The Visual Computer
https://doi.org/10.1007/s00371-020-02025-x

ORIGINAL ARTICLE

Solving the fluid pressure with an iterative multi-resolution guided network

Rong-Jie Xu[1] · Bo Ren[1]

Accepted: 12 November 2020

Abstract

In Eulerian methods, the simulation of an incompressible fluid field requires a pressure field solution, which takes a large amount of time and computation resources to solve a large coarse linear system. The pressure solver has two mathematical features. The first is that it obtains the pressure solution from a velocity divergence distribution in high-dimensional space. The second is that the pressure is iteratively solved in the projection step. Based on these two features, we investigate a convolutional-based neural network, which learns to map the fluid quantities to pressure solution iteratively by inferring from multiple grid scales. Our proposed network extracts features from multiple scales and then aligns them to obtain a pressure field in the original resolution. We trim our network structure to be compact and fast and design it to be iterative like to improve performance. Our approach requires less computation cost, while it achieves comparable performance with recently proposed data-driven methods. Our method can easily be parallelized in GPU devices, and we demonstrate its speed-up ability with the fluid field in larger input scenes.

Keywords Eulerian fluid simulation · Acceleration · Convolutional neural network (CNN) · Pressure solution

1 Introduction

Realistic and highly detailed fluid flow scenarios are widely used in the gaming and movie industry. In order to simulate such a qualified fluid animation, one common approach is to solve the incompressible Navier–Stokes (NS) equations in discrete numerical models and to solve the model in finer resolution, which enables the simulation to capture more details of the complex turbulent flow [1]. The Eulerian method requires enormous computational cost, especially in the "projection" procedure, which resolves an incompressible fluid flow. Although the numerical solutions for the NS equations have been a long-standing research topic in the graphics area, an efficient solution for an incompressible velocity field is still a primary challenge as the fluid field increases to a real-life scale. In our work, we aim at saving the computational cost while restoring real-life fluid details. Instead of resolving a large linear system, we target at inferring the divergence-free velocity information directly from the fluid quantities. As an analogy to image-to-image translation [6], the volumetric fluid data can be "translated" to obtain the desired velocity solution with the help of learning-based methods, like convolutional neural network (CNN). Despite the success in image processing and generation, CNN-based methods need additional modifications to achieve qualified results when applied in volumetric fluid data. We combine CNN descriptors with physical laws and design our network structure to maximize its capability. We will demonstrate that our designed learning-based model is effective to extend the advantages of traditional methods and avoid the unnecessary computational cost.

Recovering fluid flow details while saving computational cost is a long-standing topic in the graphics community. Some previous researches try to enrich an existing fluid simulation with artificial details and, consequently, ensure the simulated outcomes with fine-resolution turbulence and vorticity. The research in [9] utilizes wavelet turbulence and enriches the original coarse fluid field with procedural methods. The work in [30] improves the temporal coherence artifact with

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s00371-020-02025-x.

✉ Bo Ren
rb@nankai.edu.cn

Rong-Jie Xu
xrj@mail.nankai.edu.cn

[1] Nankai University, Tianjin, China

a discriminator which tracks the flow field quantities overtime and learns to generate detailed fluid field from a coarse version under the temporal coherence discrimination. Such solutions, basically synthesizing high-resolution flow field from a coarse version by frame-to-frame upsampling, have been proven effective for visual effect. Both two solutions require additional efforts to ensure the temporal coherence of fluid animation in consecutive time steps. In order to be physically faithful, the model in [28] accelerates fluid simulation by replacing the standard pressure solution with a convolutional neural network, which learns the distribution of the pressure field from extracted features from the original fluid field. They apply most of the mathematical calculation in the original scale, and as a result, the total convolutional computation cost is still considerable. The work in [27] trains a neural network to infer the velocity and pressure field in the turbulent flow. Motivated by previous work, we start from a similar viewpoint but with different approaches. We replace the traditional pressure solution in fluid simulation with a feature extraction network for reducing the dimensionality of an original fluid field and an alignment strategy for aligning the coarse and the dense grid scale of the fluid features. Our pressure solution network is like an iterative process to adapt to different inputs. In this way, the traditional fluid simulation process is facilitated by the CNN methods, leading to an efficient and accurate fluid simulation.

In our approach, the fluid information from the original resolution first goes through a feature extraction strategy, during which the features with reduced dimensionality are fused in multiple layers. Then, the features from different layers are combined to obtain the pressure field, where special care is taken to preserve the alignment coherence in order to avoid excess error accumulation and simulation breakdown. The above process can be contained in a CNN block structure and self-stacked, iteratively refining the output. Thus, we name our presented model BlockNN, and we will introduce it in detail in the following sections.

Our contributions are as follows:

- A novel pressure solver that uses CNN to extract features into multiple coarse layers, then combines and aligns the fused features back to the original resolution to obtain the pressure field.
- An iterative-like network architecture that can be applied dynamically in runtime to improve the outcome accuracy.
- A compact network design that can be quick to converge during training and save computation cost in runtime.

2 Related works

The realistic animation of fluid, such as water waves, smoke plumes and other phenomena, has long been an important research topic in the graphics community [14]. The essential approach to model the dynamics of fluid is to solve the physical Navier–Stokes equation. The reference book [1] introduces the basic process of solving the Navier–Stokes equation numerically with the following steps, adding external forces, advecting velocity field, computing diffusion and applying incompressible projection operation. After the introduction of an unconditionally stable model, using a combination of a semi-Lagrangian advection method and implicit solvers [23], the stable Eulerian methods have been developed with many additional efforts to enhance fluid flow reality. A common problem is that the first order integration and finite resolution of an Eulerian grid would cause numerical artifacts in fluid simulation. Algorithms in [4] and [21] estimate the underlying numerical error and correct it. The numerical deficiencies of the grid structure can also be reduced by using Lagrangian methods [15]. The previously mentioned works improve the visual result of the Navier–Stokes solver by refining the underlying numerical method. Another way to enhance physical reality is to add artificial details. [9] works on the frequency spectrum and utilizes the frequency information in the fluid field. A more realistic simulation could be achieved by adding artificial noise patterns in higher frequency domains to the existing fluid field. As our method mainly focuses on the solution of the incompressible velocity field, these early accurate advection algorithms can work well in combination with our proposed scheme.

While many efforts have been made to improve the fluid simulation performance, the computation and memory cost of a fluid solver is still a remained problem. [17] comes up with a multi-grid preprocessing method, which aims to speed up the PCG solution in large-scale scenarios. However, its solving complexity depends greatly on the input data. When facing complicated and irregular boundary settings, it requires nearly the same time to process the input field. An alternative method [18] makes use of particles to model fluid motion, and such a method can be performed in a parallel GPU setting. However, incompressibility is not straightforward to be ensured in particle methods [3]. Our method shares the advantages of both the grid approaches and particle approaches. The projection procedure is solved with our multi-resolution CNN model, which is natural to perform in parallel in GPU devices. In this way, our presented model can speed up calculation with parallelization and save the overall computation cost.

The partial differential equations (PDEs) are required to be solved for the pressure field during the fluid simulation. There are attempts to solve the PDEs with artificial neural networks for years [10,11,19,22]. [22] trains a deep neural network to satisfy the differential operator, initial condition and boundary condition, and they solve the PDEs in up to 200 dimensions. As the PDEs are used to describe physical phenomena, [19] uses physical law as prior information and

Journal: 371 MS: 2025 TYPESET DISK LE CP Disp.:2020/11/30 Pages: 10 Layout: Large

train their network to learn the nonlinear PDEs. Similarly, our research also makes use of the semantic information of the correspondence of fluid divergence and pressure field, but we target at PDEs with millions of dimensions.

In previous years, the research in data-driven methods has achieved tremendous success in the area of object detection [24], volume translation [6], and so on. Convolutional neural network (CNN) is one of the most popular techniques in the field of image and video processing [12]. CNN is shown to be more powerful and effective than fully connected layers, and its deep version is capable of classification, feature extraction and geometry inference [24] in 2D and 3D domain. The research in [6] shows that deep CNN can learn representation from unlabeled data and generate the corresponding natural image. Data-driven methods are also shown effective in learning the temporal evolutions of physical functions in [29], though their work does not include the consideration of solid obstacles in the simulation domain. As in our case, we demonstrate that the features and geometric information of fluid volumetric data can also be extracted by CNN, and hence, the distribution of fluid pressure filed can be inferred. In addition, the research of network architecture takes an active role in the development history of deep learning. The U-Net structure allows the network to localize and propagate content information by skip connection [20]. In [27], authors utilize a U-Net architecture to infer pressure and velocity distribution. Residual block architecture avoids deep layers from degradation [5], and the enhanced deep residual network proposed in [13] improves the performance in super-resolution tasks. In our case, we make use of the residual blocks in the hidden layers in our model to infer fluid features.

In the field of learning-based fluid simulation, there are also many successful practices. Deep learning methods can be utilized to capture fluid data distribution and be combined with applications like interpolations of different parameters during fluid animation [8], and super-resolution with generative adversarial neural network (GAN) that synthesizes fluid details as well as ensures temporal coherence [30]. Fluid motions can also be predicted with learning-based approaches. [7] replaces the SPH method with a random regression tree by predicting each particle's state with the states of nearby particles. This prediction scheme using the local information has also been proposed in [28,31]. [31] trains a convolution neural network and predicts the pressure field in the next time step by taking local geometry, velocity divergence and pressure field from the last time step as input. [28] tries to solve the problem in a similar way, but it uses an unsupervised training strategy and ensures the generalization capability of the trained model. This approach shares the most similar idea with ours. We also target at the solution of the incompressible velocity field. But in our model, we combine the pressure solution and velocity projection in a typical iterative structure to improve the inference ability.

3 Fluid field solution

3.1 Fluid dynamics

Physical-based fluid simulation can be modeled by the Navier–Stokes equation [25]. An ideal fluid without viscosity can be described by the following simplified NS equations, also known as incompressible Euler equations:

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

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

where $\mathbf{u}$ represents the velocity(a 2D or 3D vector field when simulating in 2D or 3D scene), t denotes time, p denotes the pressure, f is the overall external forces and ρ is the fluid density. Equation 1 describes the fluid motion with Newton's second law and explains how the fluid changes the velocity with given forces acting on it. Equation 2 is the incompressible fluid condition, which ensures the volume of the fluid remains constant. In order to solve the NS equations, the graphics community usually split the process into three steps, i.e., advection, force applying and projection. In the advection step, fluid advects their property according to their velocity. In the force-adding step, we add external forces on fluid for a small time interval. And in order to acquire incompressible fluid, we apply the "projection" step to the existing fluid field, which includes acquiring the pressure field by solving the Poisson equation:

$$\nabla \cdot \nabla p = \frac{1}{\nabla t} \nabla \cdot \mathbf{u}, \tag{3}$$

and applying velocity field correction with the gradient of the pressure as

$$\mathbf{u}_{t+1} = \mathbf{u}_t - \triangle t \nabla p_t. \tag{4}$$

3.2 Boosting projection solution

Solving Eq. 3 is equivalent to solve a large and sparse linear system, whose size increases exponentially as the input size increases. Iterative methods like PCG and Jacobi, are usually used to solve the mathematical problem. As the input size increases, the memory cost and computation cost increases. The iteration number grows dramatically to converge. However, high resolution simulation of the realistic fluid field has always been the chasing goal of real-life applications.

In our work, we aim at replacing the traditional pressure solution with deep learning approaches. The work in [28] and [31] sheds some light on exploring the distribution and correspondence rules of the fluid divergence field and pressure field with a convolutional neural network. As their work

Journal: 371 MS: 2025 TYPESET DISK LE CP Disp.:2020/11/30 Pages: 10 Layout: Large

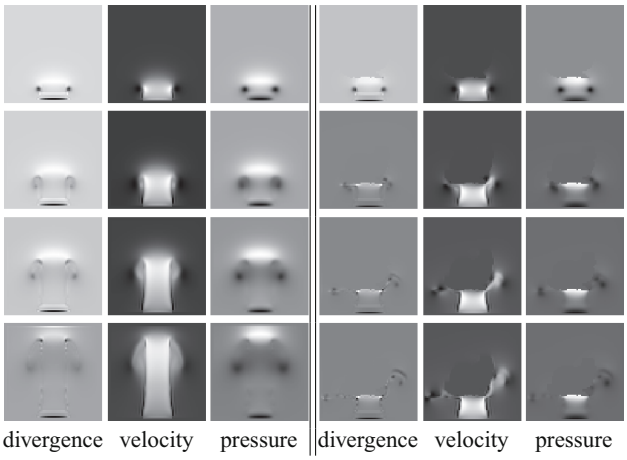


Fig. 1 Correspondences among velocity divergence, velocity and pressure. The first three columns show the fluid field quantities of a smoke plume scenario in different time steps (from top to bottom). The last three columns show fluid quantities in another fluid scenario with obstacles. The divergence and the corresponding pressure field seem to have a closer relationship

did provide a feasible solution, the convolution computation cost is still not negligible. Although the convolution operation largely reduces the size of trainable parameters and computation cost compared to a fully connected layer, the computation and time command grows dramatically as the input fluid size increases, especially when simulating the fluid field in three-dimensional domain. This not only holds back the performance of the solution but also limits the design of a more effective network structure.

We adopt a similar idea from previous work of replacing the solution of the Poisson equation with data-driven approaches [28,31]. The pressure solution can be replaced as an "image translation" problem, which is typically a 3D image for the fluid field in three dimensions. We visualize the velocity, divergence and the corresponding pressure in Fig. 1, and we can infer from the visualizations intuitively that the fluid quantities can be interpreted and translated as images.

Facing the existing problems, we design our network structure and make it flexible to achieve comparable outcome accuracy with much less computation. We apply convolutional multiplications in the fluid field and gradually extract features in the coarse grid to solve for a pressure field. As the pressure field is solved by inferring from the coarse grid, we design an additional connection layer to fuse up the coarse grid features and the original fluid quantities, so that the pressure field solution is aligned with the original resolution. Such alignment strategy in the network provides fluid simulation with the same level of details but requires much less computation cost. In this way, we can apply most convolutional manipulation in the coarse grid, and thus, a large scale of computation cost is reduced.

Algorithm 1: Boosting solution for the NS equation

initialize a velocity field $\mathbf{u}_0$;
for *time step t = 1,2,...* **do**
  advect fluid components with $\mathbf{u}_{t-1}$;
  advect self velocity field with $\mathbf{u}_{t-1}$;
  apply additional force ;
  input divergence $\nabla \cdot \mathbf{u}_{t-1}$ and obstacle flags to the pressure solver model(BlockNN);
  solve the incompressible velocity $\mathbf{u}_t$ with the BlockNN model output p_t ;
  check and satisfy the boundary condition;
end

4 Pressure solver model

Our pressure solver model is composed of convolutional layers and traditional fluid operation, i.e., divergence operation and velocity correction. We design our pressure solver network with the consideration of the input selection, network structure and training strategy, making it more powerful and efficient in varied situations. We propose the OneBlock model, to extract features from the fluid field in different resolutions, make alignments between the fused features and output the corresponding pressure field. To improve the model capability and tackle complicated fluid scenarios, we stack several OneBlock structures in our model and connect every two consecutive blocks with velocity correction and divergence calculation. This whole network arrangement is composed of several stacked OneBlock structures; hence, we name our pressure solver network BlockNN and its structure overview is shown in Fig. 2. We will clarify the whole model structure in the following subsections.

4.1 Network input and training targets

We target to replace the Poisson equation solution with our iterative BlockNN network. Referring to the traditional projection process in Eq. 3, we at least need the information of the velocity divergence and the fluid geometry setting to solve a pressure field. So we take the velocity divergence $\nabla \cdot \mathbf{u}_t$ and the obstacle flags f as our network input.

We want the pressure field to make the fluid field more incompressible, so we use L2 norm of the outcome divergence of the velocity field in the final outputs, represented as $\mathbf{u}$, to measure the loss during training. As the loss function shown in Eq. 5, the whole training procedure is unsupervised:

$$\text{Loss} = \|\nabla \cdot \mathbf{u}\|^2. \tag{5}$$

For the completeness of our exploration, we have also investigated to employ the fluid velocity as input, but that indirect information causes difficulties during learning and will not lead to a comparable training result. If we use the

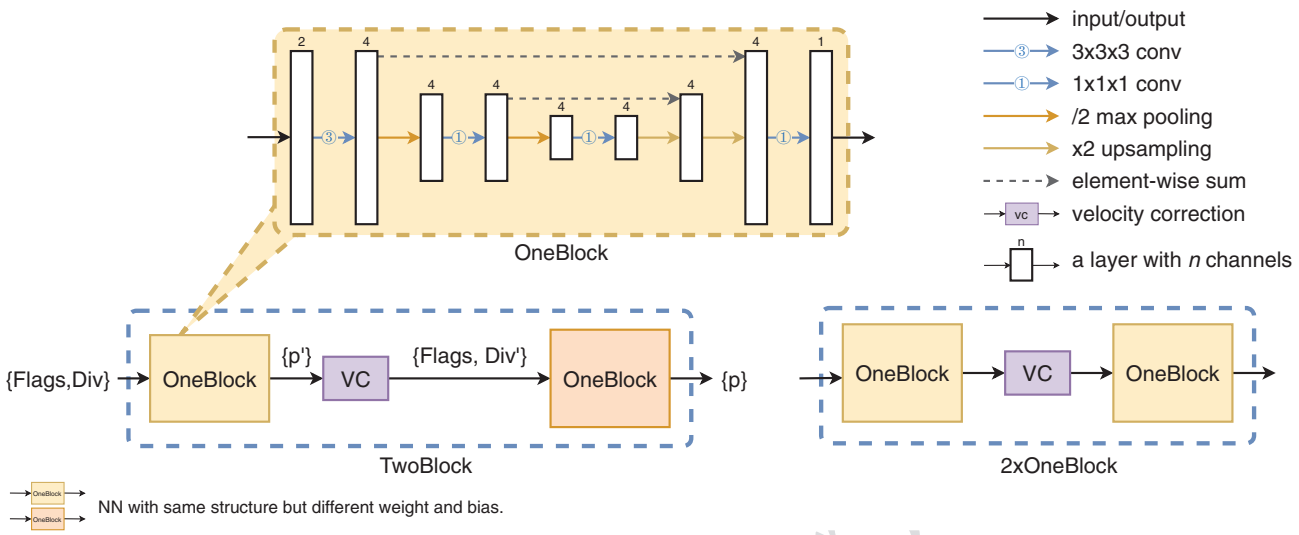


Fig. 2 Network structure of the BlockNN. The BlockNN is composed of multiple OneBlocks (shown in dashed box) and formed as an iterative-like structure. The direct pressure field $p\prime$ output of the hidden OneBlock is used to correct the velocity field and then set up a new divergence input $\nabla \cdot \mathbf{u}\prime$ for the next OneBlock. In a OneBlock, the input fluid quantities are applied with convolutional operations for feature extraction, a max pooling operation for setting up a coarse grid and a nearest upsampling operation to restore to the original resolution. The element-wise summation between convolutional layers aligns fluid messages of different grid sizes. The 2× OneBlock and the TwoBlock have exactly the same structure. But in the TwoBlock, each OneBlocks has different trainable weights and biases, while 2× OneBlock shares those parameters

pressure data as ground truth and train our model in supervised mode, the training result failed to reach a comparable level either. A possible explanation might be that the pressure guiding causes the training to fit into some fiddly cases and accordingly cause the network to be overfitted. As a result, we train our unsupervised model with obstacles flags and divergence as input.

4.2 OneBlock for feature extraction and alignment

We come up with a feature extraction and alignment structure and refer to it as the OneBlock structure, as shown in Fig. 2. In a OneBlock, we first extract features from the input fluid field quantities in the original grid scales x, and gradually compact these features into coarser grids with size $x/2$ and $x/4$ separately. Then, the extracted features in different grid scales are upsampled and aligned to the original scale. As the fluid field is composed of both large- and small-scale fluid motion, the feature extraction applied in different grid scales gives full consideration to the fluid states and thus ensures the output accuracy. We will elaborate on the details of the feature extraction and alignment in the following context.

The input to our OneBlock is a fluid field in x^3 grid sizes with two channels. The feature extraction is first applied to the input fluid field, using kernels with receptive field size $3 \times 3 \times 3$. The number of channels in all hidden layers in our OneBlock is 4. The extracted features in the original grid size x^3 are then filtered with a max pooling operation and form a set of coarser features with size $(x/2)^3$. The coarser features are fused with themselves by applying a $1 \times 1 \times 1$ convolution operation. The max pooling operation helps to select the most significant feature components in the local fluid area. The fusion operation reorganizes the features in different channels and sets up a series of new features. With a similar process, the coarse features in scale $x/2$ are used to extract a set of coarser features in scale $x/4$. These $x/4$ features are fused as well. Now we obtain features fused in different scales, and before we utilize these features to output a pressure solution, we need to make alignments among them.

The fused features in the coarse scale $x/4$ are first upsampled with a nearest neighbor method into scale $x/2$. The upsampled features and the previous fused features in scale $x/2$ are integrated together with an element-wise summation. Similarly, we can apply the alignment to the features in the grid scales of $x/2$ and x. Since the coarser features tend to learn the fluid distribution with larger receptive field size and the features in the original scale maintain the details in small scale, this alignment operation makes full use of the advantages of multi-resolution information from the input. Finally, the aligned features are fused together with a $1 \times 1 \times 1$ convolutional layer to construct the final output, i.e., the pressure field solution.

In the OneBlock, the convolutional operation applied in the coarse grid greatly reduces the computational cost. The alignment operation is constructed as an identity mapping and element-wise summation, requiring no additional train-

Journal: 371 MS: 2025 TYPESET DISK LE CP Disp.:2020/11/30 Pages: 10 Layout: Large

able parameters or complex computation. During training, our network learns to adapt the parameters to balance the feature information among the coarse grids and the dense girds.

4.3 BlockNN for iterative solution

As shown in Fig. 2, we stack two OneBlocks in our network, making the BlockNN structure iterative like. We can stack the OneBlocks in two ways, just as in the 2× OneBlock and the TwoBlock. The OneBlocks in the 2× OneBlock share identical weights and biases, and obviously in this way, we can easily form 3× OneBlock, 4× OneBlock, etc. structures. The TwoBlock has two independent OneBlocks and their parameters are trained separately. As can be seen, the BlockNN has flexible and versatile structures. We can also stack a OneBlock and a TwoBlock together and name it as One_TwoBlock. We evaluate BlockNNs' performance in Table 2 in Sect. 5. With the consideration of accuracy and computation cost, we choose the TwoBlock structure as our final model structure.

The design of this iterative network structure is a tradeoff between accuracy and efficiency. To some extent, a convolutional network just with a deeper structure is capable of achieving better accuracy, but that accuracy gets saturated quickly and computational burden increases when the network depth increases. The BlockNN structure is similar to the iterative operation in traditional PCG and Jacobi solution [1]. This stacked structure turns out to improve the output accuracy iteratively and they can be dynamically adapted to balance the outcome accuracy and efficiency.

Another advantage of this iterative placement is to ease the training burden. During training, the network is actually learning to fit into a desired function mapping. As the direct mapping from velocity divergence to pressure in a one-time calculation is not an easy task in traditional solutions, this direct mapping is not easy for a network to learn. Compared to the network formulation that learns a direct mapping, our iterative network structure is more consistent with the process of mathematical iterative solutions. Our BlockNN also avoids the network from a very deep structure. As our training procedure is unsupervised, a deeper network is harder to be trained. Therefore, the iterative BlockNN setting helps to train the network.

4.4 Training procedure

The diversity of the training dataset is crucial to the generalization ability of the neural network. In order to collect samples that can represent the status of the real world fluid as much as possible, we initialize fluid scenes with different random settings and add random perturbations during the simulation. We adopt the data creation approach proposed in [28].

For the initialization of each scene, we randomly place several solid objects in the fluid field and generate a random fluid velocity field. The 3D solid objects are selected from the NTU 3D Model Database [2], and they are randomly scaled, translated and rotated. We ensure the randomly generated velocity to be divergence free by utilizing the wavelet turbulent noise [9] with random noise parameters.

In order to add perturbations during the simulation, we place inflow sources with random speeds and sizes at random positions and time durations. We initialize 200 and 20 scenes for training and testing, respectively. During the simulation, we advect the fluid field, apply forces and solve pressure with the mantaflow software framework [26]. Each scene will be simulated for 256 steps and the fluid data are collected every eight steps. Finally, we generate 6000 and 600 samples for training and testing, respectively.

When training the TwoBlock in an end-to-end way, the measured loss fluctuates in the first few epochs and gradually converges (Fig. 3). In order to acquire a stable training for the TwoBlock, we adopt a pretrained scheme. We first set up and train the OneBlock network. Then, we initialize the parameters of each OneBlock in our TwoBlock model and fine-tune its parameters. This training strategy helps the network to converge more quickly.

5 Experiments

The main purpose of our pressure solver model is to achieve acceptable accuracy with less computation cost. In our experiments, we test the effect and efficiency of our model and compare it with previous works. We also prove the importance of the alignment operation and iterative structure in our model. We train and evaluate our model on the machine with an NVIDIA 1080Ti and an Intel Xeon E5-2690 CPU.

As our desired outcome is a divergence-free velocity field, we measure the $\mathcal{L}2$ Norm of the output velocity divergence as our evaluation criterion, which is described in Eq. 6:

$$\text{error} = \|\nabla \cdot \mathbf{u}\|^2. \tag{6}$$

The error measured during training is plotted in Fig. 3.

5.1 Speedup

The BlockNN structure is very compact and it requires less computation cost in hidden layers. The training time of a OneBlock model is less than 1 h, and the fine-tuning of a TwoBlock model with pretrained block features converges even faster, as shown in Fig. 3.

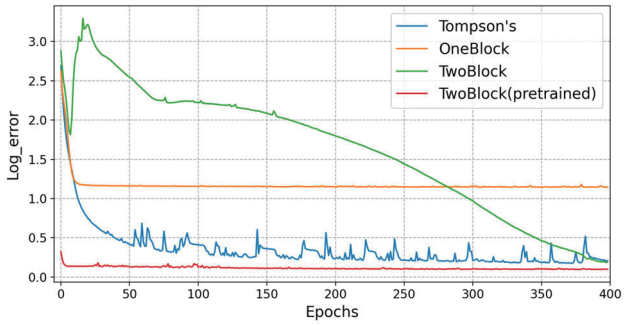


Fig. 3 Error measured during training. The OneBlock model can be trained quickly and steadily, but its performance is not impressive enough. The TwoBlock model struggles to converge with an oscillated curve. The TwoBlock model with pretrained OneBlock converges to a good point quickly

Table 1 Speed and accuracy comparison of the TwoBlock and previous approach

Method	Error	GFlops	Runtime (ms)		
			64^3	128^3	320^3
TwoBlock	3.9178e–5	2.02	2.315	17.25	275.3
Tompson's [28]	4.61046e–5	16.82	3.366	26.29	428.5

We measure the error of our TwoBlock model and compare it with the CNN method in [28], as shown in Table 1. The error is tested using the collected testing data in 64^3 resolution. We also compare the computational cost in 64^3 resolution, and plot the runtime in resolution 64^3, 128^3 and 320^3, respectively. Our BlockNN structure achieves better outcome accuracy while only taking an eighth computation cost. When the size of the input fluid grows, our model gradually shows its strengths in speed.

5.2 Flexibility and stability

Using multiple OneBlocks in the network is like the iteration in traditional solution and this strategy improves the outcome accuracy as a proper amount of the OneBlocks are used. The BlockNN structure can also be edited dynamically to fit into different fluid scenarios. In the following context, we will evaluate our BlockNN in different fluid settings(with or without geometry) and different input sizes of fluid scenes. We also plot our model's performance over each time step and show its stability.

We show how the number and setting of the OneBlocks affect the outcome accuracy in Table 2. The results in Table 2 indicate that the interactive placement of OneBlocks helps to improve the output performance. And fine-tuning the parameters in each OneBlock(as in the TwoBlock setting), is beneficial for better accuracy.

Table 2 Performance of different BlockNN settings

Method	Tompson's [28]	OneBlock	2×OneBlock
Error	4.61046e–5	0.00043626	6.8730e–5
Method	3× OneBlock	TwoBlock	One_TwoBlock
Error	2.8278e–5	3.9178e−5	1.4504e−5

With more OneBlock structure used, the model turns out to perform better. The fine-tuning of the TwoBlock model also increases the output performance

Table 3 The adaptability of the BlockNN to fit into variable fluid settings

Error: e–5	64^3		128^3	
	Fluid	Fluid + solid	Fluid	Fluid + solid
Tompson's [28]	3.0813	4.6105	3.6431	3.7844
TwoBlock	3.6688	3.9178	4.4544	4.0180
One_TwoBlock	1.5655	1.4504	2.4975	2.1582

The BlockNN models trained with 64^3 fluid data work well in 128^3. The model accuracy has only subtle differences between the fluid–solid and pure-fluid systems

For different fluid training tasks, i.e., a fluid field with different grid sizes and unseen geometric shapes, our network is flexible to adapt its structure to meet the new demand for different fluid resolutions. It's easy and flexible to adapt the number of the OneBlocks to meet the demand. For example, when we find the result of the TwoBlock setting not satisfactory, we might simply rearrange the network setting, to become a OneBlock and a TwoBlock stacking together. This new setting requires no extra training and can be adapted dynamically. We show our results in Table 3.

The stability of velocity divergence during fluid simulation is important. We use about 600 scenes with different random initial settings and measure our output divergence error over 128 time steps in Fig. 4. The BlockNN model shows less error than the Jacobi solver.

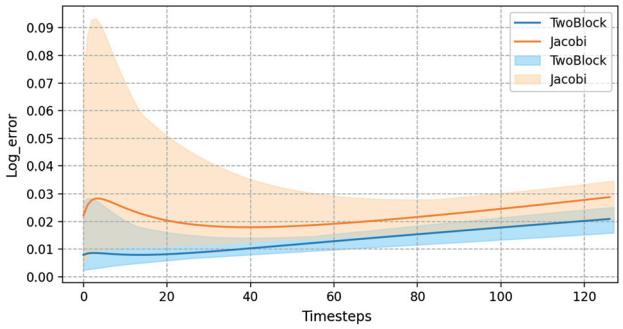
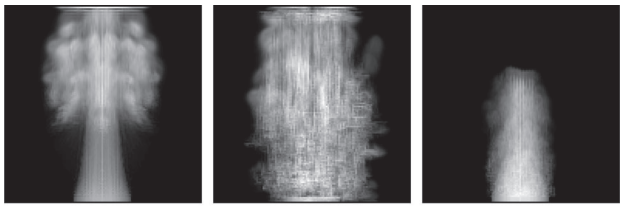


Fig. 4 Error measured during the simulation. The curve in this figure plots the mean error in 600 samples, and the shadowed area represents the [min, max] range among these scenes

Journal: 371 MS: 2025 TYPESET DISK LE CP Disp.:2020/11/30 Pages: 10 Layout: Large



(a) TwoBlock (b) UpSample (c) NoAlignment

Fig. 5 Simulation results with different pressure calculation schemes. Left our proposed TwoBlock model. Middle solve the pressure in the coarse grid and upsample the pressure with artificial interpolation. Right a similar structure to TwoBlock but remove the alignment connection inside the block

5.3 Analysis of alignment

Using the alignment by applying the skip connection achieves significant improvement. Figure 5 shows the simulation comparison.

Figure 5a shows the result with our proposed TwoBlock. The simulated field is natural and realistic.

The result in Fig. 5b is obtained by solving the pressure field with coarse features only and upsampling the pressure field with artificial methods, i.e., linear interpolation. Such pressure solution cannot correctly ensure the fluid velocity to be incompressible and would even disturb the original field, making it a chaotic and unstable condition. This result demonstrates that simply interpolating the pressure field from a coarse grid is not feasible.

Figure 5c shows the result which solves the pressure field with a convolutional neural network but without the alignment strategy. Such a network fails to learn the actual distribution of the fluid field but results in smoothing out the fluid field. The coarse grid features actually lose high-resolution information and the pressure field solution needs the high-resolution features to localize precisely.

5.4 Visual results

In Fig. 6, we compare the simulated visual results of our methods with others. In "Buoyancy" and "Architecture," we add buoyancy to the smoke and make it rise over time. In "Gravity," we pour smoke from the top and allow it to drop under gravity. The previous method [28] tends to smooth out some high-frequency features and cause slower flow motion along complex obstacle surfaces (as shown in the "Gravity" case). Our approach aligns different levels of features and

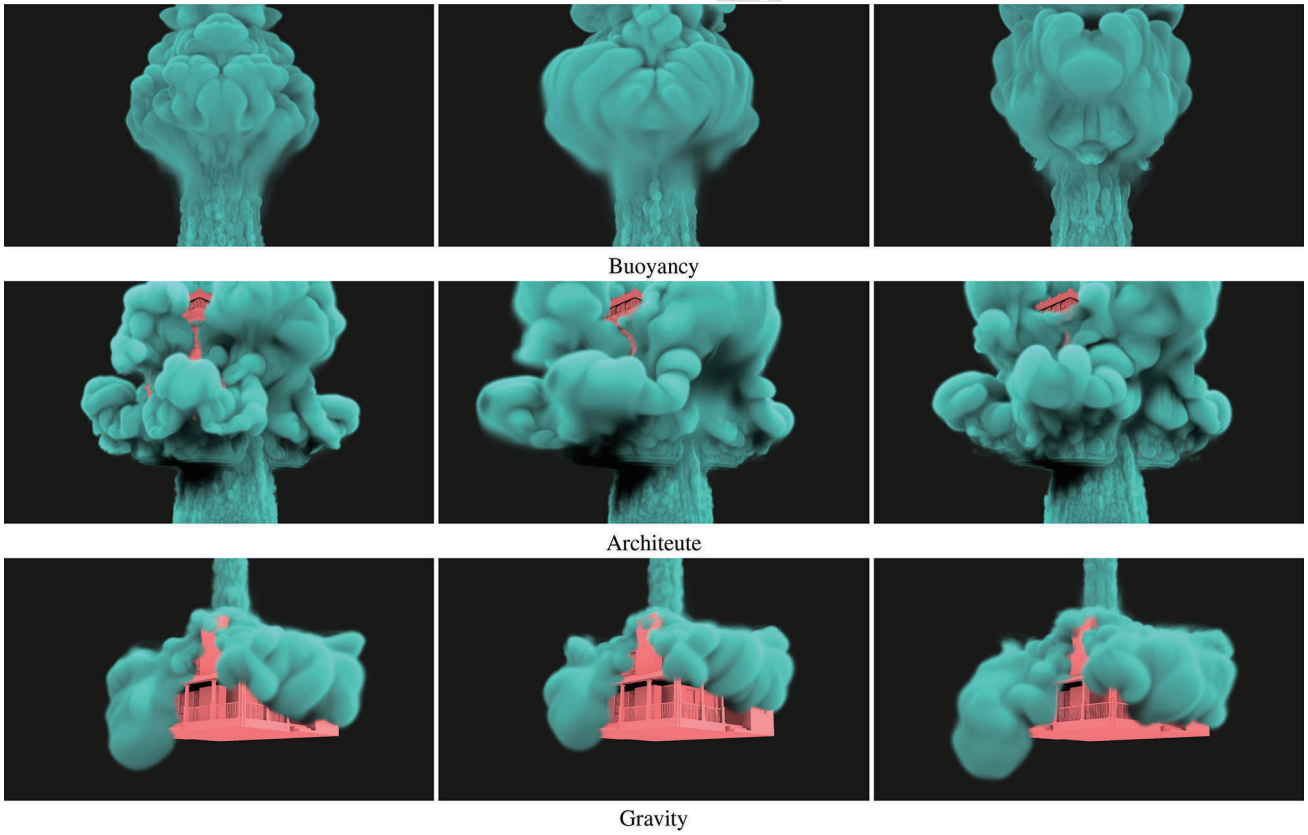


Fig. 6 Comparisons of visual effects with different scene settings. The first column shows the results of our proposed method. The second column shows the results using Tompson's method [28]. The third column shows the results with the Jacobi method

Journal: 371 MS: 2025 TYPESET DISK LE CP Disp.:2020/11/30 Pages: 10 Layout: Large

preserves better detailed structures as well as the macroscopic flow speed.

6 Discussions and conclusions

We present a flexible network structure to solve the pressure field in fluid simulation by learning fluid features in coarser resolution and make alignment with the fused features. With our network architecture, the time and computational cost can be saved in training as well as runtime. Our solution requires much less computation cost compared with similar convolutional methods [28] and achieves better simulated results. We can easily adapt our network structure to meet more complicated tasks.

The drawback of our network is that, by far, we only tackle with simple flags types, i.e., obstacles and fluid. As we use flags types as input to the convolutional layer, it's difficult for the network to distinguish more complicated types. Currently, we only involve smoke simulations in our work. A typical simulation of fluid, like water, might be assisted with particles [1] and involve an additional boundary condition between fluid and air. In order to apply our work to other types of incompressible fluid simulations, we need additional efforts. One possible future solution is to train a classifier to separate the pressure solution tasks with different flag types.

As for future work, our work is beneficial for physically faithful fluid simulation in high resolution. With our presented model, fast prediction of fluid motion can be achieved and typical applications include providing a surrogate model for reinforcement research [16], which needs a large amount of interaction data during training, and helping hydraulic engineering tasks which require quick previews of their designs. The instant result obtained by our BlockNN model can be additionally fine-tuned with traditional methods, i.e., PCG, contributing to faster convergence.

Funding ~~This study was funded by National Natural Science Foundation of China (61602265).~~

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

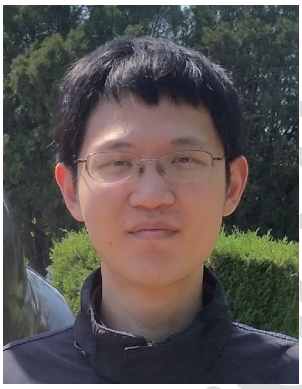
References

1. Bridson, R.: Fluid Simulation for Computer Graphics. AK Peters/CRC Press, Boca Raton (2015)
2. Chen, D.Y., Tian, X.P., Shen, Y.T., Ouhyoung, M.: On visual similarity based 3d model retrieval. Comput. Graph. Forum **22**, 223–232 (2003)
3. Cornelis, J., Bender, J., Gissler, C., Ihmsen, M., Teschner, M.: An optimized source term formulation for incompressible SPH. Vis. Comput. **35**(4), 579–590 (2019)
4. Dupont, T.F., Liu, Y.: Back and forth error compensation and correction methods for removing errors induced by uneven gradients of the level set function. J. Comput. Phys. **190**(1), 311–324 (2003)
5. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
6. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1125–1134 (2017)
7. Jeong, S., Solenthaler, B., Pollefeys, M., Gross, M., et al.: Data-driven fluid simulations using regression forests. ACM Trans. Graph. (TOG) **34**(6), 199 (2015)
8. Kim, B., Azevedo, V.C., Thuerey, N., Kim, T., Gross, M., Solenthaler, B.: Deep fluids: a generative network for parameterized fluid simulations. Comput. Graph. Forum **38**, 59–70 (2019)
9. Kim, T., Thürey, N., James, D., Gross, M.: Wavelet turbulence for fluid simulation. ACM Trans. Graph. (TOG) **27**, 20 (2008)
10. Lagaris, I.E., Likas, A., Fotiadis, D.I.: Artificial neural networks for solving ordinary and partial differential equations. IEEE Trans. Neural Netw. **9**(5), 987–1000 (1998)
11. Lagaris, I.E., Likas, A.C., Papageorgiou, D.G.: Neural-network methods for boundary value problems with irregular boundaries. IEEE Trans. Neural Netw. **11**(5), 1041–1049 (2000)
12. LeCun, Y., Bengio, Y., et al.: Convolutional networks for images, speech, and time series. Handb Brain Theory Neural Netw **3361**(10), 1995 (1995)
13. Lim, B., Son, S., Kim, H., Nah, S., Mu Lee, K.: Enhanced deep residual networks for single image super-resolution. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops, pp. 136–144 (2017)
14. Lim, J.G., Kim, B.J., Hong, J.M.: Water simulation using a responsive surface tracking for flow-type changes. Vis. Comput. **32**(5), 641–651 (2016)
15. Losasso, F., Gibou, F., Fedkiw, R.: Simulating water and smoke with an octree data structure. In: ACM SIGGRAPH 2004 Papers, pp. 457–462 (2004)
16. Ma, P., Tian, Y., Pan, Z., Ren, B., Manocha, D.: Fluid directed rigid body control using deep reinforcement learning. ACM Trans. Graph. (TOG) **37**(4), 96 (2018)
17. McAdams, A., Sifakis, E., Teran, J.: A parallel multigrid poisson solver for fluids simulation on large grids. In: Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 65–74. Eurographics Association (2010)
18. Monaghan, J.J.: Smoothed particle hydrodynamics. Ann. Rev. Astron. Astrophys. **30**(1), 543–574 (1992)
19. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. arXiv preprint arXiv:1711.10561 (2017)
20. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention, pp. 234–241. Springer (2015)
21. Selle, A., Fedkiw, R., Kim, B., Liu, Y., Rossignac, J.: An unconditionally stable maccormack method. J. Sci. Comput. **35**(2–3), 350–371 (2008)
22. Sirignano, J., Spiliopoulos, K.: Dgm: a deep learning algorithm for solving partial differential equations. J. Comput. Phys. **375**, 1339–1364 (2018)
23. Stam, J.: Stable fluids. Siggraph **99**, 121–128 (1999)
24. Szegedy, C., Toshev, A., Erhan, D.: Deep neural networks for object detection. In: Advances in neural information processing systems, pp. 2553–2561 (2013)

Journal: 371 MS: 2025 ☐ TYPESET ☐ DISK ☐ LE ☐ CP Disp.:2020/11/30 Pages: 10 Layout: Large

25. Temam, R., Chorin, A.: Navier Stokes equations: theory and numerical analysis. Handb Numer Anal **6**, 503–689 (1978)
26. Thuerey, N., Pfaff, T.: MantaFlow (2018). http://mantaflow.com
27. Thuerey, N., Weißenow, K., Prantl, L., Hu, X.: Deep learning methods for Reynolds-averaged Navier-Stokes simulations of airfoil flows. AIAA J. **58**, 1–12 (2019)
28. 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, vol. 70, pp. 3424–3433. JMLR. org (2017)
29. Wiewel, S., Becher, M., Thuerey, N.: Latent Space Physics: Towards Learning the Temporal Evolution of Fluid Flow. Computer Graphics Forum, vol. 38, pp. 71–82. Wiley Online Library, Amsterdam (2019)
30. Xie, Y., Franz, E., Chu, M., Thuerey, N.: tempogan: a temporally coherent, volumetric gan for super-resolution fluid flow. ACM Trans. Graph. (TOG) **37**(4), 95 (2018)
31. Yang, C., Yang, X., Xiao, X.: Data-driven projection method in fluid simulation. Comput. Anim. Virtual Worlds **27**(3–4), 415–424 (2016)

Rong-Jie Xu is currently a Master student in College of Computer Science, Nankai University, under the supervision of A.P. Bo Ren. Her research interests include physically based simulation and rendering.



Bo Ren received his Ph.D. degree in computer science from Tsinghua University, Beijing, in 2015. He is currently an associate professor in the College of Computer Science, Nankai University, Tianjin. His research interests include physically based simulation and rendering, scene geometry reconstruction and analysis. His recent research focuses on multi-fluid and multi-phase simulations in computer graphics.