional vector, and the number of neurons in the hidden layer is 56. Moreover, the number of all trainable parameters of the network is about 11 000.

In addition, after determining the feature vector and loss function, we design and train a linear NN to prove the non-linearity of the task. The structure of a linear NN is similar to the network structure described above, except for the activation function. Here, we use a linear activation function $a(x) = x$. The figure below shows the simulation results using a non-linear activation function. Significant artefacts appear in the simulation results. This can explain the rationality of treating pressure solving tasks as non-linear tasks.

5.3. Applications

Our data-driven simulation method with integrated NN pressure predictor can achieve different applications. The experimental results show that our NN pressure solver can reasonably predict the pressure under various conditions.

5.3.1. *Reconstruction*

Figure 8 shows a simulation comparison of two scenarios solved using PCG and NNs, respectively. In general, our method can restore the scenes in the training set under the same parameters and achieve similar results. Note that our purpose is not to completely reproduce the original scenes, since we have included the divergence term in the loss function.

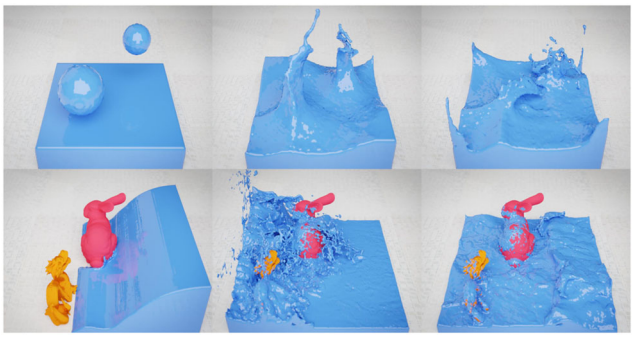

Figure 9: *New simulations with our methods: TOP (80^3): Two water balls hit the surface; Bottom (160^3): A breaking dam impacts two animals. Our methods can generate realistic effect in different scenes with different resolutions.*

5.3.2. *Scale expansion*

Our training data were collected from simulations with a static resolution of 80^3. As shown in Figure 10, to simulate more sufficient details and more realistic behaviours, we apply the trained models to scenarios of various grid scales by a quite simple but useful manner that stitches the scenes of 80^3 resolution proportionally to expand the size of the simulation scenes. It should be pointed out that although the expanded scale in the figure is a multiple of the original scale, theoretically, our method can be applied to grids of any size.

Our training data are sampled from simulations with a constant space size Δh of $1/80$ and the NN has only learned the mapping of data collected from Δh. Directly expanding multi-scale scenes will lead to serious compression problems because of the mismatched space steps between training set and generating scenes. However, we fix the spatial step size as Δh and consider changing the spatial scale to $R/80$, where R is the target grid scale. It is not a complicated procedure but greatly improves liquid incompressibility and achieves good results. Figure 11 shows the simulation results of our method at different spatial scales, which illustrates that our method has good scalability and displays more realistic details than low-resolution dose. This variable grid scales method provides a simple and efficient way, which achieves as good performance as high-resolution simulations.

5.3.3. *New scenes synthesis*

The NN we propose mainly focuses on the local solution of pressure, which is not affected by the global composition. Therefore, for scenes that do not appear in the training set, as long as the pressure values involved in database, our method can generate liquid simulation with realistic visual effects. Since we have almost 50 million training samples, it could cover the vast majority of pressure changes. With our trained network, once new boundaries and liquid initial conditions are provided, we can generate new liquid animations in a very short time. Figure 9 shows the results of our approach to simulate new scenarios. Both these two scenarios have not appeared in our training set, we can still simulate the realistic behaviours of liquid animation and liquid–solid interactions.

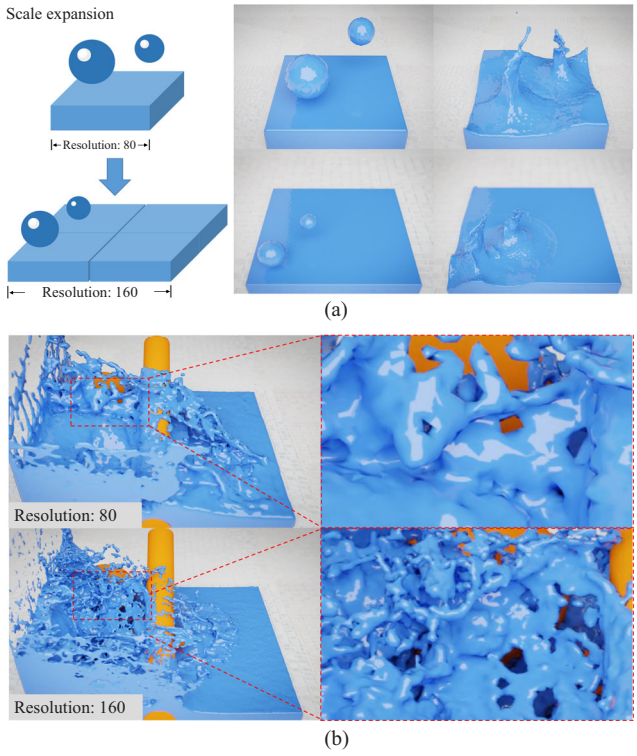

Figure 10: *(a) Illustration of expanding large-scale scenes. Take expansion of 80^3 resolution to 160^3 as example, the original scene involves two water balls falling free. We stitch three same scale areas to generate a new scene with two times grid scale. (b) Comparison between different grid scales. To get a clear contrast, we expand a scene of resolution 80^3–160^3 and double the obstacle size at the same time. It can be seen that the simulation with larger scale can display more sufficient details.*

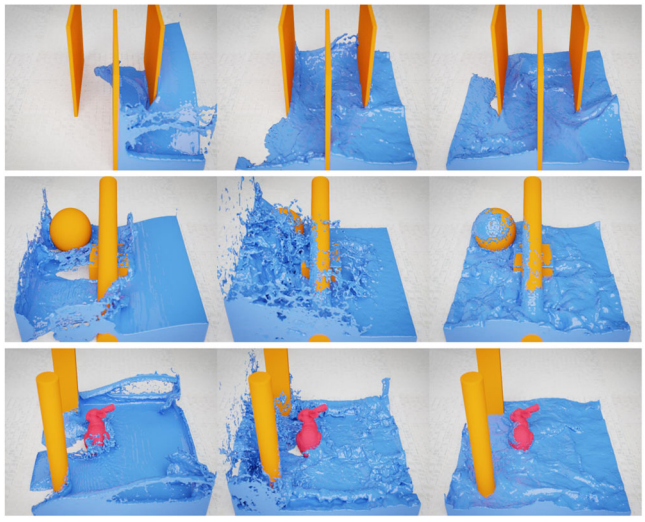

Figure 11: *Generated scenes of different resolutions. Our methods can handle varying spatial scale, the resolutions are 128^3, 160^3 and 240^3 from top to bottom.*

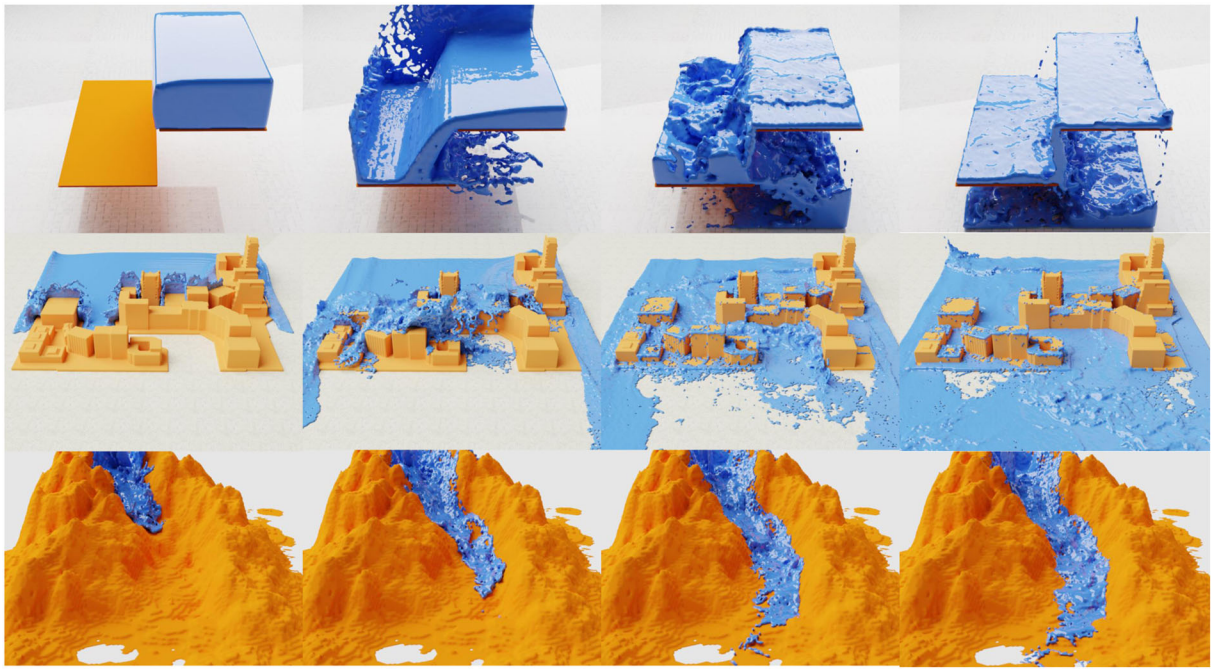

Figure 12: *Top (80^3): A water block pouring to two steps; Middle (350^3): Flood hits city models in a wide open boundary scenario. Bottom (370^3): Flood hits the mountain model. These scenes are quite different from the training data.*

In order to showcase the generalizability of our approach, we have also conducted an experiment which is significantly different from the training data, as shown in Figure 12 (top). It shows the water block pouring to two steps. Besides, we also do another two experiments of flood hitting city (Figure 12 middle). As shown in Figure 12 bottom, flood hits the mountain (Figure 12 bottom) in wide open boundaries. In this scene, both the resolution and phenomenon are too different to the training data, it is difficult for the NN to reasonably learn the pressure of this high-resolution scene, which causes obvious volume compression.

5.4. Analysis and discussion

5.4.1. *Time efficiency*

The key innovation in this paper is improving the efficiency of liquid simulation while ensuring visual effects. In PCG solution, the factors affect the computation time to include the complexity of the linear system and the number of iterations. Actually, when generating training set, the linear system calculation time mainly depends on the number of grid cells that need to be solved for pressure. In practice, the number of liquid grid cells in a simulated scene is generally much smaller than the total number of grids divided from the entire space, so we can save a lot of storage consumption. At the same time, both our training model and generated model are implemented on GPU algorithm to further improve efficiency. Table 1 shows the average calculation time per frame for several scenes with different resolutions. It can be seen that our method is much more efficient than traditional solvers, and as the resolutions increase, the advantages of our approach are becoming more obvious.

Table 1: *Comparison of average calculation time in different scenes and resolutions (average time per frame). Time is measured in milliseconds. The accuracy of PCG solution is 10^{-5}, and the accuracy of Gauss–Seidel iteration and Jacobi iteration is 10^{-4}. All of these iterations use CUDA for parallel computing acceleration with an RXT-2080 GPU.*

Scenes	Res	PCG	Jacobi	Gauss–Seidel	NN
Figure 8 (Bottom)	80^3	51	34	43	7
Figure 9 (Top)	80^3	71	41	47	7
Figure 9 (Bottom)	160^3	123	101	111	23
Figure 11 (Top)	128^3	80	69	77	11
Figure 11 (Middle)	160^3	149	110	119	23
Figure 11 (Bottom)	240^3	622	579	522	43
Figure 12 (Top)	80^3	73	46	59	10
Figure 12 (Middle)	350^3	–	–	–	84
Figure 12 (Bottom)	370^3	–	–	–	112

In the case of low-accuracy requirements, our method has extremely fast calculation speed with acceptable visual performance. However, for other applications like accurate incompressible fluid simulations, our method cannot control the accuracy of the pressure calculation. So, we cannot conclude from this table that our method is superior to other numerical methods.

5.5. Long-term stability

To demonstrate the long-term stability of our method, we perform an experiment in which a liquid tank makes a sinusoidal, nonstop

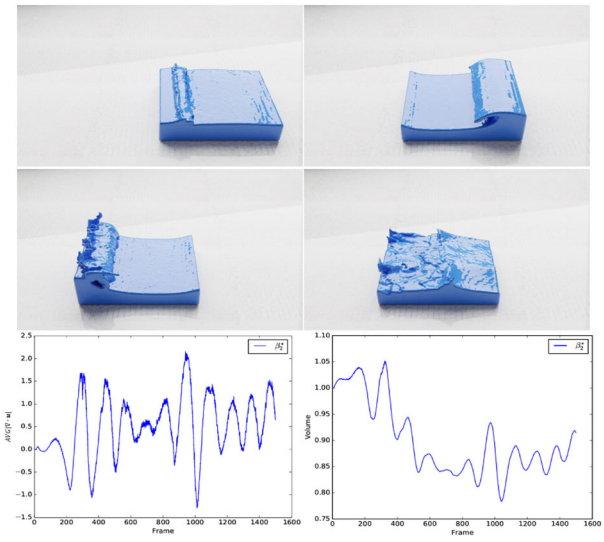

Figure 13: *Reciprocating tank. The top three pictures are renderings of the tank experiment at different times. The two graphs below are the divergence and volume curves.*

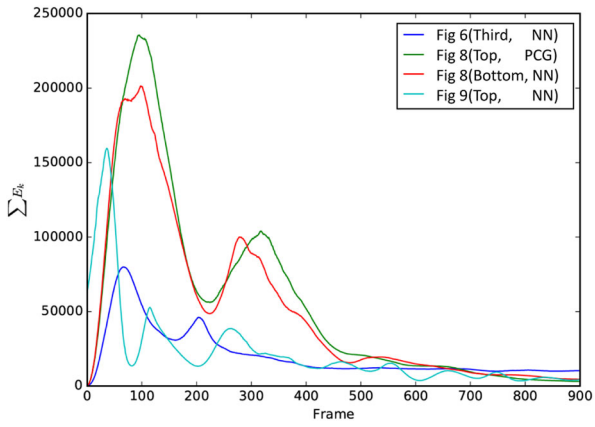

Figure 14: *Kinetic energy of different scenes. We calculate the trend of kinetic energy over time for different scenes, and the experiments have good stability and convergence.*

left and right reciprocating motion, as shown in Figure 13. As seen from the figure, the liquid can still maintain the correct physics during the reciprocating motion, divergence varies periodically with the frequency of motion. In addition, we also plot the volume change curve, and find that the volume is also changing periodically but approximately within a stable range.

5.5.1. *Convergence*

To test the convergence of our method, we calculate the kinetic energy of three representative scenarios. As shown in Figure 14, the kinetic energy changes as liquid collides with boundaries or streams as time goes by, and finally decays and converges to zero. We could conclude that our proposed method has good stability and convergence.

5.5.2. *Limitations*

A non-negligible problem of our method is the volume compression. Since it is a local solution method, it may lead to the phenomenon of 'unresponsiveness' of the liquid. For example, when the fluid hits an obstacle, the fluid in the front part has already touched the obstacle and starts to decelerate, but the latter part fluid cannot sense the information of the collision in time, and continues to move at the current speed, which may lead to a high probability of generating volume compression. More severe the liquid impact motion is, the compression problems may become more obvious.

Our method also encounter similar problems as other deep learning methods. Because the training data cannot cover all cases, the pressure solvers may not reasonably predict situations that are too far away from the training data, which limits the range of resolution that the NN can expand to. All our training data are collected from simulation scenarios with a resolution of 80^3. The data distribution that can be collected from these scenarios is sufficient. However, when the resolution of the scene is increased to 240^3 or higher, it is difficult for the NN to learn from the low-resolution training data to reasonably extrapolate the pressure of the high-resolution scene, which will also cause volume compression (Figure 12 (bottom)).

6. Conclusion

In this paper, we proposed a new approach of solving liquid simulation using an improved data-driven method. We have analysed and discussed the differences between liquids and smokes during simulation, and tried different combinations between feature vectors and loss functions of the NN. By adding information such as the level set and velocity of the liquid to the input, the NN could more accurately sense the state of liquid grids. To accurately predict the pressure and minimize the global divergence, we designed loss functions based on pressure and divergence, which could give our network more accurate pressure feedback. Benefiting from the characteristic of local solution, our pressure prediction network could easily expand to various simulation scales through a simple improvement of spatial step size.

Compared with the traditional method, our method greatly improves the solution efficiency of Poisson equation while using pressures of only few frames from numerical solver. What is more, with our trained NN, we could generate multifarious new simulations of various boundary conditions and expand to generate different scenes with varying scales very fast. Our future works include the high-resolution simulations through subdividing grids in special size to achieve multi-resolution simulations. In addition, our method uses dimensional input and output. The trained network model only learns the data mapping relationship under a certain time, space step and density. If one of these variables is changed, the model cannot reasonably calculate the pressure. Considering the dimensionless input and output of the model to achieve variable space step size and density simulation is also a direction worth of further studies.

Acknowledgement

This research is supported in part by National Key R & D Program of China (No. 2017YFF0106407), Beijing Advanced Innovation Center for Biomedical Engineering (ZF138G1714), National Natural Science Foundation of China (NO. 61672077 and 61532002), National Science Foundation of USA (IIS-1715985 and IIS-1812606), Key R & D Program of Zhejiang Province (No. 2020C03061) and Beijing Natural Science Foundation Haidian Primitive Innovation Joint Fund (No. L182016).

References

[APKG07] Adams B., Pauly M., Keiser R., Guibas L. J.: Adaptively sampled particle fluids. *ACM Transactions on Graphics (TOG) 26*, 3 (2007), 48.

[ATW13] Ando R., Thürey N., Wojtan C.: Highly adaptive liquid simulations on tetrahedral meshes. *ACM Transactions on Graphics (TOG) 32*, 4 (2013), 103.

[Bri15] Bridson R.: *Fluid Simulation for Computer Graphics*. AK Peters/CRC Press, Natick, 2015.

[CM11] Chentanez N., Müller M.: Real-time Eulerian water simulation using a restricted tall cell grid. *ACM Transactions on Graphics (TOG) 30*, 4 (2011), 82.

[CT17] Chu M., Thuerey N.: Data-driven synthesis of smoke flows with CNN-based feature descriptors. *ACM Transactions on Graphics (TOG) 36*, 4 (2017), 69.

[DC99] Desbrun M., Cani M.-P.: *Space-Time Adaptive Simulation of Highly Deformable Substances*. PhD thesis, INRIA, 1999.

[EMF02] Enright D. P., Marschner S. R., Fedkiw R. P.: Animation and rendering of complex water surfaces. *ACM Transactions on Graphics (TOG) 21*, 3 (2002), 736–744.

[ENGF03] Enright D., Nguyen D., Gibou F., Fedkiw R.: Using the particle level set method and a second order accurate pressure boundary condition for free surface flows. In *ASME/JSME Joint Fluids Summer Engineering Conference* (2003), pp. 337–342.

[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.

[FF01] Foster N., Fedkiw R.: Practical animation of liquids. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (2001), pp. 23–30.

[FM96] Foster N., Metaxas D.: Realistic animation of liquids. *Graphical Models and Image Processing 58*, 5 (1996), 471–483.

[GLQ*19] Gao Y., Li S., Qin H., Xu Y., Hao A.: An efficient FLIP and shape matching coupled method for fluid–solid and two-phase fluid simulations. *The Visual Computer 35*, 12 (2019), 1741–1753.

[GLQH17] Gao Y., Li S., Qin H., Hao A.: A novel fluid-solid coupling framework integrating FLIP and shape matching methods. In *Proceedings of the Computer Graphics International Conference* (2017), pp. 11:1–11:6.

[ICS*14] Ihmsen M., Cornelis J., Solenthaler B., Horvath C., Teschner M.: Implicit incompressible SPH. *IEEE Transactions on Visualization and Computer Graphics 20*, 3 (2014), 426–435.

[JSP*15] Jeong S., Solenthaler B., Pollefeys M., Gross M., Gross M.: Data-driven fluid simulations using regression forests. *ACM Transactions on Graphics (TOG) 34*, 6 (2015), 199.

[KD13] Kim T., Delaney J.: Subspace fluid re-simulation. *ACM Transactions on Graphics (TOG) 32*, 4 (2013), 62.

[MCPN08] Molemaker J., Cohen J. M., Patel S., Noh J.: Low viscosity flow simulations for animation. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2008), pp. 9–18.

[Mon92] Monaghan J. J.: Smoothed particle hydrodynamics. *Annual Review of Astronomy and Astrophysics 30*, 1 (1992), 543–574.

[SKK07] Song O.-y., Kim D., Ko H.-S.: Derivative particles for simulating detailed movements of fluids. *IEEE Transactions on Visualization and Computer Graphics 13*, 4 (2007), 711–719.

[SP09] Solenthaler B., Pajarola R.: Predictive-corrective incompressible SPH. *ACM Transactions on Graphics 28*, 3 (2009), 40.

[TDF*15] Takahashi T., Dobashi Y., Fujishiro I., Nishita T., Lin M. C.: Implicit formulation for SPH-based viscous fluids. *Computer Graphics Forum 34*, 2 (2015), 493–502.

[TLP06] Treuille A., Lewis A., Popović Z.: Model reduction for real-time fluids. *ACM Transactions on Graphics (TOG) 25*, 3 (2006), 826–834.

[TP18] Thuerey N., Pfaff T.: MantaFlow, 2018. http://mantaflow.com.

[TSSP17] Tompson J., Schlachter K., Sprechmann P., Perlin K.: Accelerating Eulerian fluid simulation with convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning* (2017), pp. 3424–3433.

[UHT18] Um K., Hu X., Thuerey N.: Liquid splash modeling with neural networks. *Computer Graphics Forum 37*, 8 (2018), 171–182.

[WBT18] Wiewel S., Becher M., Thuerey N.: Latent-space physics: Towards learning the temporal evolution of fluid flow. arXiv preprint arXiv:1802.10123 (2018).

[XFCT18] Xie Y., Franz E., Chu M., Thuerey N.: TempoGAN: A temporally coherent, volumetric GAN for super-resolution fluid flow. arXiv preprint arXiv:1801.09710 (2018).

[YYX16] Yang C., Yang X., Xiao X.: Data-driven projection method in fluid simulation. *Computer Animation and Virtual Worlds 27*, *3–4* (2016), 415–424.

[ZGL*18] Zheng Z., Gao Y., Li S., Qin H., Hao A.: Robust and efficient SPH simulation for high-speed fluids with the dynamic particle partitioning method. In *Pacific Graphics* (2018).

[ZHQH] Zhai X., Hou F., Qin H., Hao A.: Fluid simulation with adaptive staggered power particles on gpus. *IEEE Transactions on Visualization and Computer Graphics*. https://ieeexplore.ieee.org/abstract/document/8573859

[ZHQH17] Zhai X., Hou F., Qin H., Hao A.: Inverse modelling of incompressible gas flow in subspace. *Computer Graphics Forum 36*, 6 (2017), 100–111. https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12861

Supporting Information

Additional supporting information may be found online in the Supporting Information section at the end of the article.

Video S1