IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 9 July 2025, accepted 18 August 2025, date of publication 21 August 2025, date of current version 2 September 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3601241

RESEARCH ARTICLE

Learning Representation of Turbulent Vector Fields via Efficient Moving Least Squares Based on Monte Carlo Method

JONG-HYUN KIM[1] AND JUNG LEE[2]

[1]College of Software and Convergence (Department of Artificial Intelligence, Design Technology), Graduate School of Electrical and Computer Engineering, Inha University, Michuhol-gu, Incheon 22212, South Korea
[2]Department of Computer Engineering, Hanbat National University, Yuseong-gu, Daejeon 34158, South Korea

Corresponding author: Jung Lee (airjung@hanbat.ac.kr)

This work was supported in part by Inha University Research Grant (40%); in part by the National Research Foundation of Korea (NRF) grant funded by Korean Government (MSIT) under Grant RS-2023-00254695, 30%; and in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) Grant funded by Korean Government (MSIT) (Artificial Intelligence Convergence Innovation Human Resources Development, Inha University, 30%) under Grant RS-2022-00155915.

ABSTRACT In this study, we develop a numerical method to represent turbulent flow in various 2D vector fields using the Monte Carlo method-based MLS (Moving Least Squares) from a density field and express it as a learning representation through a neural network. Conventional MLS performs high-order interpolation solely based on vector-based constraints, making it difficult to effectively reflect the characteristics of a density field, which limits its applicability in various fields. Additionally, equations of higher degree require significant computational effort, making them costly in terms of computation time and resources when applied to simulation tasks that require per-frame calculations. To address these issues, this study integrates the Monte Carlo method-based weighting into MLS to efficiently consider the characteristics of the density field in the input data and design an algorithm that represents it as various forms of vector fields. Furthermore, we extend the solver to express this approach as a learning representation through a neural network. To validate the applicability of our method, we conducted experiments extracting turbulent vector fields from various density fields. Since conventional MLS does not guarantee temporal continuity, directly applying it to simulations results in noise. To resolve this, our method generates turbulent flow by analyzing the angular variation between the generated velocity and the underlying fluid velocity, ensuring stable advection of the density. As a result, our method efficiently and accurately extracts turbulent flow from a density field and integrates it into an underlying fluid solver, enabling the practical use of high-order interpolation in physics-based simulations. Experimental results across various scenarios demonstrate that our method improves both computation time and quality compared to previous methods, yielding enhanced turbulent flow fields.

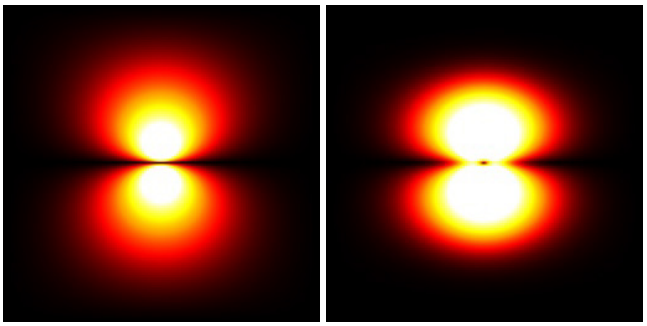
INDEX TERMS Fluid simulation, turbulent vector fields, turbulent flow, moving leasts squares, Monte Carlo method, high-order interpolation.

I. INTRODUCTION

MLS is primarily used to perform a smooth approximation or high-order interpolation for scattered data. When applied to a PDF (Probability Density Function), MLS enables more accurate probability distribution calculations, making it useful in various fields such as fluids [1], [2], deformable bodies [3], and surface reconstruction [4], [5]. Polynomial-based interpolation is commonly used to retrieve data between sampling points, making it particularly effective for representing the dynamic movement of continuous materials in applications such as fluid simulation and character modeling. In such cases, the selection of an appropriate interpolation method is crucial, as it

The associate editor coordinating the review of this manuscript and approving it for publication was Songwen Pei.

© 2025 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License.
For more information, see https://creativecommons.org/licenses/by/4.0/



(a) Standard MLS (b) Divergence-constrained MLS

FIGURE 1. Comparison of standard MLS and pseudo divergence-constrained vector interpolation [6].

significantly affects the quality of the final output [6], [7], [8].

In geometry processing, interpolation is used to approximate mesh surfaces from point cloud data. Since point clouds lack connectivity information, a high-order numerical scheme is employed to implicitly construct a mesh made up of triangles [4]. In the field of physics-based simulation, the process of obtaining sampled data through interpolation is a crucial step in physics-based simulation. High-order numerical interpolation methods are used to improve the simulation accuracy [9]. To enhance the semi-Lagrangian advection [10], which relies on linear interpolation for the advection term in smoke simulation, several advanced techniques have been proposed, including BFECC (Back and Forth Error Compensation and Correction) [11], CIP (Constrained Interpolation Profile) [12], USCIP (Unsplit Semi-Lagrangian CIP) [13], stream-guided smoke [14], frequency-domain smoke [15], adaptive vortex particle methods [16], and Polynomial PIC (Particle-In-Cell) [17].

Feldman et al. computed the MLS based on the normal component of the velocity to obtain a smooth vector field on a hybrid mesh [18]. Although MLS is a technique for high-order interpolation and is applied in various fields, as mentioned earlier, it has not been widely utilized in simulations because of difficulties in satisfying conditions such as continuity and physical constraints. To address this, Hong et al. proposed a divergence-constrained MLS method, which enables the generation of a vector field that mimics fluid flow without solving the Poisson equation [6]. This approach was later applied to smoke simulation to represent the motion of complex turbulent flow [7].

Figure 1 illustrates the differences in vector interpolation results between standard MLS and divergence-constrained MLS proposed by Hong et al. [6]. Figure 1a shows the result of computing MLS using two opposing vertical vectors as constraints, while Figure 1b presents the vector field obtained using divergence-constrained MLS. Divergence-constrained MLS ensures incompressibility, making it a suitable interpolation method for fluid simulation (see Figure 1b), and its difference from standard MLS is clearly evident (see Figure 1a). Notably, this approach allows vector constraints to be expressed differently based on divergence conditions, making it useful for designing various types of vector fields or controlling physics-based simulations. However, since constraints are represented solely in vector form, they do not account for density characteristics. Given that density is a crucial physical property in smoke and flame simulations, its characteristics must be properly considered.

In this study, we used the Monte Carlo method to estimate the locations where the density is distributed and used the distances between these estimated locations and the surrounding data as MLS weights to visualize a density-sensitive vector field. In addition, we apply this method to physics-based simulation to demonstrate its capability in stably representing turbulent flow. Finally, we extend the solver using a neural network to further maximize its efficiency.

MLS has been widely used in meshless methods such as point clouds and SPH (Smoothed Particle Hydrodynamics) for applications including surface reconstruction [19], fracture [20], and fluid simulation [1]. It is also utilized for approximating surfaces in point set surfaces, which lack connectivity information.

In SPH simulations, blobby noise often appears at boundaries composed of particles, leading to unintended motion. To address this issue, MLS has been used in the form of MLS pressure boundaries [21]. Additionally, in hybrid meshes, where the structure is not in a regular grid form, velocity fields may appear non-smooth, and MLS has been applied to mitigate this effect [18].

Another application is MLS-MPM (Moving Least Squares Material Point Method), where Galerkin-style MLS has been utilized to efficiently compute MPM [22]. As mentioned earlier, MLS has primarily been used to obtain smoothing fields in polynomial-based spaces. However, because of its high computational cost, its use has been limited to specific fields.

A. PROBLEM STATEMENT

In this study, our objective is to apply grid-based MLS for vector interpolation to compute turbulent flow. Kim and Hong proposed a method to synthesize turbulent flow in Eulerian fluid simulation using MLS [7]. Although they replaced semi-Lagrangian advection [10] with divergence-constrained MLS [6], their approach has several limitations, which we address in this work.

1) MLS vector interpolation without considering density features: In the previous method, MLS was designed for grid-based interpolation by following the approach used in semi-Lagrangian schemes, where constraints are defined using the fluid velocity at simulation nodes [7] (see Figure 2). The semi-Lagrangian method traces the velocity field backward by $u\Delta t$ and determines the sampling position x. The value of a variable stored and advected at the sample nodes for an example at $p_{i,j}$ is updated by calculating it at x by building an interpolation profile from neighboring node values at $p_{i,j}$, $p_{i+1,j}$, $p_{i,j+1}$, and $p_{i+1,j+1}$ (red: simulation node,

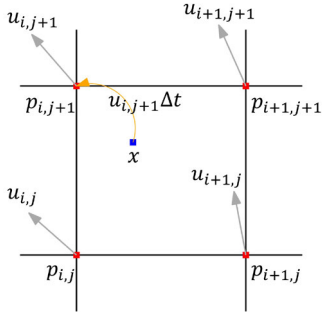


FIGURE 2. Illustration of the semi-Lagrangian method.

blue: sampling position, gray arrow: fluid velocity, orange arrow: backward sampling). However, despite the fact that smoke and fire possess physical properties, such as density and temperature, the characteristics of the density field were not considered when computing turbulent flow within the grid. In this study, we efficiently address this issue by integrating MLS based on the Monte Carlo method.

2) Noisy turbulent flow caused by a lack of correlation with the underlying fluid flow: Turbulence is typically used to enhance the subgrid details of the fluid by building on the underlying fluid flow. However, the MLS proposed in the previous method results in an uncontrolled noise field, making it difficult to claim that it effectively improves the details of the underlying fluid [7] (see Figure 3). In this study, instead of directly using MLS-based turbulence, we refine the noise-like turbulent flow by leveraging the angular difference between the velocity of the underlying fluid and the turbulent flow computed using MLS.

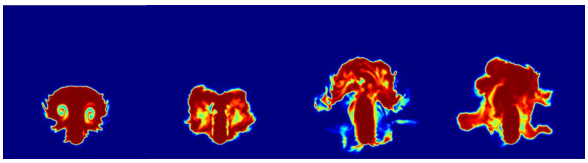
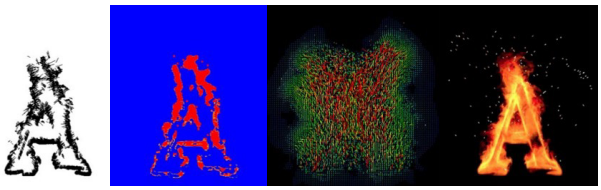


FIGURE 3. Turbulent flow with previous method [7].

3) Small time step(Δt) required due to the noisy field generated by MLS: In the previous method, very small Δt or substeps were used as a subproblem related to the aforementioned issue [7]. This problem is not solely due to MLS itself, but rather arises because the vector field computed using divergence-constrained MLS is inherently unstable, and directly using it for advection exacerbates the issue. In this study, we improve the stability of the simulation to the extent that larger time steps can be accommodated.
4) High computational cost due to the size of the $N \times N$ matrix, which corresponds to the number of nodes: Due to the low simulation stability, a small Δt is determined using the CFL condition. However, the high computational cost of MLS results in long computation times [7].



(a) Feature vector **(b)** Flame region **(c)** Velocity field **(d)** Fire-flake

FIGURE 4. The previous method of extracting a feature vector from an image, calculating the flow from it, and simulating the secondary effects of fire-flake [23].

B. IMPORTANCE OF 2D SIMULATIONS AS MUCH AS 3D

In physics-based fluid simulation, utilizing turbulent flow for video content creation is challenging because it requires not only complex numerical analysis but also a deep understanding of algorithms, making it less accessible to artists and creators. Additionally, most methods are designed with 3D simulations in mind, although the 2D market is just as important. Furthermore, 3D simulations demand significant computational resources, making real-time processing difficult without high-performance hardware. To overcome these limitations, Kim et al. proposed a method that uses videos or animated GIFs as input data to quickly compute fluid flow and simulate fire sparks in real time [23]. In this study, we propose an MLS-based turbulent flow method specifically designed for 2D rather than 3D. In the results section, we demonstrate not only the MLS turbulent flow generated by our method but also its applicability by integrating it with the previous method.

II. RELATED WORK

A. TURBULENT FLOW WITH NUMERICAL ANALYSIS

In the field of fluid simulation, various advection and external force methods have been proposed to enhance the visual detail of turbulent flow [10], [24], [25]. The semi-Lagrangian advection method was introduced to achieve a stable fluid simulation [10], while the vorticity confinement method estimated the vorticity force in a grid-based representation of turbulent flow [26]. This approach was later improved using Lagrangian particles as vortex carriers, leading to the development of the vortex particle method [27]. Furthermore, several techniques have been proposed to further enhance turbulent flow simulation, including BFECC [11], CIP [12], USCIP [13], stream-guided smoke [14], frequency-domain smoke [15], adaptive vortex particle methods [16], and Polynomial PIC [17].

Many techniques have sought to enhance the details of the subgrid through particle simulation. Kim et al. [28] and Losasso et al. [29] used the SPH model to address subgrid splashes. Greenwood et al. [30], Hong et al. [31], and Thüerey et al. [32] employed hybrid approaches that incorporate particles to achieve more detailed representations

of multiphase fluids. Gao et al. used Lagrangian particles to capture the characteristics of high-speed gas flows [33]. Incorporating physical models into subgrid dynamics can enhance details in multiphase interfaces [34], flames [35], [36], and liquids [37]. Hong et al. proposed a divergence-constrained MLS method capable of representing fluid movement without solving the Poisson equation [6]. Later, this method was extended and integrated into the advection term of smoke simulation to represent turbulent flow [7]. However, MLS-based turbulent flow exhibits significant noisy motion, making it difficult to control in simulations. Additionally, the high computational cost of high-order equations limits its applicability across various industries.

There are also post-processing approaches that enhance flow details. Some of these methods synthesize turbulent flow from low-resolution simulations using techniques such as wavelet noise [38] and curl noise [39], [40]. Chu and Thuerey used CNNs (convolutional neural networks) to learn the relationship between low-resolution and high-resolution flows [41]. Sato et al. proposed a method that transfers detailed turbulence to low-resolution flows using patch-based texture synthesis [42]. More recently, Chen et al. introduced an integrated approach that combines continuum mechanics-based simulation with MLS to model interactions between paper and fluid, capturing tearing effects [43]. Although these approaches generate plausible details, they often introduce noise flow that deviates from the underlying fluid motion. Chen et al. extended MLS to multiphase continuum simulation, highlighting the strengths of MLS in handling complex boundary interactions [43]. While our study directly uses MLS to control fluid motion, their approach differs in that MLS is applied primarily to enforce boundary conditions.

In this paper, we use the Monte Carlo method for weighting in MLS, a sampling technique widely used in various fields. Sawhney and Crane proposed a Monte Carlo method for solving partial differential equations without a grid, relying solely on a geometric model [44]. More recently, Monte Carlo methods have also been applied in AI-driven approaches to 3D scene reconstruction [45].

B. DEEP LEARNING FOR FLUIDS

The use of machine learning architectures to regress fluid representation was first demonstrated by Ladický et al. [46]. They utilized regression forests to approximate a Lagrangian fluid solver, producing impressive particle-based fluid results. CNN-based architectures have been used to replace the pressure projection step in Eulerian-based solvers [47], [48], synthesize fluid simulations from reduced parameter sets [49], and approximate steady-state velocity fields to predict aerodynamic forces [50]. Low-resolution fluid simulations have been upsampled to better match high-resolution simulations using patch-based [41], [51] and dictionary-based approaches [52], [53]. Recently, differentiable simulation pipelines, which naturally integrate with computer vision and deep learning architectures, have emerged as a growing trend [54], [55], [56], [57]. Deep learning has also been applied to reconstruct volumetric flows from images using transport constraints and self-supervision [58], enable graph-based Lagrangian fluid simulation [59], enhance the super-resolution of unsteady data [60], and facilitate interactive modeling of liquids [61] and smoke using sketches [62]. In addition, deep learning has been used for meaningful control in learning-based simulations [63].

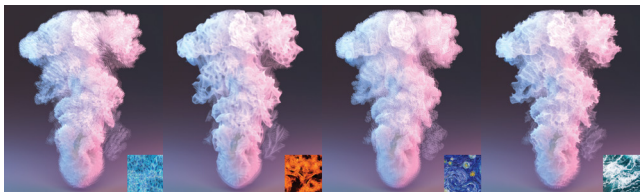


FIGURE 5. Single-view stylization for various styles [64].

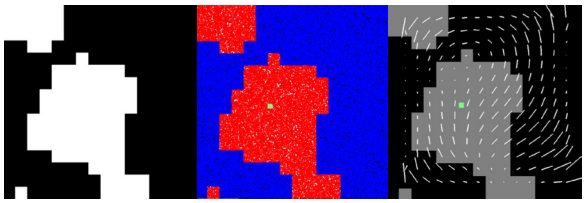
Aurand et al. proposed an efficient neural network approach for style transfer in volumetric simulations [64] (see Figure 5). Performing artistic simulations of fluids is a particularly challenging task within the field of simulation control. Recently, volumetric NST (Neural Style Transfer) techniques have been used to artistically manipulate smoke simulation data based on 2D images. However, NST differs from the turbulent flow studied in our research. As shown in Figure 5, NST does not alter the motion details of the underlying simulation due to turbulent flow; instead, it maps the style of a 2D image to the input simulation, similar to texture synthesis. In general, despite variations in the input 2D images, the NST results exhibit consistent macroscopic motion while only modifying the internal density patterns (see Figure 5).

Franz et al. proposed a method for fluid flow reconstruction from images using self-supervised learning, enabling the prediction of 3D flow fields from 2D images [58]. Kim et al. enhanced artistic controllability by reconstructing 3D smoke density fields from sketches through deep learning [62]. Chu et al. explored controllable learning-based flow generation within real fluid simulations [63]. Aurand et al. presented a deep learning-based representation method that applies style transfer to fluid simulation data, aiming to reconstruct vector or density fields effectively within a learned space—an approach aligned with the goals of this study [64].

III. PROPOSED FRAMEWORK

To explain the proposed weighting based on the Monte Carlo method for divergence-constrained MLS, we first introduce the fundamental MLS approach for scalar and vector interpolation. Then, we propose a method to estimate the position of the density field using the Monte Carlo method, compute the corresponding MLS weights, and ultimately derive the turbulent vector field.

Figure 6 illustrates the algorithm overview of our proposed method. We introduce a novel technique that adjusts the



(a) Step1 (b) Step2 (c) Step3

FIGURE 6. Algorithm overview of our method (step1: input data(2D density field), step2: calculation of MLS weighted based on Monte Carlo method, step3: calculation of divergence-constrained MLS with Monte Carlo weight).

MLS weights using the Monte Carlo method to visualize the vector field while considering the density field. This approach overcomes the limitation of conventional MLS, which relies only on vector-based constraints, enabling a more detailed representation of the vector field by incorporating density information.

A. MOVING LEAST SQUARES

1) SCALAR INTERPOLATION

If the N control points p_i are located within a specific region, the polynomial using these points is defined as follows (see Equation 1).

$$f(\mathbf{x}) = \mathbf{b}^T(\mathbf{x})\mathbf{c} \tag{1}$$

The approximate value p_i at the control point ϕ^i is as shown in Equation 2.

$$f(\mathbf{p}_i) = \mathbf{b}^T(\mathbf{p}_i)\mathbf{c} = \phi^i, \tag{2}$$

where the sampling position $\mathbf{x}$ is represented as $[x, y]$ in 2D and $[x, y, z]$ in 3D. The term $\mathbf{b}(\mathbf{x}) = [b_1(\mathbf{x}), \ldots, b_k(\mathbf{x})]$ denotes the basis functions of the polynomial, and $\mathbf{c} = [c_1, \ldots, c_k]$ is the unknown coefficient vector. In general, the number of elements in $\mathbf{b}(\mathbf{x})$ is given by $k = \frac{(d+m)!}{m!d!}$, where f is determined from $\prod_d^m$, representing a polynomial space defined by the spatial dimension m and degree d. For the 2D case (d =2), it can be expressed as shown in Equation 3, and for the 3D case (d =3), it is given in Equation 4.

$$\begin{aligned}
&\mathbf{b}(\mathbf{x}) = [1] \text{ if } m = 0 \\
&\mathbf{b}(\mathbf{x}) = [1, x, y] \text{ if } m = 1 \\
&\mathbf{b}(\mathbf{x}) = [1, x, y, x^2, y^2, xy] \text{ if } m = 2
\end{aligned} \tag{3}$$

$$\begin{aligned}
&\mathbf{b}(\mathbf{x}) = [1] \text{ if } m = 0 \\
&\mathbf{b}(\mathbf{x}) = [1, x, y, z] \text{ if } m = 1 \\
&\mathbf{b}(\mathbf{x}) = [1, x, y, z, x^2, y^2, z^2, xy, yz, xz] \text{ if } m = 2
\end{aligned} \tag{4}$$

As explained in Equation 2, MLS can be obtained by applying constraints to all values. The weight according to the constraint is given by $\omega(|\mathbf{r}_i|)$, where $\mathbf{r}_i = \mathbf{p}_i - \mathbf{x}$. This equation can then be formulated as shown in Equation 5.

$$\begin{bmatrix} \omega(|\mathbf{r}_1|) & & \\ & \ddots & \\ & & \omega(|\mathbf{r}_N|) \end{bmatrix} \begin{bmatrix} \mathbf{b}^T(|\mathbf{p}_1|) \\ \vdots \\ \mathbf{b}^T(|\mathbf{p}_N|) \end{bmatrix} \mathbf{c} = \begin{bmatrix} \omega(|\mathbf{r}_1|) & & \\ & \ddots & \\ & & \omega(|\mathbf{r}_N|) \end{bmatrix} \begin{bmatrix} \phi_i \\ \vdots \\ \phi_N \end{bmatrix} \tag{5}$$

This can be simplified as shown in Equation 6.

$$\mathbf{W}(\mathbf{x})\,\mathbf{B}\mathbf{c}(\mathbf{x}) = \mathbf{W}(\mathbf{x})\,\Phi, \tag{6}$$

where $\mathbf{W}$ is an $N \times N$ diagonal matrix, $\mathbf{B}$ is an $N \times k$ matrix, and Φ is a vector in $\mathbb{R}^N$. Applying the normal equation to this equation yields the following expression.

$$\mathbf{B}^T(\mathbf{W}(\mathbf{x}))^2\,\mathbf{B}\mathbf{c}(\mathbf{x}) = \mathbf{B}^T(\mathbf{W}(\mathbf{x}))^2\,\Phi, \tag{7}$$

Solving this equation as a linear system yields $\mathbf{c}(\mathbf{x})$. Using this, the value of f at position $\mathbf{x}$ is computed using Equation 1. In this study, we performed this computation using the singular value decomposition.

2) VECTOR INTERPOLATION

The previous section explained how to extend scalar interpolation to a vector field $\mathbf{u} = [u, v, w]^T$. By simply replacing f with a vector, the values of $\mathbf{u}$ can be calculated, leading to the constraints formulated in Equation 8.

$$\mathbf{u}(\mathbf{p}_i) = \begin{bmatrix} u(\mathbf{p}_i) \\ v(\mathbf{p}_i) \\ w(\mathbf{p}_i) \end{bmatrix} = \begin{bmatrix} \mathbf{b}^T(\mathbf{p}_i\mathbf{c}_x) \\ \mathbf{b}^T(\mathbf{p}_i\mathbf{c}_y) \\ \mathbf{b}^T(\mathbf{p}_i\mathbf{c}_z) \end{bmatrix} = \begin{bmatrix} u_i \\ v_i \\ w_i \end{bmatrix} = \mathbf{u}_i. \tag{8}$$

In general, vector interpolation can be computed by independently interpolating each component. However, when there are interdependencies between components and a divergence constraint must be satisfied, the interpolation must be reformulated. Consequently, to interpolate the vector, we rewrite Equation 2 as Equation 9.

$$\mathbf{W}_{vec}(\mathbf{x})\,\mathbf{B}_{vec}\mathbf{c}_{vec}(\mathbf{x}) = \mathbf{W}_{vec}(\mathbf{x})\,\Phi_{vec}, \tag{9}$$

where

$$\mathbf{W}_{vec} = \begin{bmatrix} \omega(|\mathbf{r}_1|) & & & & & \\ & \omega(|\mathbf{r}_1|) & & & & \\ & & \omega(|\mathbf{r}_1|) & & & \\ & & & \ddots & & \\ & & & & \omega(|\mathbf{r}_N|) & \\ & & & & & \omega(|\mathbf{r}_N|) \\ & & & & & \omega(|\mathbf{r}_N|) \end{bmatrix}, \quad \mathbf{B}_{vec} = \begin{bmatrix} \mathbf{b}(\mathbf{p}_1) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{b}(\mathbf{p}_1) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{b}(\mathbf{p}_1) \\ & \vdots & \\ \mathbf{b}(\mathbf{p}_N) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{b}(\mathbf{p}_N) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{b}(\mathbf{p}_N) \end{bmatrix}. \tag{10}$$

where $\mathbf{c}_{vec} = [\mathbf{c}_x, \mathbf{c}_y, \mathbf{c}_z]$ and $\Phi_{vec} = [\mathbf{u}_1, \ldots, \mathbf{u}_N]$. The matrix $\mathbf{W}_{vec}$ is a $dN \times dN$ diagonal matrix, $\mathbf{B}_{vec}$ is a $dN \times dk$ matrix, and Φ is a vector in $\mathbb{R}^{dN}$. In this study, vector interpolation is computed by solving Equation 9 using the same approach as solving Equation 6 for scalar interpolation.

B. DIVERGENCE-CONSTRAINED MLS WITH MONTE CARLO WEIGHTS

In this section, we introduce the diffuse derivatives and the MLS method based on Monte Carlo weights.

1) DIFFUSIVE DERIVATIVES

As shown in the following equation, differentiating Equation 1 provides the exact derivative of f (see

Equation 11).

$$\frac{\partial f}{\partial \mathbf{x}} = \frac{\partial \mathbf{b}^T}{\partial \mathbf{x}}\mathbf{c} + \mathbf{b}^T \frac{\partial \mathbf{c}}{\partial \mathbf{x}} \tag{11}$$

In Equation 11, the second term on the right-hand side can be computed straightforwardly, but its integration into MLS is computationally complex. By replacing this term with diffuse derivatives, an efficient approximation of the computationally expensive derivative can be obtained [6] (see Equation 12).

$$\frac{\partial f}{\partial \mathbf{x}} = \frac{\partial \mathbf{b}^T}{\partial \mathbf{x}}\mathbf{c} + \mathbf{b}^T \frac{\partial \mathbf{c}}{\partial \mathbf{x}} \tag{12}$$

Using diffuse derivatives, the divergence of the vector field $\nabla \cdot \mathbf{u} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}$ can be approximated as shown in Equation 13.

$$\nabla \cdot \mathbf{u} \simeq \frac{\partial \mathbf{b}^T}{\partial x}\mathbf{c}_x + \frac{\partial \mathbf{b}^T}{\partial y}\mathbf{c}_y + \frac{\partial \mathbf{b}^T}{\partial z}\mathbf{c}_z \tag{13}$$

2) DIVERGENCE CONSTRAINT

Huerta et al. improved the accuracy of incompressibility by enforcing the divergence constraint $\nabla \cdot \mathbf{u} = \sigma$ with $\sigma = 0$ [65]. In this study, we allow σ to take nonzero values, allowing divergence control during the velocity interpolation process. To incorporate the divergence constraint into $\mathbf{B}_{vec}$, Equation 13 must be rewritten as Equation 14.

$$\nabla \cdot \mathbf{u} \simeq \mathbf{b}_{div}^T(\mathbf{x}) \cdot \mathbf{c} \tag{14}$$

where $\mathbf{b}_{div}$ is computed as shown in Equation 15.

$$\begin{aligned}\mathbf{b}_{div}(\mathbf{x}) &= \frac{\partial \left[\mathbf{b}^T(\mathbf{x}) \;\; 0 \;\; 0\right]}{\partial x} + \frac{\partial \left[0 \;\; \mathbf{b}^T(\mathbf{x}) \;\; 0\right]}{\partial y} \\ &\quad + \frac{\partial \left[0 \;\; 0 \;\; \mathbf{b}^T(\mathbf{x})\right]}{\partial z} \\ &= \left[\frac{\partial \mathbf{b}^T(\mathbf{x})}{\partial x} \;\; 0 \;\; 0\right] + \left[0 \;\; \frac{\partial \mathbf{b}^T(\mathbf{x})}{\partial x} \;\; 0\right] \\ &\quad + \left[0 \;\; 0 \;\; \frac{\partial \mathbf{b}^T(\mathbf{x})}{\partial z}\right] \\ &= \left[\frac{\partial \mathbf{b}^T(\mathbf{x})}{\partial x} \;\; \frac{\partial \mathbf{b}^T(\mathbf{x})}{\partial y} \;\; \frac{\partial \mathbf{b}^T(\mathbf{x})}{\partial z}\right]\end{aligned} \tag{15}$$

The partial derivatives of $\mathbf{b}$ are computed by differentiating each element individually. For example, when $d = 3$ and $m = 2$, the result is given in Equation 16.

$$\begin{aligned}\frac{\partial \mathbf{b}(\mathbf{x})}{\partial x} &= [0, 1, 0, 0, 2x, 0, 0, y, 0, z] \\ \frac{\partial \mathbf{b}(\mathbf{x})}{\partial y} &= [0, 0, 1, 0, 0, 2y, 0, x, z, 0] \\ \frac{\partial \mathbf{b}(\mathbf{x})}{\partial z} &= [0, 0, 0, 1, 0, 0, 2z, 0, y, x]\end{aligned} \tag{16}$$

By incorporating the divergence-constraint condition $\mathbf{b}_{div}^T \cdot \mathbf{c} = \sigma$ into Equation 9, the divergence-constraint-based MLS can be computed (see Equation 17).

$$\mathbf{W}_{div}(\mathbf{x})\,\mathbf{B}_{div}\mathbf{c}_{vec}(\mathbf{x}) = \mathbf{W}_{div}(\mathbf{x})\,\Phi_{div} \tag{17}$$

This equation can be computed using the MLS method described earlier. The MLS constraints used in this study are as follows: $\mathbf{W}_{div} = \begin{bmatrix}\mathbf{W}_{vec} \\ \omega_{div}\end{bmatrix}$, $\mathbf{B}_{div} = \begin{bmatrix}\mathbf{b}_{vec} \\ \omega_{div}\end{bmatrix}$, $\Phi_{div} = [\Phi, \sigma]$ where $\mathbf{W}_{div}$ is a $(dN+1) \times (dN+1)$ diagonal matrix, $\mathbf{B}_{div}$ is a $(dN+1) \times dk$ matrix, and Φ_{div} is a vector in $\mathbb{R}^{dN+1}$.

3) MONTE CARLO-BASED WEIGHTS

The Monte Carlo method approximates values using stochastic random numbers instead of solving problems through direct numerical computation. Its accuracy improves with the number of random samples and the uniformity of their distribution. One of the simplest and most well-known examples of the Monte Carlo method is the estimation of π.

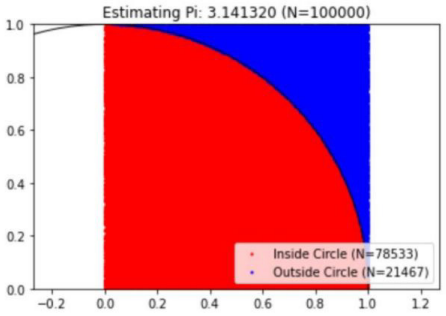


FIGURE 7. An example of calculating π using the Monte Carlo method.

First, a square and a circle with a radius of $\frac{1}{4}$ of the side length of the square are drawn, and N random points are generated within the square (see Figure 7). Then, the distance between the center of the circle and each random point is calculated to determine whether the point is within the circle. When a sufficient number of random points are generated, the relationship in Equation 18 is maintained where α represents the number of points within the circle and N is the total number of points generated.

$$\frac{\pi}{4} : 1 = a : N,\ \pi = \frac{4a}{N} \tag{18}$$

In this study, the Monte Carlo method is used to estimate high-density regions, following a principle similar to π estimation. In Equation 19, d represents the estimated density, a denotes the number of random samples within regions that exceed a certain density threshold and N is the total number of random samples generated.

$$d : 1 = a : N,\ d = \frac{a}{N} \tag{19}$$

In this study, to efficiently detect high-density regions, we first binarize the density field. Figure 8a shows the original density field, while Figure 8b presents the binarized result using a threshold of 0.5. Figures 8c and 8d illustrate the density field estimated using the Monte Carlo method with 1,000 and 100,000 random samples, respectively. When the mean density of the original field is 0.3476, the estimated density in Figure 8c is measured as 0.324, whereas Figure 8d

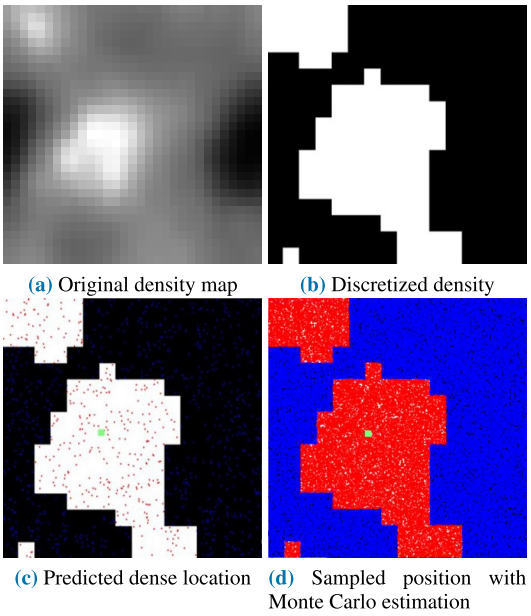


FIGURE 8. A density field estimated using the Monte Carlo method.

gives a more accurate estimate of 0.3452, demonstrating that a higher number of random samples leads to a more precise estimate. Based on this, high-density regions in the density field can be identified. The green points in Figures 8c and 8d represent the detected high-density locations. Although the dense regions in Figure 8c vary slightly between simulations, Figure 8d produces more consistent results.

The dense regions of the density field computed using the Monte Carlo method are identified, and the distances between these regions and each constraint node are calculated. The results for the constraint nodes are then normalized to the range of 0 to 1. The Monte Carlo method is used to construct the MLS weight $\omega^{\star}(|\mathbf{r}_i|)$, as formulated in Equation 20.

$$\omega^{\star}(|\mathbf{r}_i|) = \omega(|\mathbf{r}_i|)(\mathbf{q}_i - \mathbf{x}) \tag{20}$$

where $\omega(|\mathbf{r}_i|)$ is the weight computed based on the distance between a node and a constraint node, and $(\mathbf{q}_i - \mathbf{x})$ represents the distance between a node and the dense regions estimated from the density field using the Monte Carlo method. Using these values, the weight matrix $\mathbf{W}$ is constructed to compute MLS.

C. SOLVER EXTENSION

In this section, we propose a method to efficiently represent vector fields generated by Monte Carlo weight-based divergence-constrained MLS by training a neural network to learn MLS, a type of polynomial interpolation. Since MLS must perform vector interpolation instead of scalar interpolation to construct vector fields, the size of matrices and vectors increases, leading to higher computational costs. High-order interpolation is computationally expensive, making it difficult to apply in simulation tasks that require per-frame energy field calculations. To mitigate this issue, we employ

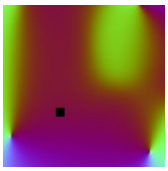


FIGURE 9. Geometry image (black: high-density location).

a learning representation. As a result, we demonstrate that the trained Monte Carlo weight-based MLS can effectively represent turbulent vector fields.

1) CONVERSION BETWEEN MLS VECTOR FIELD AND GEOMETRY IMAGE

In this study, we used geometry images to enable the learning representation of vector fields generated by Monte Carlo weight-based MLS. This requires converting the velocity at each node into an RGB color representation, $[r, g, b]$ (see Figure 9).

Since the encoded values must not exceed the valid color range, we constrain the magnitude of the constraint vectors to be no greater than 1. When we convert back from color fields to velocity fields, we apply the inverse function. To train the model, we construct a dataset consisting of the following data pairs for various vector fields: 1) The original velocity field and the locations of high-density regions. 2) The vector field generated using Monte Carlo weight-based MLS.

2) NEURAL NETWORKS

There are various approaches to learning representations of vector fields; however, in this study, we used a relatively simple image super-resolution neural network to learn the Monte Carlo weight-based MLS vector field (see Figure 10).

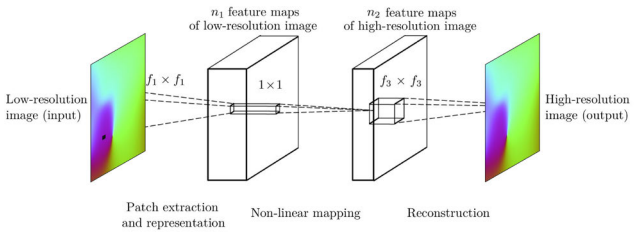
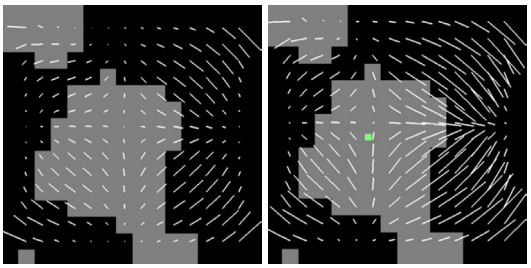


FIGURE 10. Neural network architecture.

Super-resolution (SR) is a technique that transforms low-resolution data into high-resolution representations. Dong et al. applied CNNs to SR to enhance image quality [66]. In this study, we train the network by converting the vector field (u, v, w) into an RGB representation $[r, g, b]$. Our method takes a low-resolution velocity field and high-density location data as input, processes them through convolutional filters, and generates a high-resolution Monte Carlo weight-based MLS vector field. The network model used in this study follows the same structure as CNN-based image SR,



(a) Previous method [6] (b) Our method

FIGURE 11. Comparison of vector fields generated by the previous method and our method (scene1).

consisting of three stages: patch extraction / representation, nonlinear mapping and reconstruction [66].

The first stage is patch extraction and representation. This step extracts multiple patches from the low-resolution image and represents them as a set of feature maps. This process can be formulated as shown in Equation 21.

$$F_1(Y) = max(0, W_1 Y + B_1) \tag{21}$$

where Y represents the original velocity field, and the equation extracts patches through a convolution filter, generating the feature map $F_1(X)$.

The second stage is nonlinear mapping, where the feature maps representing low-resolution velocity field patches are nonlinearly mapped to high-resolution MLS velocity field patch representations. This process can be formulated as shown in Equation 22.

$$F_2(Y) = max(0, W_2 F_1(Y) + B_2) \tag{22}$$

The feature map of the low-resolution original velocity field, generated in the patch extraction and representation stage, is transformed into the feature map of the high-resolution MLS velocity field.

The third stage is reconstruction, where the transformed feature map set of the MLS velocity field is reconstructed into the high-resolution MLS velocity field. This process is formulated as shown in Equation 23.

$$F(Y) = W_3 F_2(Y) + B_b \tag{23}$$

In the reconstruction stage, the feature map representing the high-resolution MLS velocity field, transformed in the nonlinear mapping stage, is passed through a convolution filter to reconstruct the high-resolution MLS velocity field. After the image of the low-resolution velocity field Y passes through these three stages, the reconstructed MLS velocity field $F(Y)$ closely approximates the high-resolution MLS velocity field X.

IV. IMPLEMENTATION DETAILS

Detailed Description of the Neural Network Architecture: In this study, we adopt a Super-Resolution (SR)-based CNN architecture to efficiently learn and represent the Monte Carlo-based MLS vector field.

- The overall network consists of a three-stage architecture, with each component structured as follows:
  1) Patch Extraction and Representation
     a) Kernel size: 9×9, Number of channels: 64, Activation function: ReLU
     b) Input: Low-resolution velocity field and high-density location information
  2) Nonlinear Mapping
     a) Kernel size: 1×1, Number of channels: 32, Activation function: ReLU
     b) Learning nonlinear relationships between intermediate feature maps
  3) Reconstruction
     a) Kernel size: 5×5, Number of output channels: 3 (vector field encoded as RGB)
     b) Outputs the final high-resolution vector field.
- Training Conditions:
  1) Optimizer: Adam (learning rate = 0.0003)
  2) Loss Function: Mean Squared Error (MSE)
  3) Batch size: 128, Epoch: 200
  4) The vector field is represented as a geometry image by mapping the $[u, v]$ values to $[r, g, b]$.
- Comparison of Elapsed Time at 256×256 Resolution:
  1) Direct MLS computation takes 10 to 72 seconds.
  2) The trained model generates results in approximately 1.2 seconds on average.

V. EXPERIMENTAL RESULTS

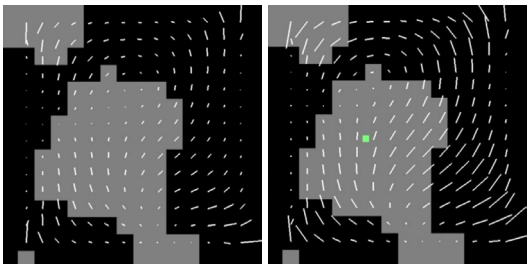
To generate the results of this study, we used a computer equipped with an Intel Core i7-7700K CPU, 32GB RAM, and a GeForce GTX 1080Ti GPU. Our approach demonstrates not only the generation of turbulent vector fields from 2D density field data but also improved results when applied to smoke simulation and a fire-flake solver.

A. EXTRACTING A VECTOR FIELD FROM A 2D DENSITY FIELD

1) RESULTS WITH NUMERICAL METHOD

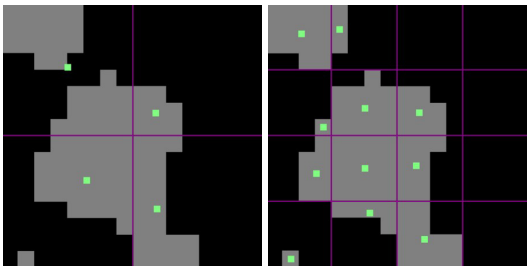
In this section, we describe the process of extracting vector fields using Monte Carlo weight-based MLS. Figure 11 presents the extracted vector field in a scene with constraint nodes. Figure 11a shows the vector field computed using the previous method, divergence-constrained MLS, with four constraint nodes positioned at the corners: ($p1(0.0558, -0.9984)$, $p2(-0.4041, 0.9147)$, $p3(-0.6049, -0.7962)$ and $p4(-0.7164, 0.6976)$). In the previous method, the resulting vector field is determined only by the direction of the constraint nodes, without considering the characteristics of the density field. Figure 11b shows the result when Monte Carlo-based weighting is applied to the MLS under the same conditions. Unlike Figure 11a, the vector field flow is more detailed while maintaining the structure of the constraint nodes, as it incorporates the dense regions of the density field.

Figure 12 presents the results of an experiment using a different set of constraint nodes compared to Figure 11,



(a) Previous method [6] (b) Our method

FIGURE 12. Comparison of vector field results generated by the previous method and our method (scene2).



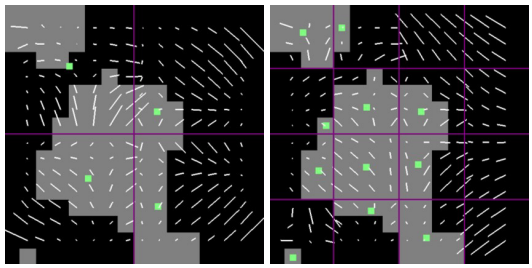
(a) 2×2 (b) 4×4

FIGURE 13. Grid-partitioning scheme for Monte Carlo weight control.

with nodes positioned at ($p1$ (0.9177, 0.397), $p2$ (0.9386, −0.3448), $p3$ (0.2805, −0.9598) and $p4$ (0.9249, 0.3802)). In the previous method, the vector field was heavily influenced by the constraint nodes, leading to cases where the vector magnitude was too small to be effectively represented, despite using divergence-constrained MLS. In contrast, the proposed method preserves vortex structures around high-density regions, preventing dissipation and ensuring a clearer representation of the vector field.

The previous results represent high-density regions as single locations, which is insufficient to fully capture the characteristics of the density field. To address this limitation, we refine the spatial resolution by dividing the domain into finer grid cells and computing Monte Carlo weights for each grid. Figure 13 illustrates the high-density regions estimated for each subdivided grid. Unlike the previous results, increasing the resolution of the grid improves the accuracy of the estimation of the high-density region, demonstrating a more precise representation of the density distribution.

Figure 14 presents the results of applying our method using different grid resolutions, with constraint nodes at ($p1$(0.0558, −0.9984), $p2$(−0.4041, 0.9147), $p3$(−0.6049, −0.7962) and $p4$(−0.7164, 0.6976)). Multiple high-density locations were estimated using the Monte Carlo method, and the weights required for MLS were determined based on the high-density locations corresponding to the node positions. Compared to the same scene in Figure 11, our method more accurately captures vortex structures and the flow of the vector field according to the density distribution.



(a) Our method (2×2) (b) Our method (4×4)

FIGURE 14. Generation of vector field with our method.

2) RESULTS WITH LEARNING REPRESENTATION

In this section, we describe the results of extracting vector fields using a learning representation. To prepare the dataset for training, we used a 128 × 128 velocity field and high-density location data as input, while the output consisted of Monte Carlo weight-based MLS vector fields. For data pre-processing, we set the stride to 14, cropping each input velocity field image to 33 × 33 and the MLS velocity field to 19 × 19. The size of the convolution filter was set to $f_1 = 9, f_2 = 1$, and $f_3 = 5$. We used the ADAM optimizer with a learning rate of 0.0003 and applied the Mean Squared Error as the loss function. The batch size was set to 128, and training was conducted for 200 epochs. The proposed method required approximately 18 minutes for training, with each epoch taking around 3∼5 seconds. The testing phase took approximately 1 second per sample.

Figure 15 presents the upscaled MLS velocity field obtained using the proposed method, showing improved detail in the velocity field. In our system, directly computing an MLS velocity field at a resolution of 256 × 256 takes between 10 and 72 seconds. In contrast, the proposed method generates results in an average of 1.2 seconds, making it suitable for real-time integration into physics-based simulations that require per-frame velocity field updates.

B. SYNTHESIS OF SMOKE TURBULENT FLOW

In this section, we present the results of integrating the turbulent flow generated using the proposed Monte Carlo weight-based MLS in a smoke simulation. In the previous method, divergence-constrained MLS was used in the advection term; however, as mentioned earlier, it results in noisy motion that differs from the underlying flow (see Figures 3 and 16).

Figure 16b shows the result of replacing semi-Lagrangian advection with MLS. Since the underlying flow structure is difficult to discern, the result appears more like noise rather than an improvement in detail. Similarly, even when using divergence-constrained MLS for advection, the result still exhibits noise-like behavior (see Figure 16c). To address this, we integrate the Monte Carlo weight-based MLS computed earlier into the advection process, as formulated

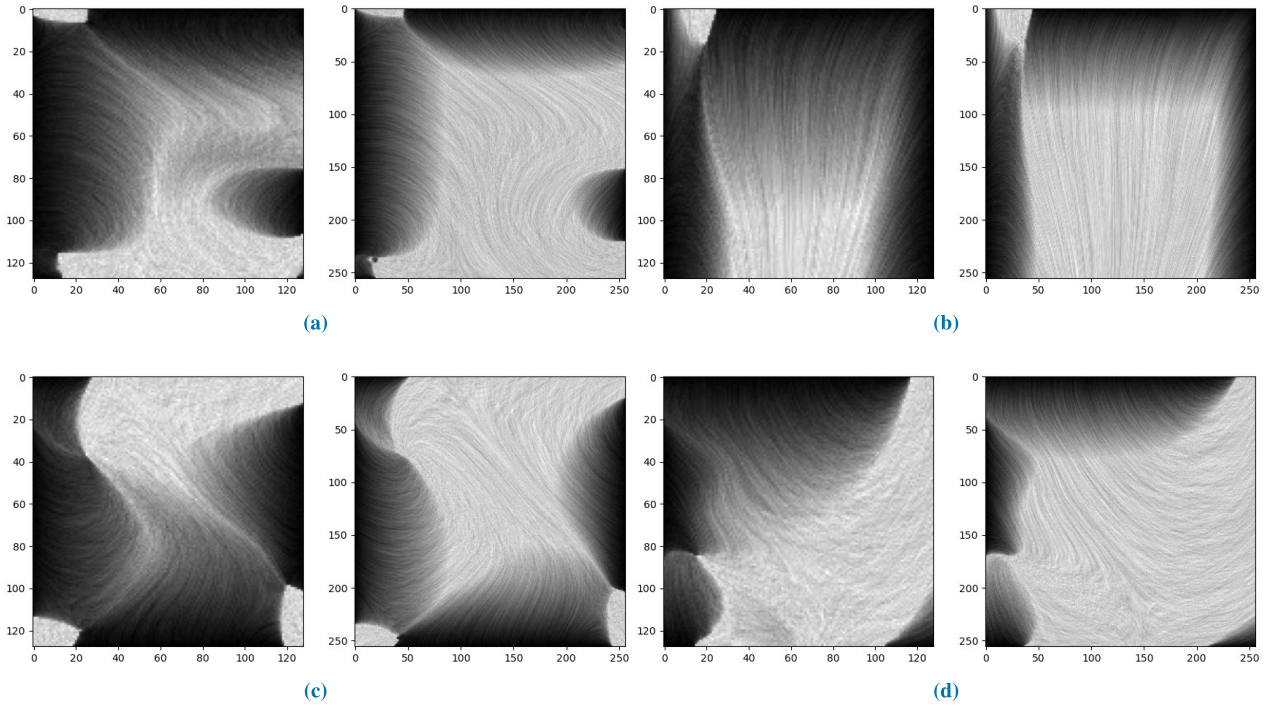
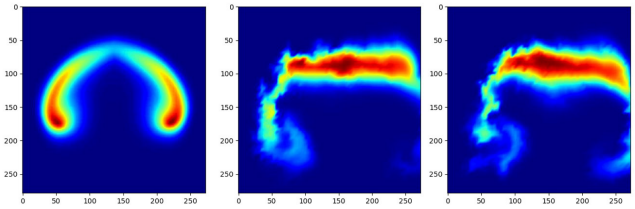


FIGURE 15. MLS vector field inferred by learning representation (left: input velocity field, right: our method).



(a) Previous method (input data) [10] (b) Previous method (only MLS) [7] (c) Previous method [7]

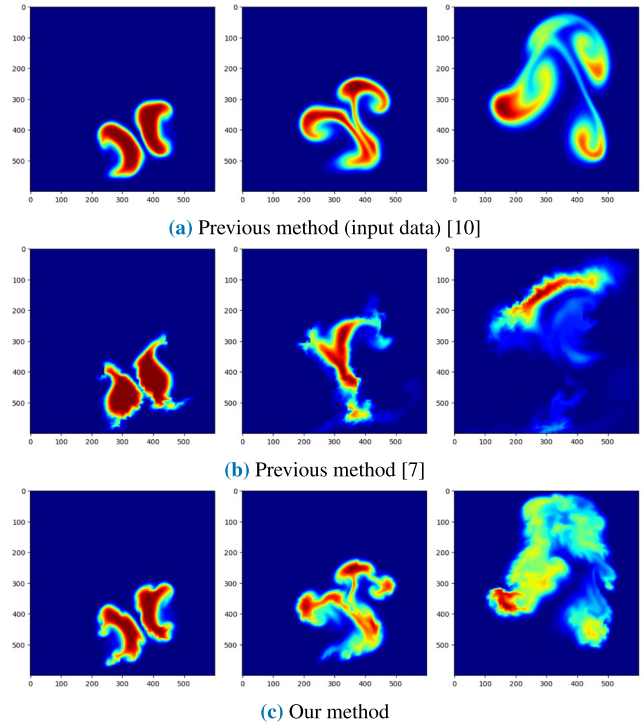
FIGURE 16. Experimental results with the previous methods.

in Equation 24.

$$\mathbf{u}\left(\overrightarrow{x},t\right)=\underbrace{\mathbf{u}\left(\overrightarrow{x},t-\Delta t\right)}_{term1}+\underbrace{\left(\mathbf{u}^{\dagger}\left(\overrightarrow{x},t\right)\Delta t\rho\alpha\right)}_{term2} \tag{24}$$

where *term*1 represents the velocity computed using semi-Lagrangian advection, while *term*2 is the velocity obtained from Monte Carlo weight-based MLS, with $\mathbf{u}^{\dagger}$ additionally incorporated. The parameter α controls the weighting of the MLS velocity field and is set to 10.0 in our experiment. To enhance the clarity of the vorticity force in the velocity field generated by our method, we apply vorticity confinement based on $\mathbf{u}^{\dagger}$ (see Equation 25). Additionally, to satisfy incompressibility, we perform a projection step.

$$\begin{aligned}\omega &= \nabla\times\mathbf{u}^{\dagger}\\ \mathbf{N} &= \frac{\eta}{|\eta|}, where\,(\eta=\nabla|\omega|)\\ \mathbf{f}_{conf} &= \epsilon^{\dagger}h\,(\mathbf{N}\times\omega)\end{aligned} \tag{25}$$



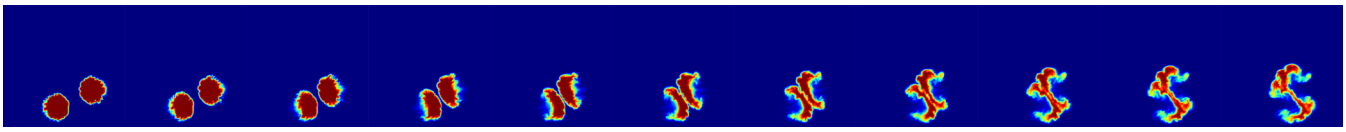
(a) Previous method (input data) [10]

(b) Previous method [7]

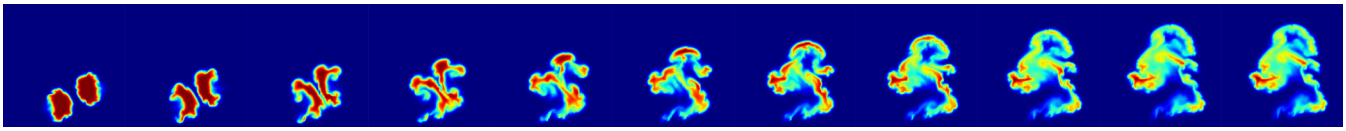
(c) Our method

FIGURE 17. Colliding smoke jets.

where $\epsilon^{\dagger} > 0$ is used to control the amount of small-scale details reintroduced into the flow field. Its dependence on spatial discretization h ensures that a physically correct



(a) Δt : 0.06



(b) Δt : 0.12

FIGURE 18. Stability test of the simulation for different time steps (capture interval: 2∼12 frames).

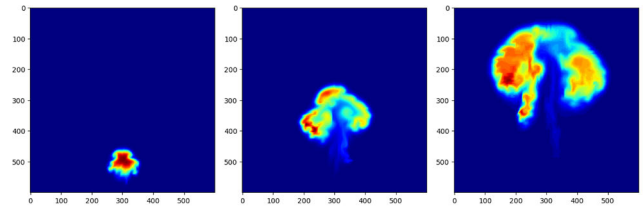


FIGURE 19. Rising smoke with our method.

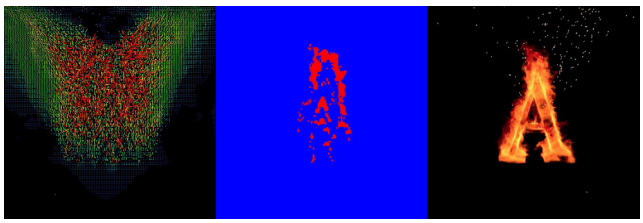
solution is maintained even as the mesh is refined. In the above equation, $\epsilon^\dagger$ is automatically determined based on $\mathbf{u}^\dagger$ as: $\frac{\hat{\mathbf{u}}\cdot\hat{\mathbf{u}}^\dagger}{2} + 0.5$. In the simulation results presented earlier, $\phi^\dagger$ was inferred using neural networks, while the remaining calculations were performed numerically.

Figure 17 illustrates the turbulent flow generated as two smoke jets collide. In the simulation of the underlying fluid without turbulent flow, the post-collision motion exhibits laminar flow characteristics, resulting in the lack of detailed flow structures (see Figure 17a). In the previous method, which applies divergence-constrained MLS in the advection stage, turbulent flow does appear. However, advection deviates from the underlying flow, leading to noticeable flickering effects and a synthesized velocity field that appears noisy. Furthermore, the simulation tends to be unstable, which requires a very small Δt or a substepping approach to maintain stability.

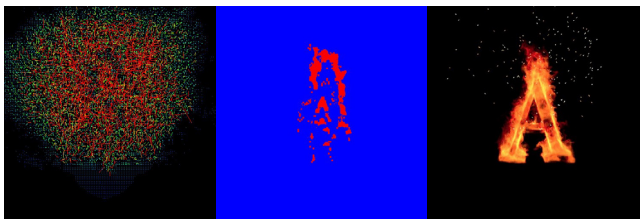
As shown in Figure 19, our method not only captures the turbulent flow generated by collisions but also effectively and stably represents turbulent flow in rising smoke.

C. SIMULATION STABILITY

In this section, we test the simulation stability in various Δt values. In the previous method, MLS-based advection generated a noise field that deviated from the underlying flow, which required a small Δt for stability. In our study, we improved advection using Monte Carlo weight-based MLS, obtaining the velocity field $\mathbf{u}^\dagger$. This was then combined with vorticity confinement to generate turbulent flow, with Δt set to 0.025. Furthermore, we tested our method without using substepping in $\Delta t = 0.06$ and $\Delta t = 0.12$, and it remained stable while effectively generating turbulent flow (see Figure 18).



(a) Velocity field (b) Flame region (c) Fire-flake



(d) Velocity field (e) Flame region (f) Fire-flake

FIGURE 20. Comparison of improved results by integrating our method into a fire-flake simulator (only previous method [23]: (a)∼(c), integration of our method into the previous method: (d)∼(f)).

D. SYNTHESIZING TURBULENT FLOW OF FIRE-FLAKE

In this section, we evaluate the extensibility of our method by applying the turbulent flow generated using Monte Carlo weight-based MLS to a fire-flake solver. As mentioned in Section I-B, Kim et al. developed a method to rapidly predict fluid flow from an input video or GIF image and compute fire-flake motion in real time (see Figure 20c).

However, since the velocity field depends on the shape of the input data, it results in a static velocity field, causing the fire-flake motion to follow a similar pattern. As shown in Figure 20a, the velocity field is inferred based on the input shape of the letter *A*, leading to a fire-flake motion that is biased in one direction (see Figure 20c). In our study, we first extracted the flame region using the previous method (see the red area in Figure 20e) and treat this region as a density field. We then apply our method to introduce turbulent flow. With our method, the velocity field, which previously

depended solely on the input shape, transforms into a more evenly distributed field flow (see Figure 20d). This change is also reflected in the fire-flake motion, where the previously unidirectional movement is improved to a more spread-out advection pattern (see Figure 20f).

E. RATIONALE FOR TIME-STEP SELECTION

The reason for setting the Δt value to 0.025 in this study is that the Monte Carlo-based MLS interpolation method and the turbulence coupling approach with the underlying flow are closely related to the numerical stability of the simulation. Traditional divergence-constrained MLS methods tend to generate high-frequency noise unrelated to the base flow in the vector field, which necessitates very small timesteps (e.g., $\Delta t < 0.01$) or mandatory substepping. However, in this study, we were able to ensure sufficient stability and turbulence expressiveness with $\Delta t = 0.025$ due to the following improvements, and thus adopted this value.

1) Rationale for Choosing $\Delta t = 0.025$
   - After interpolating the turbulence vector field based on MLS, it was combined with the underlying flow as a perturbation flow modulated by the angular difference between them. This approach effectively suppressed high-frequency components that were inconsistent with the base flow, mitigating unnecessary energy growth and enabling more stable simulations.
   - Through multiple numerical experiments, we confirmed that $\Delta t = 0.025$ provided an optimal balance that satisfied the following conditions.
     - The primary structure of the fluid is preserved while detailed turbulent flows are sufficiently captured.
     - No amplification of computational errors occurs during the interpolation and synthesizing stages.
     - Physically natural flow is maintained.
     - No drift or divergence observed during the simulation.
2) Experiments with various Δt values: In this study, we conducted simulations with multiple time-step sizes (Δt), and the results are presented in Figure 18. Specifically, the following experiments were conducted for validation.
   - $\Delta t = 0.06$: A reasonable level of turbulence was still captured, and visually stable results were generated. However, as the time-step size increased, a noticeable reduction in fine flow structures was observed.
   - $\Delta t = 0.01$: While highly stable, the excessively small time-step resulted in significantly increased computational cost, making it less practical in terms of simulation performance.

Considering these experimental results comprehensively, we determined that setting $\Delta t = 0.025$ provides the most appropriate balance between turbulence representation, computational efficiency, and numerical stability. In summary, $\Delta t = 0.025$ was chosen with careful consideration of both numerical stability and physics-based fidelity, and its validity was confirmed through extensive experimentation.

F. LIMITATION

As demonstrated earlier, the proposed method not only performs stably across various density field shapes but also more effectively represents turbulent flow based on MLS compared to the previous method. However, there are still some limitations.

1) When computing Monte Carlo weights, the key factor is identifying high-density locations. However, the accuracy of the density estimation method used for this calculation is not very high. Since most density values are less than 1, they may be lost during processing. To address this, we discretized the density field into a binary form (see Figure 8b). In this process, only density values above a certain threshold were used to determine high-density locations, meaning that flow characteristics such as density gradients and curvature were not considered. This limitation makes it challenging to directly apply our method to 3D simulations. In future work, we plan to address this issue and extend the solver to 3D. To address this limitation, we plan to incorporate the following mechanisms in future work:
   - Adaptive Thresholding: By dynamically adjusting the binarization threshold based on the distribution of density values or local variations, the accuracy of high-density region estimation can be significantly improved.
   - Convolution-Based High-Density Region Extraction Filter: To extract high-density regions more accurately, CNN-based filters can be applied to learn and predict spatial patterns within the density field. This approach allows for more precise representation of curvature and density gradients.
   - Currently, Monte Carlo sampling is performed based on a binarized density field, which does not account for higher-order features such as density gradients or curvature. In future work, we plan to address this limitation by incorporating gradient-aware weighting or curvature-based sampling techniques.
2) Since the grid partition used for computing Monte Carlo weights must be determined manually, finding the optimal resolution is challenging. Higher grid resolution increases computational cost, which requires careful selection based on the specific scenario. Although adaptive spatial partitioning could help mitigate this issue, it has not yet been considered in our current approach. To address this limitation, we plan to incorporate the following mechanisms in future work:

- Adaptive Spatial Subdivision: In regions with high variance in density distribution, finer grid cells are employed, while coarser blocks are used in more uniform regions. This strategy allows for a balance between computational efficiency and spatial accuracy.
- Learning-Based Optimal Resolution Selection: By analyzing the simulation environment or the statistical characteristics of the input density field, a predictive model can be constructed to automatically adjust the grid resolution for optimal performance.

3) Since this study employs a simple network architecture, there are limitations in making the network deeper. At this stage, we mainly tested the feasibility of using a learning representation. However, in the future, the use of a deep neural network (DNN)-based approach could enable a more optimized algorithm design. Additionally, when our method is applied to fluid simulation, incorporating grid partitioning into the learning process could help to compute an optimal Monte Carlo weight-based MLS. To address this limitation, we plan to incorporate the following mechanisms in future work:
   - Deep Network based on Residual Block: By incorporating a ResNet architecture, we aim to maintain training stability even in deeper networks and improve the precision of vector field inference.
   - Grid-aware Feature Fusion: To more effectively leverage both the input velocity field and grid partitioning information, we plan to introduce a grid-aware attention module. This will enable the network to learn Monte Carlo-based MLS interpolation with greater accuracy.

Through these improvement mechanisms, we plan to systematically address the current limitations and further enhance the accuracy and efficiency of the Monte Carlo-based MLS framework in future work.

VI. TECHNICAL CONTRIBUTION

The main contributions and novelties of this study can be summarized in the following three aspects.

1) A novel design of MLS weighting scheme that incorporates the density field
   - Traditional high-order polynomial-based Moving Least Squares (MLS) methods primarily rely on velocity vectors for interpolation. This presents a limitation in physics-based simulations, where critical physical properties such as density or temperature are not adequately reflected. To address this issue, our study introduces a novel MLS weighting scheme that is sensitive to density. Specifically, we estimate the distribution of the density field using Monte Carlo sampling and compute the weights based on the distances between these high-density locations and the simulation nodes. This represents a new approach that has not been explored in prior MLS-related research.
2) Learning representation to overcome the numerical limitations of MLS
   - Due to its reliance on high-order equations, MLS involves substantial computational overhead, making it challenging to apply in real-time simulations where frame-wise computation is required. To overcome this limitation, our study transforms the vector field generated by Monte Carlo-based MLS into a learnable form and designs an efficient prediction framework using a neural network-based super-resolution architecture. This approach significantly reduces computational cost while enabling the rapid generation of high-resolution vector fields. It represents a practical and novel extension that has not been previously explored in existing research.
3) A novel flow generation method that resolves instability and noise issues of previous approaches
   - Conventional turbulence generation methods based on divergence-constrained MLS fail to capture subgrid details, often resulting in noise and flickering artifacts. In contrast, our study proposes a novel flow generation approach that analyzes the angular difference between the underlying flow and the generated turbulent vectors and corrects it into a controlled form. This enables stable advection of the density field and maintains simulation stability even under large time steps (Δt).

As previously mentioned, this study introduces clear technical novelties that distinguish it from prior work in three key aspects: the design of MLS weights that directly incorporate the density field, the extension of MLS to a neural network-based learning representation, and the generation of corrected flow by considering the correlation between turbulence and the underlying flow.

VII. QUANTITATIVE EVALUATION METRICS

To enhance the objectivity and reproducibility of the proposed method, we consider the inclusion of quantitative evaluation metrics to be essential. Therefore, in this section, we present the following numerical indicators for quantitative assessment.

- MSE (Mean Squared Error)–Based Comparison: To quantitatively evaluate the error of the estimated MLS vector fields generated by the learning-based method, we measured the MSE between the results of the proposed method and those of the baseline approach.
  1) Experimental Conditions
     a) Resolution: 128×128, 256×256
     b) Ground Truth: Vector field computed using Monte Carlo–based MLS (numerical method)

TABLE 1. Quantitative comparison of MSE values across resolutions.

Resolution	Our method (MSE)
128×128	0.0153
256×256	0.0141

TABLE 2. Comparison between previous method and our method across different thresholds.

Threshold	Previous method [7]	Our method
0.3	0.612	0.724
0.5	0.587	0.693
0.7	0.502	0.661

c) Prediction: Vector field output by the learning-based model
d) Evaluation Region: The entire valid simulation domain

2) Summary(MSE) (see Table 1):

- IoU (Intersection over Union)–Based Density Field Comparison: To evaluate the results generated in the smoke simulation based on the overlap between density distribution images, we employed the IoU metric. This serves as an effective quantitative measure of both the accuracy and structural consistency in physics-based simulations.
  1) Experimental Conditions
     a) Thresholds = 0.3, 0.5, 0.7 (binary classification thresholds for density values)
     b) Reference Simulation: Previous MLS-based method
     c) Comparison Target: Results generated using the proposed method
  2) Summary of Results (Average IoU) (see Table 2): The proposed method achieved higher IoU values across all thresholds, indicating greater accuracy in maintaining and reproducing the shape of the density flow.

VIII. CONCLUSION

In this study, we proposed a numerical method for representing turbulent flow from a density field using MLS based on the Monte Carlo method. Traditional MLS performs high-order interpolation by considering only vector-based constraints, failing to account for the characteristics of the density field. In contrast, our method integrates Monte Carlo method-based weighting into MLS, effectively incorporating the properties of the density field in the input data and enabling the representation of various vector field structures. Furthermore, we extend the solver to allow this approach to be expressed as a learning representation through a neural network.

As demonstrated in the experimental results, the proposed method effectively and efficiently represents turbulent flow from a density field. By integrating it into smoke and fire-flake solvers, we facilitated the practical use of high-order interpolation in physics-based simulations. Compared to the previous method, our approach showed improvements in both computation time and quality, with superior performance confirmed across various scenarios. In summary, this study not only presents a method for extracting turbulent flow from a density field but also demonstrates the practicality of applying high-order interpolation in physics-based simulations.

REFERENCES

[1] C. L. Felter, J. H. Walther, and C. Henriksen, "Moving least squares simulation of free surface flows," *Comput. Fluids*, vol. 91, pp. 47–56, Mar. 2014.
[2] L. White, R. Panchadhara, and D. Trenev, "Flow simulation in heterogeneous porous media with the moving least-squares method," *SIAM J. Sci. Comput.*, vol. 39, no. 2, pp. B323–B351, Jan. 2017.
[3] Y. Zhu and S. J. Gortler, "3D deformation using moving least squares," Tech. Rep., 2007.
[4] S. Fleishman, D. Cohen-Or, and C. T. Silva, "Robust moving least-squares fitting with sharp features," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 544–552, Jul. 2005.
[5] C. L. Kang, T. N. Lu, M. M. Zong, F. Wang, and Y. Cheng, "Point cloud smooth sampling and surface reconstruction based on moving least squares," *Int. Arch. Photogramm., Remote Sens. Spatial Inf. Sci.*, vol. XLII-3, pp. 145–151, Feb. 2020.
[6] J.-M. Hong, J.-C. Yoon, and C.-H. Kim, "Divergence-constrained moving least squares for fluid simulation," *Comput. Animation Virtual Worlds*, vol. 19, nos. 3–4, pp. 469–477, Jan. 2008.
[7] S.-T. Kim and J.-M. Hong, "Visual simulation of turbulent fluids using MLS interpolation profiles," *Vis. Comput.*, vol. 29, no. 12, pp. 1293–1302, Dec. 2013.
[8] A. Cuno, C. Esperança, A. Oliveira, and P. R. Cavalcanti, "3D as-rigid-as-possible deformations using MLS," in *Proc. 27th Comput. Graph. Int. Conf.*, 2007, pp. 115–122.
[9] T. Sato, T. Igarashi, C. Batty, and R. Ando, "A long-term semi-Lagrangian method for accurate velocity advection," in *Proc. SIGGRAPH Asia Tech. Briefs*, Nov. 2017, pp. 1–4.
[10] J. Stam, "Stable fluids," in *Seminal Graphics Papers: Pushing the Boundaries*, vol. 2, 1999, pp. 121–128.
[11] B. Kim, Y. Liu, I. Llamas, and J. Rossignac, "Advections with significantly reduced dissipation and diffusion," *IEEE Trans. Vis. Comput. Graphics*, vol. 13, no. 1, pp. 135–144, Jan. 2007.
[12] O.-Y. Song, H. Shin, and H.-S. Ko, "Stable but nondissipative water," *ACM Trans. Graph.*, vol. 24, no. 1, pp. 81–97, Jan. 2005.
[13] D. Kim, O.-Y. Song, and H.-S. Ko, "A semi-Lagrangian CIP fluid solver without dimensional splitting," *Comput. Graph. Forum*, vol. 27, no. 2, pp. 467–475, Apr. 2008.
[14] S. Sato, Y. Dobashi, and T. Kim, "Stream-guided smoke simulations," *ACM Trans. Graph.*, vol. 40, no. 4, pp. 1–7, Aug. 2021.
[15] Z. Forootaninia and R. Narain, "Frequency-domain smoke guiding," *ACM Trans. Graph.*, vol. 39, no. 6, pp. 1–10, Dec. 2020.
[16] S. He, H.-C. Wong, and U.-H. Wong, "An efficient adaptive vortex particle method for real-time smoke simulation," in *Proc. 12th Int. Conf. Comput.-Aided Design Comput. Graph.*, Sep. 2011, pp. 317–324.
[17] C. Fu, Q. Guo, T. Gast, C. Jiang, and J. Teran, "A polynomial particle-in-cell method," *ACM Trans. Graph.*, vol. 36, no. 6, pp. 1–12, Dec. 2017.
[18] B. E. Feldman, J. F. O'Brien, and B. M. Klingner, "Animating gases with hybrid meshes," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 904–909, Jul. 2005.
[19] M. H. Mohammadi, F. Sotiropoulos, and J. Brinkerhoff, "Moving least squares reconstruction for sharp interface immersed boundary methods," *Int. J. Numer. Methods Fluids*, vol. 90, no. 2, pp. 57–80, May 2019.
[20] W. Li, N. Nguyen-Thanh, J. Huang, and K. Zhou, "Adaptive analysis of crack propagation in thin-shell structures via an isogeometric-meshfree moving least-squares approach," *Comput. Methods Appl. Mech. Eng.*, vol. 358, Jan. 2020, Art. no. 112613.
[21] S. Band, C. Gissler, A. Peer, and M. Teschner, "MLS pressure boundaries for divergence-free and viscous SPH fluids," *Comput. Graph.*, vol. 76, pp. 37–46, Nov. 2018.

[22] Y. Hu, Y. Fang, Z. Ge, Z. Qu, Y. Zhu, A. Pradhana, and C. Jiang, "A moving least squares material point method with displacement discontinuity and two-way rigid body coupling," *ACM Trans. Graph.*, vol. 37, no. 4, pp. 1–14, Aug. 2018.
[23] J.-H. Kim and J. Lee, "Fire sprite animation using fire-flake texture and artificial motion blur," *IEEE Access*, vol. 7, pp. 110002–110011, 2019.
[24] N. Foster and D. Metaxas, "Realistic animation of liquids," *Graph. Models Image Process.*, vol. 58, no. 5, pp. 471–483, Sep. 1996.
[25] N. Foster and D. Metaxas, "Modeling the motion of a hot, turbulent gas," in *Proc. 24th Annu. Conf. Comput. Graph. Interact. Techn. (SIGGRAPH)*, 1997, pp. 181–188.
[26] R. Fedkiw, J. Stam, and H. W. Jensen, "Visual simulation of smoke," in *Proc. 28th Annu. Conf. Comput. Graph. Interact. Techn.*, Aug. 2001, pp. 15–22.
[27] A. Selle, N. Rasmussen, and R. Fedkiw, "A vortex particle method for smoke, water and explosions," in *Proc. ACM SIGGRAPH Papers*, Jul. 2005, pp. 910–914.
[28] J. Kim, D. Cha, B. Chang, B.-K. Koo, and I. Ihm, "Practical animation of turbulent splashing water," in *Proc. Symp. Comput. Animation*, 2006, pp. 335–344.
[29] F. Losasso, J. O. Talton, N. Kwatra, and R. Fedkiw, "Two-way coupled SPH and particle level set fluid simulation," *IEEE Trans. Vis. Comput. Graphics*, vol. 14, no. 4, pp. 797–804, Jul. 2008.
[30] T. S. Greenwood and D. H. House, "Better with bubbles: Enhancing the visual realism of simulated fluid," in *Proc. ACM SIGGRAPH/Eurographics Symp. Comput. Animation*, 2004, pp. 287–296.
[31] J.-M. Hong, H.-Y. Lee, J.-C. Yoon, and C.-H. Kim, "Bubbles alive," *ACM Trans. Graph.*, vol. 27, no. 3, pp. 1–4, Aug. 2008.
[32] N. Thürey, F. Sadlo, S. Schirm, M. Müller-Fischer, and M. Groß, "Real-time simulations of bubbles and foam within a shallow water framework," in *Proc. ACM SIGGRAPH/Eurographics Symp. Comput. Animation*, 2007, pp. 191–198.
[33] Y. Gao, C.-F. Li, S.-M. Hu, and B. A. Barsky, "Simulating gaseous fluids with low and high speeds," *Comput. Graph. Forum*, vol. 28, no. 7, pp. 1845–1852, Oct. 2009.
[34] J. Hong and C. Kim, "Discontinuous fluids," *ACM Trans. Graph. (TOG)*, vol. 24, no. 3, pp. 915–920, 2005.
[35] D. Q. Nguyen, R. Fedkiw, and H. W. Jensen, "Physically based modeling and animation of fire," in *Proc. 29th Annu. Conf. Comput. Graph. Interact. Techn.*, Jul. 2002, pp. 721–728.
[36] J.-M. Hong, T. Shinar, and R. Fedkiw, "Wrinkled flames and cellular patterns," *ACM Trans. Graph.*, vol. 26, no. 99, p. 47, Jul. 2007.
[37] D. Kim, O.-Y. Song, and H.-S. Ko, "Stretching and wiggling liquids," in *Proc. ACM SIGGRAPH Asia Papers*, Dec. 2009, pp. 1–7.
[38] T. Kim, N. Thürey, D. James, and M. Gross, "Wavelet turbulence for fluid simulation," *ACM Trans. Graph.*, vol. 27, no. 3, pp. 1–6, Aug. 2008.
[39] R. Narain, J. Sewall, M. Carlson, and M. C. Lin, "Fast animation of turbulence using energy transport and procedural synthesis," *ACM Trans. Graph.*, vol. 27, no. 5, pp. 1–8, Dec. 2008.
[40] H. Schechter and R. Bridson, "Evolving sub-grid turbulence for smoke animation," in *Proc. ACM SIGGRAPH/Eurographics Symp. Comput. Animation*, 2008, pp. 1–7.
[41] M. Chu and N. Thuerey, "Data-driven synthesis of smoke flows with CNN-based feature descriptors," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 1–14, Aug. 2017.
[42] S. Sato, Y. Dobashi, T. Kim, and T. Nishita, "Example-based turbulence style transfer," *ACM Trans. Graph.*, vol. 37, no. 4, pp. 1–9, Aug. 2018.
[43] X.-S. Chen, C.-F. Li, G.-C. Cao, Y.-T. Jiang, and S.-M. Hu, "A moving least square reproducing kernel particle method for unified multiphase continuum simulation," *ACM Trans. Graph.*, vol. 39, no. 6, pp. 1–15, Dec. 2020.
[44] R. Sawhney and K. Crane, "Monte Carlo geometry processing: A grid-free approach to PDE-based methods on volumetric domains," *ACM Trans. Graph.*, vol. 39, no. 4, Aug. 2020.
[45] Y. Xu, T. Fan, Y. Yuan, and G. Singh, "Ladybird: Quasi-Monte Carlo sampling for deep implicit field based 3D reconstruction with symmetry," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 248–263.
[46] L. Ladický, S. Jeong, B. Solenthaler, M. Pollefeys, and M. Gross, "Data-driven fluid simulations using regression forests," *ACM Trans. Graph.*, vol. 34, no. 6, pp. 1–9, Nov. 2015.
[47] J. Tompson, K. Schlachter, P. Sprechmann, and K. Perlin, "Accelerating Eulerian fluid simulation with convolutional networks," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 3424–3433.
[48] C. Yang, X. Yang, and X. Xiao, "Data-driven projection method in fluid simulation," *Comput. Animation Virtual Worlds*, vol. 27, nos. 3–4, pp. 415–424, May 2016.
[49] B. Kim, V. C. Azevedo, N. Thuerey, T. Kim, M. Gross, and B. Solenthaler, "Deep fluids: A generative network for parameterized fluid simulations," *Comput. Graph. Forum*, vol. 38, no. 2, pp. 59–70, May 2019.
[50] N. Umetani and B. Bickel, "Learning three-dimensional flow for interactive aerodynamic design," *ACM Trans. Graph.*, vol. 37, no. 4, pp. 1–10, Aug. 2018.
[51] Y. Xie, A. Franz, M. Chu, and N. Thuerey, "TempoGAN: A temporally coherent, volumetric GAN for super-resolution fluid flow," *ACM Trans. Graph.*, vol. 37, no. 4, pp. 1–15, Aug. 2018.
[52] K. Bai, W. Li, M. Desbrun, and X. Liu, "Dynamic upsampling of smoke through dictionary-based learning," *ACM Trans. Graph.*, vol. 40, no. 1, pp. 1–19, Feb. 2021.
[53] K. Bai, C. Wang, M. Desbrun, and X. Liu, "Predicting high-resolution turbulence details in space and time," *ACM Trans. Graph.*, vol. 40, no. 6, pp. 1–16, Dec. 2021.
[54] P. Holl, V. Koltun, and N. Thuerey, "Learning to control PDEs with differentiable physics," 2020, *arXiv:2001.07457*.
[55] Y. Hu, J. Liu, A. Spielberg, J. B. Tenenbaum, W. T. Freeman, J. Wu, D. Rus, and W. Matusik, "ChainQueen: A real-time differentiable physical simulator for soft robotics," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 6265–6271.
[56] Y. Hu, X. Zhang, M. Gao, and C. Jiang, "On hybrid Lagrangian–Eulerian simulation methods: Practical notes and high-performance aspects," in *Proc. ACM SIGGRAPH Courses*, Jul. 2019, pp. 1–246.
[57] C. Schenck and D. Fox, "SPNets: Differentiable fluid dynamics for deep neural networks," in *Proc. Conf. Robot Learn.*, 2018, pp. 317–335.
[58] E. Franz, B. Solenthaler, and N. Thuerey, "Global transport for fluid reconstruction with learned self-supervision," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 1632–1642.
[59] S. Li, X. Xu, L. Nie, and T.-S. Chua, "Laplacian-steered neural style transfer," in *Proc. 25th ACM Int. Conf. Multimedia*, Oct. 2017, pp. 1716–1724.
[60] J. Han and C. Wang, "TSR-VFD: Generating temporal super-resolution for unsteady vector field data," *Comput. Graph.*, vol. 103, pp. 168–179, Apr. 2022.
[61] G. Yan, Z. Chen, J. Yang, and H. Wang, "Interactive liquid splash modeling by user sketches," *ACM Trans. Graph.*, vol. 39, no. 6, pp. 1–13, Dec. 2020.
[62] B. Kim, X. Huang, L. Wuelfroth, J. Tang, G. Cordonnier, M. Gross, and B. Solenthaler, "Deep reconstruction of 3D smoke densities from artist sketches," *Comput. Graph. Forum*, vol. 41, no. 2, pp. 97–110, May 2022.
[63] M. Chu, N. Thuerey, H.-P. Seidel, C. Theobalt, and R. Zayer, "Learning meaningful controls for fluids," *ACM Trans. Graph.*, vol. 40, no. 4, pp. 1–13, Aug. 2021.
[64] J. Aurand, R. Ortiz, S. Nauer, and V. C. Azevedo, "Efficient neural style transfer for volumetric simulations," *ACM Trans. Graph.*, vol. 41, no. 6, pp. 1–10, Dec. 2022.
[65] A. Huerta, Y. Vidal, and P. Villon, "Pseudo-divergence-free element free Galerkin method for incompressible fluid flow," *Comput. Methods Appl. Mech. Eng.*, vol. 193, nos. 12–14, pp. 1119–1136, Mar. 2004.
[66] C. Dong, C. C. Loy, K. He, and X. Tang, "Pseudo-divergence-free element free Galerkin method for incompressible fluid flow," *Comput. Methods Appl. Mech. Eng.*, vol. 193, nos. 12–14, pp. 1119–1136, 2004.