Neural Physical Simulation with Multi-Resolution Hash Grid Encoding

Haoxiang Wang[1], **Tao Yu**[1†], **Tianwei Yang**[1], **Hui Qiao**[1, 2†], **Qionghai Dai**[1]

[1] Department of Automation & BNRist, Tsinghua University
[2] Shanghai Artificial Intelligence Laboratory

Abstract

We explore the generalization of the implicit representation in the physical simulation task. Traditional time-dependent partial differential equations (PDEs) solvers for physical simulation often adopt the grid or mesh for spatial discretization, which is memory-consuming for high resolution and lack of adaptivity. Many implicit representations like local extreme machine or Siren are proposed but they are still too compact to suffer from limited accuracy in handling local details and a long time of convergence. We contribute a neural simulation framework based on multi-resolution hash grid representation to introduce hierarchical consideration of global and local information, simultaneously. Furthermore, we propose two key strategies: 1) a numerical gradient method for computing high-order derivatives with boundary conditions; 2) a range analysis sample method for fast neural geometry boundary sampling with dynamic topologies. Our method shows higher accuracy and strong flexibility for various simulation problems: e.g., large elastic deformations, complex fluid dynamics, and multi-scale phenomena which remain challenging for existing neural physical solvers.

Introduction

Physics simulations involve continuous spatiotemporal systems governed by partial differential equations. In traditional simulations, time and space need to be discretized due to the lack of an effective continuous representation. Mesh/grid is often a common topic in classical simulation. However, several issues arise for these mesh-based approaches: first of all, they cannot simulate in a continuous spatiotemporal domain owing to the limited resolution of discretization. Moreover, the introduction of mesh or grid through spatial discretization also hinders the adaptability to dynamic geometries. Methods like re-meshing or adaptive grids to handle dynamic geometries are difficult to implement and require a large memory usage. Some current advanced data-driven simulation methods (Morimoto et al. 2021; Pfaff et al. 2020) try to fill these gaps with features on the discretized structures, but they are often not easy to generalize and do not ensure the physical equations.

As a promising alternative, mesh-free simulations are favored over mesh-based methods. In recent years, the development of implicit representation has opened up many possibilities for mesh-free simulations, with various implicit representations (Park et al. 2019; Mildenhall et al. 2021; Mescheder et al. 2019; Chen and Zhang 2019; Sitzmann et al. 2020) proposed for efficient encoding and consideration of spatiotemporal characteristics. However, leveraging current implicit representations for continuous simulation remains challenging because:

1. the existing implicit representations based on neural network are too compact, characterized by high correlations of values corresponding to neighboring sample points, leading to low representation efficiency and slow convergence during training;
2. the other representations based on kernel functions suffer from high computational cost and heavy memory storage burden;
3. most of the representations are struggling to handle dynamic geometries effectively due to inefficient sampling.

To tackle these challenges, we propose a general and efficient physics simulation method based on the implicit neural representation. This method enhances the performance and accuracy of simulations, while also demonstrating the potential to overcome simulation challenges under dynamic geometry changes, which is the case that traditional methods struggle with. Specifically, to address the compactness issue in the existing approaches, we adopt the multi-resolution hash grid as a positional input encoder to facilitate representation from large to extremely small levels with limited memory and automatically focus on the relevant details. The utilization of the hash grid for simulation, however, poses various challenges, such as the locality issue when calculating analytical gradient w.r.t position. Therefore, we propose a finite-difference-based numerical gradient computation method to enhance the joint optimization of multiple hash entries. Moreover, to resolve the issue of topological variation under dynamic geometry changes, we propose a range-analysis-based fast adaptive object boundary sampling method to improve the sampling efficiency. In summary, we make the following contributions:

- A novel mesh-free neural simulation framework based on multi-resolution hash grid encoding, with strong flexibility to support different physical simulation tasks.

†: corresponding author

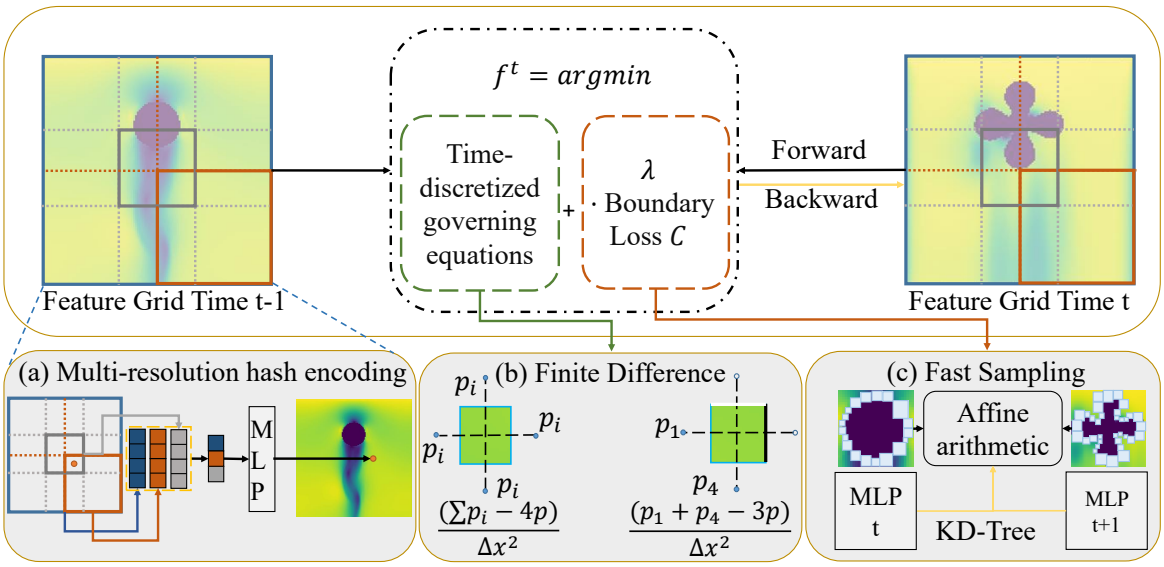

Figure 1: The paradigm of our proposed method.

- A numerical gradient computation method based on Finite Difference (FD) with the ability to handle boundary conditions.
- A range-analysis-based object boundary sampling strategy to improve simulation efficiency under dynamic changing geometries.

Experiments and results under different simulation tasks demonstrate the effectiveness, efficiency, and potential of our method. Our approach takes a significant stride in addressing fluid-solid interaction (FSI) challenges, particularly in scenarios involving complex geometries, which is important for practical applications like differential design (e.g. Aquatic swimmer design) and visual effects creation (hair/water simulation).

Related Work

Implicit Neural Representation Implicit neural representation leverages neural networks to parameterize spatial dependent functions (Mescheder et al. 2019; Chen and Zhang 2019). The method is first highly valued in the 3D signed distance field representation (Park et al. 2019). The neural network's representation capability can better benefit the extraction of spatial information. The idea is carried and promoted in many 3D tasks, like 3D reconstruction (Wang et al. 2021), neural rendering (Mildenhall et al. 2021), and geometry processing (Yang et al. 2021). In the process of utilization, people also found it inefficient and hard to converge in training the simple MLP representation directly with coordinate inputs (Michalkiewicz et al. 2019). Glances are attracted by efficient spatial positional encoding. Siren as a pioneer of positional encoding is proposed (Sitzmann et al. 2020) and makes much progress in performances. However, the training and convergence of Siren is still inefficient since the representation is too compact and the training process without adequate designs will lead to an undesirable outcome. Some new approaches are proposed to introduce more local characteristics. Tri-plane representation (Chen et al. 2022) and multi-resolution hash grid (Müller et al. 2022) are typical among the literature. The former assumes the spatial domain is sparse and takes advantage of sparse tensor decomposition and the latter utilizes a multi-resolution hash grid to facilitate representation from large to extremely small levels with limited memory. For the consideration of the efficiency, performance, and generalization capability, we regard a multi-resolution hash grid as the first choice.

Neural Physical Simulation Neural physical simulation can be divided into two main streams. The first one is the data-driven simulation. This type of method often aim at solving simulation problems based more on data but less on the governing equation. They often adopt the training data from the classical solvers or the real world observation to make the neural network learn the physical rules and generalize it to other scenes. Some convolution network approaches for fluid (Morimoto et al. 2021), graph neural networks for meshes (Pfaff et al. 2020), or other designed networks (Lu, Jin, and Karniadakis 2019) show higher efficiency than the classical solvers. Neural operator approach (Li et al. 2020) makes full use of the Fourier layer and becomes an important milestone for the type of methods. However, these methods can typically not be well generalized to the other initial/boundary conditions, material parameters, or geometries. The training data acquisition and time-consuming training processes also block their wide applications. Another stream of the line is to embed the governing equations into the network. For this type, one representative direction we should mention is Physical-Inform Neural Network (PINN) (Raissi, Perdikaris, and Karniadakis 2019). The method designs the physical loss term according to the governing equation and the neural network is trained to extract the features of the spatiotemporal correlation and fit the target field. However, the ways to directly force the neural network to fit all the physical rules make the training pro-

cess difficult. The training will cost a very long time and often can not achieve the required accuracy. To fulfill this issue, the implicit neural representation (Zehnder et al. 2021; Chen et al. 2023) is also introduced to better describe the spatiotemporal dependencies and reduce the burden of network training. But the methods mainly use simple MLP or Siren positional encoding, which can not combine the spatial information for both global and local levels and bring challenges for sampling and training. Hence, in this work, we construct a novel framework that embeds a multi-resolution hash grid into the neural physical simulation to better improve the efficiency and performance.

Method

In this part, we present details of our proposed framework for solving the physical simulation problem. We show our whole pipeline in Fig. 1. We propose the representation and calculation process of our method. For each time step, we utilize a feature grid to extract the spatial information with time-discretized governing equations terms and boundary loss according to the time-dependent PDEs for our simulation. The feature grid is designed as a multi-level hash grid. For better characterizing the time-discretized governing equations and boundary terms, our framework provides two important components: 1) finite difference method for high order differential operator calculation for the time-discretized governing equations terms; 2) fast sampling for dynamic geometry change of the boundary.

Multi-resolution Hash Grid for Simulation

Most of the simulation problems can be denoted as finding a suitable temporal-spatial vector field: $f(x,t)$, where $x \in \Omega$ and Ω is the spatial domain, such as the deformation field in elastic problem or the velocity and pressure field in fluid problem. We parameterize f with weights θ and derive $f(x,t,\theta)$. Our framework adopts time discretization and spatial implicit representation to solve f. Specifically, $t \in [T]$ and f can be represented as several $f(x,\theta^t)$. We rewrite it as $f^t(x,\theta)$ for a fixed spatial domain. With the initial conditions and the governing equation, we need to determine the accurate value for $f^t(x,\theta)$ time by time.

Regarding the spatial implicit representation in our framework, f is designed as an encoder-decoder neural network. A neural encoder is utilized for spatial feature representation and an MLP is served as a decoder to map the spatial feature field to the spatial vector field.

Multi-resolution hash grid representation The encoder in our method is a multi-resolution hash grid. It is a milestone method in the neural rendering (Müller et al. 2022). The representation enjoys high spatial efficiency. For each point in the domain, we first map it to $[0,1]^d$, where d is the spatial dimension. Then we divide the whole space into an $L-$level multi-resolution grid to extract the features. The hierarchies will carry features for different spatial scales. For a single point, the point features can be calculated by interpolating each single level's grid vertices and wrapping them together for both global and local relationships. The feature vectors at the vertices of a grid are stored in a hash table and we can find it through a hash mapping h for the T feature vectors as $h(x) = \left(\oplus_{i=1}^{d} x_i \pi_i\right) \bmod T$, where $\oplus$ denotes the bit-wise XOR operation and π_i are unique, large prime numbers. We know when the finest resolution becomes extremely large, the corresponding vertices number is also huge. But in this representation, the hash table size is fixed and the subsequent hash collision will be just solved by the multi-resolution design since the probability that one point collides with the other on every level is extremely small. In other words, the representation supports remarkably high spatial resolution with limited memory.

After constructing our encoder above, we derive $y = enc(x;\xi)$ as features and input it to a fully connected network decoder $m(y;\Phi)$ to resolve the remaining conflicts and derive the target vector field.

Block time-marching scheme After obtaining the spatial representation, we construct a block time-marching scheme for simulation. The idea is well adopted in (Dong and Li 2021; Chen et al. 2023). More specifically, for most of simulation problems, the requisites of the vector field in each discrete time $f^t(x,\theta)$ can be divided into time-discretized governing equations terms $\mathcal{I}$ and boundary conditions terms $\mathcal{C}$, such as least square terms constructed from forward/backward Euler time discretization schemes or the variational form in the elastic problem. Then the simulation problem can be formulated as:

$$\begin{aligned} f^{t+1} = argmin_{f^{t+1}} & \sum_{x\in\Omega} \mathcal{I}(\Delta t, \{f^s(x)\}_{s=0}^{t+1}, \{\nabla f^s(x)\}_{s=0}^{t+1}, \\ & \{\nabla^2 f^s(x)\}_{s=0}^{t+1}) + \lambda \sum_{x\in\partial\Omega} \mathcal{C}(\Delta t, \{f^s(x)\}_{s=0}^{t+1}, \\ & \{\nabla f^s(x)\}_{s=0}^{t+1}, \{\nabla^2 f^s(x)\}_{s=0}^{t+1}), \end{aligned} \tag{1}$$

For each timestep, we can construct a sample set and utilize the samples to derive the approximate optimization results of f^t parameterized by the above ξ and Φ. Note that our multi-resolution hash grid does not conduct explicit spatial discretization, hence we also need to propose an efficient sampling and calculation process for the high accuracy and fast convergence for function f.

To better characterize our framework, we take the Incompressible Euler Fluid Equations as an example in the following part. The governing equations are:

$$\begin{aligned} & \rho_f\left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\cdot\nabla\boldsymbol{u}\right) = -\nabla p, \\ & \nabla\cdot\boldsymbol{u} = 0, \end{aligned} \tag{2}$$

where f is defined as $\boldsymbol{u}$ and p here. $\boldsymbol{u}$ is the fluid velocity field, p is pressure and ρ_f is density. We apply the operator splitting scheme (Chorin 1968) and take $\mathcal{I}$ as three parts (Stam 1999): advection, pressure projection, and correction. Each part will give a fitting $\boldsymbol{u}$ or p. The advection $\mathcal{I}_{adv}$ is constructed according to semi-Lagrangian methods:

$$\mathcal{I}_{adv}^{t+1} = \|\boldsymbol{u}_{adv}^{t+1}(x) - \boldsymbol{u}^t(x - \Delta t \boldsymbol{u}^t(x))\|_2^2. \tag{3}$$

The pressure projection utilizes the fitted velocity $\boldsymbol{u}_{adv}$ for a suitable pressure field:

$$\mathcal{I}_{pro}^{t+1} = \|\frac{\Delta t}{\rho_f}\nabla^2 p^{t+1}(x) - \nabla\cdot\boldsymbol{u}_{adv}^{t+1}(x)\|_2^2. \tag{4}$$

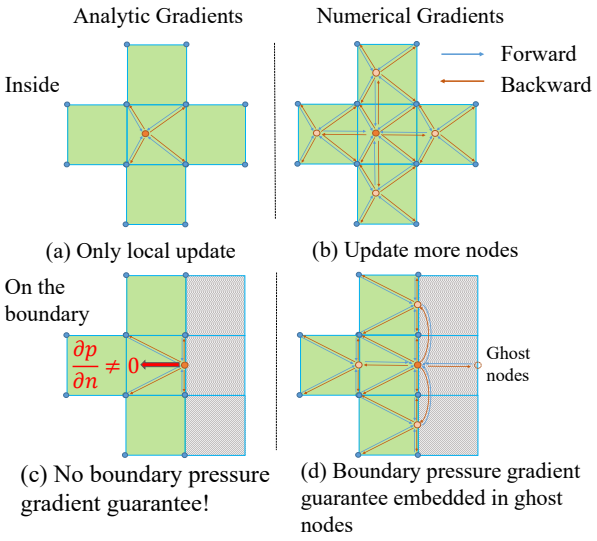

Figure 2: The illustration of our high order differential operator calculation inside/on the boundary

Then we can adopt the correct $\mathcal{I}_{cor}^{t+1}$ for the final velocity results:

$$\mathcal{I}_{cor}^{t+1} = \|\boldsymbol{u}^{t+1}(x) - (\boldsymbol{u}_{adv}^{t+1}(x) - \frac{\Delta t}{\rho_f}\nabla p^{t+1}(x))\|_2^2. \quad (5)$$

The boundary conditions that we mainly consider in this case are the solid wall boundary conditions, which means the normal component of the velocity on the boundary should be zero.

Finite Differences Methods for Numerical High-order Differential Operator

With our simulation problem characterization, the forthcoming task involves the computation of the value associated with the high-order differential operator. Since the analytical differential operator w.r.t position in the spatial multi-resolution hash grid encoding will suffer from a locality problem (Li et al. 2023), in this section, we design our simulation embedding Finite Differences Methods to eliminate the problem and guarantee optimization efficiency.

Finite Differences inside. As shown in Eq. 4, we need to calculate the second-order derivative of the pressure $\nabla^2 p = \Delta p$. Then we review the process of the hash grid encoding, each 3D point x_i will be scaled by a certain resolution V_l as $x_{i,l} = x_i \cdot V_l$ and the interpolation χ will be implemented by $\beta = \chi(x_{i,l} - \lfloor x_{i,l} \rfloor)$ as

$$r_l(x_{i,l}) = r_l(\lfloor x_{i,l} \rfloor) \cdot (1-\beta) + r_l(\lceil x_{i,l} \rceil) \cdot \beta, \quad (6)$$

where r_l is the features and $\lfloor x_{i,l} \rfloor$, $\lceil x_{i,l} \rceil$ are rounded grid vertices but not differentiable. Therefore the second-order derivative is calculated:

$$\frac{\partial^2 r_l(x_{i,l})}{\partial x_i^2} = -V_l r_l(\lfloor x_{i,l} \rfloor) - \frac{\partial^2 \chi}{\partial x_{i,l}^2} + V_l r_l(\lceil x_{i,l} \rceil) \frac{\partial^2 \chi}{\partial x_{i,l}^2}. \quad (7)$$

We can observe the derivative of hash encoding is local as shown in Fig. 2 (a). In other words, when x_i moves across grid cell borders, the corresponding hash entries will be different. We need to conduct complex joint optimization with different hash entries to ensure smoothness across the grid of the Δp, which is not always guaranteed. To overcome the locality of the analytical gradient of hash encoding, we mirror the methods in (Stam 1999) and calculate the numerical Laplacian operator with a five-point stencil (2D). That is,

$$\begin{aligned}\Delta f = \frac{1}{\epsilon^2}(&f(x+\epsilon_x) \\ &+ f(x-\epsilon_x) + f(x+\epsilon_y) + f(x-\epsilon_y) - 4f(x)),\end{aligned} \quad (8)$$

where $\epsilon_x = (\epsilon, 0)$ and $\epsilon_y = (0, \epsilon)$ represent a small step on the direction of x, y axis. As Fig. 2 (b) shows, such back-propagating will allow hash entries of multiple grids to receive joint optimization updates simultaneously.

Remark: The interpolation function is also important in the multi-resolution hash grid. The common setting is polynomial and linear functions. Fortunately, our finite difference methods can benefit both methods. For polynomial functions, our method will help reduce the locality as mentioned above. But if we choose linear form to reduce the computation burden, from Eq. 7 we can observe $\frac{\partial^2 r_l(x_i,l)}{\partial x_i^2}$ will be 0, meaning that we can not update the hash grid features. Fortunately, our proposed methods can still help find high-order differential operators approximately to support efficient backpropagation.

Finite Differences on the boundary. An important part in simulation is how to deal with the solid wall boundary.

Prior work (Chen et al. 2023) samples the boundary and just forces the velocity perpendicular to boundaries to be 0. However, it is not correct as shown in Fig. 2 (c). While handling the solid boundary, we also need the pressure gradient conditions to constrain the $\frac{\partial p}{\partial n}$ on the boundary to be 0 (Bridson 2015). Methods relying on an analytic gradient are local and need to additionally sample on the boundary parts and inject this pressure constraint into the loss term. However, if we proceed in this manner, the solver will suffer from inefficiency and hard convergence due to the extra term fitting.

Therefore, we learn from (Fedkiw et al. 1999) in the fluid and construct ghost variables when estimating the high operator on the boundary with finite difference, as shown in Fig. 2 (d). In more detail, for each point near the boundary, we simply conduct the finite difference in the section above. When some points of the stencil (e.g. $x + \epsilon_x$ on the right) are over the boundary, we take that points and change their value ($p(x+\epsilon_x)$) to the value of central of the stencil ($p(x)$) as ghost points (central orange points). Following this action, we will derive different stencils,

$$\Delta f = \frac{1}{\epsilon^2}(f(x-\epsilon_x) + f(x+\epsilon_y) + f(x-\epsilon_y) - 3f(x)), \quad (9)$$

and it automatically satisfies the pressure gradient conditions, providing much efficiency. For the Dirichlet boundary conditions, we can also construct such ghost points to fix the stencil and just set the value of the points to zero.

Fast Neural Geometry Representation Sampling for Dynamic Geometry

As shown in Eq. 1, besides the governing equations term, careful consideration must be given to the boundary conditions term. Hence, the consideration of boundary geometry is essential. One significant drawback of traditional methods is the high dependence on geometry discretization tools, like mesh or grid. It often takes much cost to conduct re-meshing or adaptive grids for an unknown changing domain. Fortunately, our implicit representation method shares a great meshfree property and only needs to process the sampled points, providing a fast and simple pipeline.

In the following part, we consider a fluid problem with the geometry of the dynamic and implicit boundary (unknown in advance). We propose a fast neural geometry representation sampling method to track the velocity perpendicular to the boundary and support the efficient simulation.

Implicit representation for dynamic geometry We utilize the neural sign distance function (SDF) (Park et al. 2019) to represent the geometry. We construct MLP $\omega(x, t, \eta)$ to represent the dynamic sign distance from point x to the boundary surface at time t. This representation enables both continuous shape interpolation to support physical simulation with arbitrary time steps and also a very detailed geometry representation with limited storage, showing great memory efficiency. Since the geometry is unknown in advance, we only train MLP with an initial shape's SDF and the final shape's, corresponding to time 0 and 1. We sample several spatial points $x \in \Omega$ and $t \in \{0, 1\}$ and training with loss with two components. The first is the mean square error of sign distance for time 0 and 1. To smooth the shape change, we also minimizes the thin plate energy, that is

$$\mathcal{L}_{cur} = \|\mathcal{H}_\omega(x, t, \eta)\|_F^2, \tag{10}$$

where $\mathcal{H}$ means Hessian matrix w.r.t x and t and $\|\cdot\|_F$ means Frobenius norm. Then we can train parameters η that embed the implicit dynamic geometry.

Fast sampling for the implicit dynamic geometry We try to sample this implicit MLP for the boundary at every time. A fast way to query the zero-level set of the implicit neural function with no prior knowledge at the interim time point is needed. Hence, for each time step, we try to construct a spatial KD tree for the zero-level set. We divide the spatial domain into the grid coarsely and use range analysis to estimate the min/max SDF value to fast determine whether the grid is in/out of the surface. When we meet the grid that crosses the surface (the min value is smaller than 0 and the max is larger), we can recursively divide the grid node. Then we can just leverage the grid near the surface with a certain KD-tree depth to determine the sample accuracy and uniformly sample in the grid. We utilize the Affine Arithmetic range analysis (Comba and Stol 1993; Sharp and Jacobson 2022) for fast queries of MLP with Affine approximation.

Remark: This method can help us reduce the calculation of the heavy MLP queries. Our KD-tree based method enjoys $O(log(1/e))$ time complexity while the trivial sampling method is $O(1/e^2)$, where e is the tolerance level. The advantages of our constructed KD-Tree will be further accentuated under our repeat sampling demand. Moreover, the sampling methods for implicit dynamic geometry can also serve any other sampling-based simulator to collect spatial points efficiently.

Methods	Error	Time
Siren	4.6e-7	5.33 h
Grid	5.4e-5	1.13 s
Ours	**4.2e-7**	2.30 h

Table 1: Quantitative results for the 1D advection example. Error: mean square error (MSE) averaged by 240 time steps, computed with the ground truth analytical solution. Time: runtime for a total of 240 time steps.

Our method also enables other types of simulation besides the fluid. We consider another two types of simulation problems: Advection Problem and Elastodynamic Problem to evaluate our proposed framework. The details about time-discretized governing equations term $\mathcal{I}$ and boundary $\mathcal{C}$ are shown in the Appendix.

Numerical Studies

In this section, we evaluate efficiency (Tab. 1-4), accuracy (Fig. 3-6, Tab. 1&3, Figure. 1b), and geometric flexibility (Fig. 7&8, Figure. 1a) of our method. The validation is based on the following three problems: advection equation, elastodynamic equation, and incompressible Euler equations.

Baseline: We choose different traditional and neural algorithms for comparison: 1) Implicit Representation with Siren (Chen et al. 2023); 2) Traditional grid method with the same resolution with our finest hash grid's; 3) Original PINN (Raissi, Perdikaris, and Karniadakis 2019); 4) PINN with temporal sub-domains (Krishnapriyan et al. 2021); 5) physics-informed DeepONet (Lu, Jin, and Karniadakis 2019); 6) Separable PINN (Cho et al. 2022). We ignore algorithms that don't enforce any physical laws (Morimoto et al. 2021) or train with a large amount of data (Li et al. 2020).

Adection Equation

We use an initial condition with a central wave as $u^0 = \exp\left(\frac{(x-\mu)^2}{2\sigma^2}\right)$ and the Dirichlet boundary condition that the boundary of the domain should be zero. The ground truth can be calculated analytically. We show quantitative results in Tab. 1. Our method achieves the same accuracy as Siren-INR methods but saves about 40% time. It demonstrates that our algorithm enables better convergence for higher simulation speed. The traditional method with the same finest resolution fails to attain comparable accuracy, underscoring the efficiency of our representation.

Elastodynamic Equation

We also construct two cases for simulation. We first take the elastic tension test. Provided with Dirichlet boundary conditions for the left and right edges of a shape in Fig. 4, we simulate the stable deformed shape with the energy function and Neo-hookean materials (described in Appendix). We show the visualization solution and the error in Fig. 4. The ground truth for all the elastic tests is coming from the high-resolution mesh-based simulation. Our

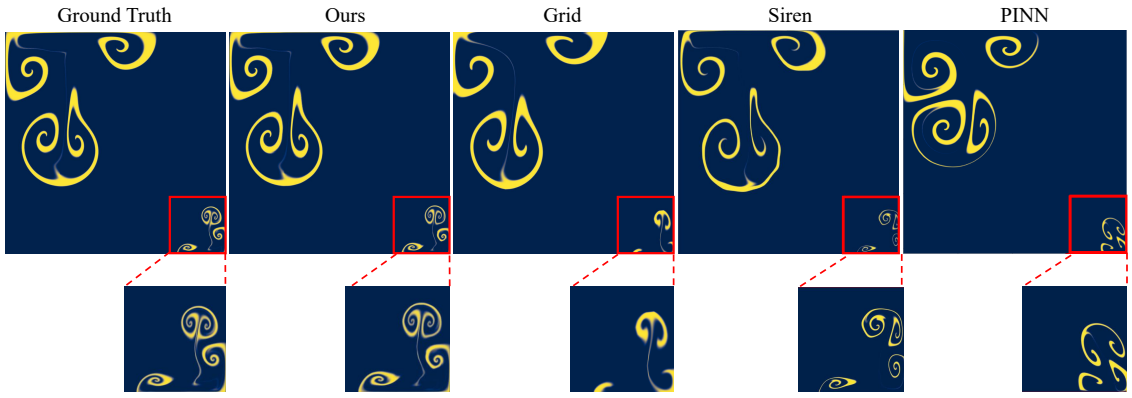

Figure 3: Density field results of two Taylor-Green vortices of different scales. We show the advected density field after 2.5 seconds with different methods. The ground truth is calculated with an extreme high-resolution grid (1024×1024). Our hash grid's finest resolution is 64.

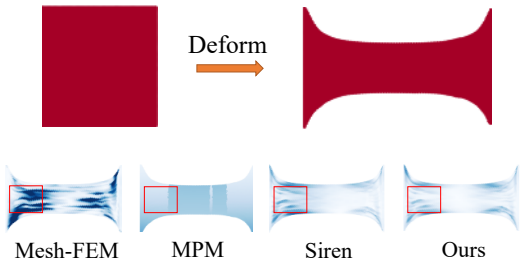

Figure 4: The solution for 2D elastic tension test

method achieves higher speed compared with the Siren-INR methods (about one time in this case) with a similar accuracy guarantee. Compared with the traditional Finite Element Method and Material Point Method (Stomakhin et al. 2013), ours also achieves higher accuracy, showing the effectiveness of our proposed framework.

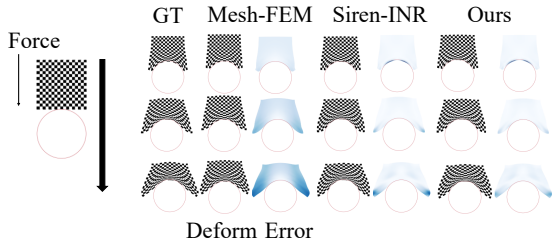

Figure 5: The solution for elastic collision

Incompressible Euler Equation

For the fluid cases, we present three cases to evaluate our framework with the Incompressible Euler equation. We first establish initial conditions to simulate a Taylor-Green flow (Taylor and Green 1937). We show the mean square error of the single Taylor-Green vortex compared with the ground truth in Fig. 6. Siren-INR's error shows an interesting pattern due to the sine function encoding while our results show better performances in the central part. The results demonstrate our algorithm can yield a substantial convergence and better preserve the analytical stable solution.

Methods	Error			Time
	0.3 s	0.5 s	0.7 s	
Siren	8.2e-4	9.3e-4	2.7e-3	18.5 m
Grid	2.4e-2	3.7e-2	4.7e-2	98.2 s
Ours	**1.0e-3**	**1.1e-3**	**2.2e-3**	10.7 m

Table 2: Quantitative results for the 2D collision example. Error: mean square error (MSE), computed with the ground truth high-resolution FEM. Time: runtime for total 10 steps.

Methods	Error	Time
Siren	2.02e-4	10.8 h
Grid	5.05e-3	1 m
PINN	2.15e-2	7.2 h
PINN-sub	3.07e-2	19.2 h
piDeepONet	2.30e-2	9.2 h
SPINN	8.89e-4	9.0h
Ours	**8.84e-5**	5.6 h

Table 3: Quantitative results for the multi-scale Taylor-Green flow. References are shown in the Baseline part. Error: mean square error (MSE) averaged by 50 timesteps and 1024×1024 resolution. Time: the total time for 50 steps.

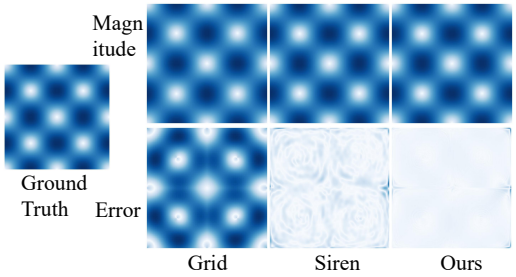

Figure 6: The Mean Square Error for Taylor Green vortex compared with analytic solution

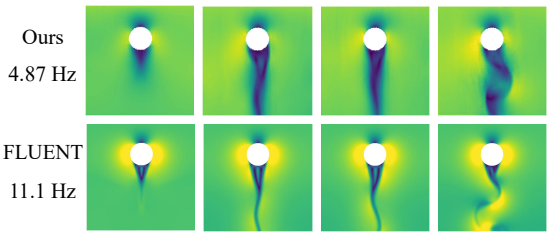

Figure 7: Velocity magnitude for different times of 2D cylinder wall flow.

Next, we expand the single vortex to multiple vortexes with different scale cases. We present the visualization in Fig. 3. We observe that our method effectively preserves multiple vortex structures, including intricate details. The numerical results in Tab. 3 illustrate that our framework saves about 50% of time compared with the Siren-INR and also achieves accuracy improvement about 2 times. In this case, we also compare with PINN, PINN with temporal subdomains (PINN-sub) and piDeepONet. However, these approaches exhibit suboptimal performance. PINN methods can not preserve the details of the vortex field. They make it a heavy burden for simple MLPs to train and fit all the information of the vector field, leading to a low convergence rate and a high possibility of being trapped into the local optimal. Instead, our method makes full use of the information of the multiple scales and fits these features separately, making the optimization easier and faster.

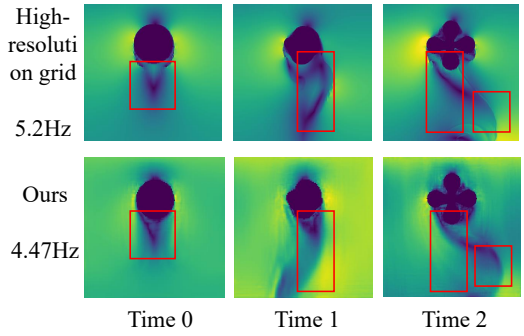

Figure 8: Comparison between the flow with dynamic geometry and ground truth.

Following the vortex cases, we proceed to simulate a 2D cylinder wall flow scene. We set the domain Ω as $[-1, 1]^2$. We set an entrance with an inflow on the top of the domain. The left and right sides provide solid wall boundary conditions. The down exit is an outflow boundary. A circle is placed near the entrance and obstructs the incoming flow. Some interesting phenomena can be observed behind the cylinder wall. We show the comparison of our simulation results and the results from the commercial software FLUENT. We can observe a vortex street phenomenon in both ours and the ground truth results in Fig. 7. It demonstrates that our framework can reproduce the real-world scene to some extent. We also calculate the frequency of the vortex street and our method performs a close magnitude to the ground truth.

	Process	Time (per step)
Total Time	KD Tree Construction	9.7 s
	Advection	63.6 s
	Pressure Solving	297.6 s
	Velocity correction	39.2 s
Sampling Time	Boundary Sampling	1.2 ms
	Field Sampling	0.1 ms
Uniform Sampling Time	Boundary Sampling	50 ms

Table 4: Time cost analysis for the dynamic geometry cases.

Finally, to verify the effectiveness of our proposed sampling strategy, we design an experiment called "dynamic geometry wall flow". That is, we change the circle boundary with a dynamic geometry boundary described with MLP in the section above. We first train a dynamic geometry implicit representation, then employ the sampling strategy proposed. We verify the correctness of our proposed method with high-resolution grid methods as multi-vortex simulation (Fig. 3) as ground truth. In Fig. 8. the close correspondence between our method and the ground truth is evident in the detailed turbulent features. The consistency is further confirmed through the validation of estimated frequency results. The time is also dumped in Tab. 4. It shows the time efficiency of our proposed sampling strategy. We expend only a marginal amount of additional time compared to the cylinder cases, and the amortized sampling cost remains relatively modest.

Discussions and Conclusion

In this work, we propose a neural physical simulation framework based on multi-resolution hash grid encoding. Compared with the traditional solvers, our method can achieve comparable accuracy and high adaptivity. Compared with the existing neural solvers, our method does not need to collect any training data. Our method can also achieve higher convergence speed than other implicit representations. While bringing up the merits, our method has limitations. For example, we still suffer from energy dissipation in the vortex street simulation because we are often trapped by the local optimal if we can not better adjust the coefficient of the boundary condition. It will lead to an inaccuracy for the long-term simulation in practice. The problem can be further solved with more classical methods involved since it can provide a good initial value for the hash grid. It could be an interesting and promising topic for future work to construct a better hybrid simulator.

Acknowledgments

This work was supported by National Natural Science Foundation of China (No. 62071271, 62322110, 62171255, 52306149), Beijing Natural Science Foundation (No. JQ21012), Tsinghua-Toyota Joint Research Fund, Guoqiang Institute of Tsinghua University (No.2021GQG0001) and China Postdoctoral Science Foundation (No. 2022TQ0180).

References

Bridson, R. 2015. *Fluid simulation for computer graphics*. CRC press.

Chen, A.; Xu, Z.; Geiger, A.; Yu, J.; and Su, H. 2022. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision*, 333–350. Springer.

Chen, H.; Wu, R.; Grinspun, E.; Zheng, C.; and Chen, P. Y. 2023. Implicit Neural Spatial Representations for Time-dependent PDEs. In *International Conference on Machine Learning*, 5162–5177. PMLR.

Chen, Z.; and Zhang, H. 2019. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5939–5948.

Cho, J.; Nam, S.; Yang, H.; Yun, S.-B.; Hong, Y.; and Park, E. 2022. Separable PINN: Mitigating the Curse of Dimensionality in Physics-Informed Neural Networks. *arXiv preprint arXiv:2211.08761*.

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

Comba, J. L. D.; and Stol, J. 1993. A ne arithmetic and its applications to computer graphics. In *Proceedings of VI SIBGRAPI (Brazilian Symposium on Computer Graphics and Image Processing)*, 9–18.

Dong, S.; and Li, Z. 2021. Local extreme learning machines and domain decomposition for solving linear and nonlinear partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 387: 114129.

Fedkiw, R. P.; Aslam, T.; Merriman, B.; and Osher, S. 1999. A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method). *Journal of computational physics*, 152(2): 457–492.

Krishnapriyan, A.; Gholami, A.; Zhe, S.; Kirby, R.; and Mahoney, M. W. 2021. Characterizing possible failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems*, 34: 26548–26560.

Li, Z.; Kovachki, N.; Azizzadenesheli, K.; Liu, B.; Bhattacharya, K.; Stuart, A.; and Anandkumar, A. 2020. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*.

Li, Z.; Müller, T.; Evans, A.; Taylor, R. H.; Unberath, M.; Liu, M.-Y.; and Lin, C.-H. 2023. Neuralangelo: High-Fidelity Neural Surface Reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8456–8465.

Lu, L.; Jin, P.; and Karniadakis, G. E. 2019. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*.

Mescheder, L.; Oechsle, M.; Niemeyer, M.; Nowozin, S.; and Geiger, A. 2019. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 4460–4470.

Michalkiewicz, M.; Pontes, J. K.; Jack, D.; Baktashmotlagh, M.; and Eriksson, A. 2019. Implicit surface representations as layers in neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 4743–4752.

Mildenhall, B.; Srinivasan, P. P.; Tancik, M.; Barron, J. T.; Ramamoorthi, R.; and Ng, R. 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1): 99–106.

Morimoto, M.; Fukami, K.; Zhang, K.; Nair, A. G.; and Fukagata, K. 2021. Convolutional neural networks for fluid flow analysis: toward effective metamodeling and low dimensionalization. *Theoretical and Computational Fluid Dynamics*, 35(5): 633–658.

Müller, T.; Evans, A.; Schied, C.; and Keller, A. 2022. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4): 1–15.

Park, J. J.; Florence, P.; Straub, J.; Newcombe, R.; and Lovegrove, S. 2019. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 165–174.

Pfaff, T.; Fortunato, M.; Sanchez-Gonzalez, A.; and Battaglia, P. W. 2020. Learning mesh-based simulation with graph networks. *arXiv preprint arXiv:2010.03409*.

Raissi, M.; Perdikaris, P.; and Karniadakis, G. E. 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378: 686–707.

Sharp, N.; and Jacobson, A. 2022. Spelunking the deep: Guaranteed queries on general neural implicit surfaces via range analysis. *ACM Transactions on Graphics (TOG)*, 41(4): 1–16.

Sitzmann, V.; Martel, J.; Bergman, A.; Lindell, D.; and Wetzstein, G. 2020. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems*, 33: 7462–7473.

Stam, J. 1999. Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, 121–128.

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

Taylor, G. I.; and Green, A. E. 1937. Mechanism of the production of small eddies from large ones. *Proceedings of the Royal Society of London. Series A-Mathematical and Physical Sciences*, 158(895): 499–521.

Wang, P.; Liu, L.; Liu, Y.; Theobalt, C.; Komura, T.; and Wang, W. 2021. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *arXiv preprint arXiv:2106.10689*.

Yang, G.; Belongie, S.; Hariharan, B.; and Koltun, V. 2021. Geometry processing with neural fields. *Advances in Neural Information Processing Systems*, 34: 22483–22497.

Zehnder, J.; Li, Y.; Coros, S.; and Thomaszewski, B. 2021. Ntopo: Mesh-free topology optimization using implicit neural representations. *Advances in Neural Information Processing Systems*, 34: 10368–10381.