Fast VEM Fluid Simulation

RUNZE ZHANG, Nankai University, China
BO REN*, Nankai University, China

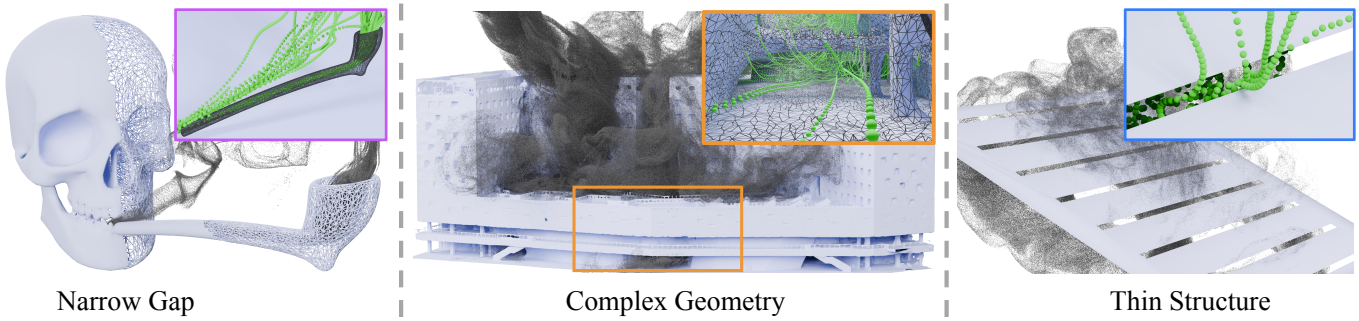


Fig. 1. FastVEM provides an efficient boundary-conforming fluid simulation framework that robustly handles narrow gaps (left), complex geometry reconstructed from drone scans (middle), and ultra-thin sheet boundary with a relative thickness of 10^{-5} (right). Embedding these challenging geometries into a 128^3 Cartesian grid, high-fidelity smoke simulation using a second-order VEM discretization for the pressure projection requires less than 1 min/frame. Smoke trajectories are visualized as green particles in the zoomed-in views; the middle inset highlights smoke motion near the interior base of the building.

The intricate motion arising from fluid–boundary interactions is visually compelling, yet notoriously difficult and computationally expensive to simulate in the presence of complex boundaries. Accurately resolving boundary geometry requires body-fitted grids constructed via cut-cell methods, which often leads to poorly conditioned linear systems in the pressure projection stage and, consequently, prohibitive computational cost. We present *FastVEM*, an efficient boundary-conforming fluid simulation framework that enables high-fidelity flow–boundary interaction at substantially reduced cost. Computational efficiency is achieved through a coordinated, top-down design spanning numerical discretization, grid construction, and linear solvers. FastVEM adopts a Virtual Element Method (VEM) discretization to robustly enforce incompressibility and boundary conditions on irregular body-fitted grids, and employs a VEM polynomial-space Particle-in-Cell scheme for advection. Complementing this discretization, a convexity-preserving cut-cell strategy is introduced to construct simulation-friendly body-fitted grids. To accelerate pressure projection, we develop a Galerkin geometric multigrid solver featuring a diffusion-free prolongation operator that prevents coarse-level matrix densification, along with a nested, boundary-aware grid hierarchy that ensures well-posed placement of coarse-level degrees of freedom. Compared to prior cut-cell–based fluid simulators, FastVEM speeds up the computationally dominant pressure projection stage by up to 100×, while robustly handling even more challenging boundary geometries.

CCS Concepts: • **Computing methodologies** → **Physical simulation**.

ACM Reference Format:
Runze Zhang and Bo Ren. 2026. Fast VEM Fluid Simulation. *ACM Trans. Graph.* 45, 4, Article 65 (July 2026), 18 pages. https://doi.org/10.1145/3811315

*Corresponding author: Bo Ren (rb@nankai.edu.cn)

Authors' Contact Information: Runze Zhang, oliverzrz.cyber@gmail.com, VCIP, College of Computer Science, Nankai University, TianJin, China; Bo Ren, rb@nankai.edu.cn, VCIP, College of Computer Science, Nankai University, TianJin, China.

ACM 1557-7368/2026/7-ART65
https://doi.org/10.1145/3811315

1 Introduction

Ensuring that fluid motion conforms to geometrically complex boundaries is essential for producing physically plausible animations. Despite substantial progress, designing an efficient boundary-conforming fluid simulator remains challenging for two primary reasons. First, Cartesian grid–based simulators achieve high computational efficiency, but struggle with non–grid-aligned boundaries. Even when augmented with area-fraction techniques, such methods remain limited in accurately capturing complex boundary geometry within a single grid cell. Increasing the grid resolution can mitigate this limitation, but quickly leads to impractical memory and computational costs. Second, body-fitted grid–based fluid simulation enables accurate capture of complex geometries. However, the resulting irregular body-fitted grids often lead to poorly conditioned linear systems, posing a significant challenge for the efficient enforcement of incompressibility and boundary conditions.

Existing works on boundary-conforming fluid simulation generally follow two directions. Adaptive resolution methods [Aanjaneya et al. 2017; Ando and Batty 2020; Losasso et al. 2004; Zhang et al. 2016] employ local grid refinement near boundaries to avoid a global increase in grid resolution. While such approaches capture finer boundary features at reduced cost, approximating arbitrary boundary surfaces with axis-aligned grids remains inherently inflexible: accurately resolving even a single non-axis-aligned plane can trigger aggressive local refinement and thus a substantial increase in the number of degrees of freedom. Alternatively, cut-cell–based methods address this limitation by simulating fluids on body-fitted grids that conform directly to boundaries [Azevedo et al. 2016; Chen et al. 2020; Tao et al. 2022; Zarifi and Batty 2017]. These methods offer higher geometric fidelity; however, accurately resolving fine-scale structures often leads to irregular grids with poorly shaped cut cells, thereby increasing the cost of enforcing incompressibility and boundary conditions. Efficient multigrid solvers can alleviate this issue; however, constructing multigrid methods on body-fitted

grids remains challenging. Non–Galerkin approaches struggle to ensure consistent boundary condition transfer across multigrid levels, while Galerkin projection on irregular grids tends to induce connectivity proliferation among degrees of freedom, leading to dense coarse–level system matrices [Xian et al. 2019].

Following the cut-cell paradigm, we propose *FastVEM*, an efficient boundary-conforming fluid simulation framework. FastVEM adopts a carefully coordinated, top-down design spanning numerical discretization, grid construction, and linear system solvers. At the numerical level, FastVEM employs a second-order Virtual Element Method (i.e., VEM of order k = 2) for pressure discretization to accurately enforce boundary conditions on body-fitted grids. For grid construction, a convexity-preserving, binary space partition–based cut-cell strategy is introduced to mitigate poorly conditioned system matrices caused by non–star-shaped elements. Finally, pressure projection is accelerated by a geometric multigrid solver featuring a diffusion-free prolongation operator to prevent coarse-level matrix densification, together with a nested, boundary-aware grid hierarchy that ensures well-posed placement of coarse-level degrees of freedom. Together, VEM discretization delivers the accuracy and robustness needed for boundary handling, while the proposed cut-cell strategy and multigrid method alleviate the otherwise prohibitive computational cost of VEM schemes. With this balance, FastVEM robustly supports a wide range of boundaries and achieves up to two orders of magnitude speedup over prior cut-cell–based fluid solvers in the computationally dominant pressure projection stage.

FastVEM extends the capabilities of cut-cell fluid simulators, making boundary-conforming fluid simulation more affordable. Algorithm 1 provides an overview of the proposed framework, and the main contributions are summarized as follows:

- A VEM-based cut-cell fluid simulator with second-order pressure discretization for accurate enforcement of boundary conditions, speeding up the computationally dominant pressure projection stage by up to 100× over existing cut-cell–based fluid simulators.
- A geometric multigrid method for VEM Poisson problems uses a Galerkin formulation to ensure consistent transfer of boundary conditions across levels, and a diffusion-free prolongation operator to avoid coarse-level system matrix densification caused by Galerkin projection.
- A binary space partitioning–based cut-cell strategy for constructing simulation-friendly body-fitted grids, together with a nested, boundary-aware grid hierarchy construction strategy that supports the proposed multigrid method.

The code for this paper is located at https://oliver-zrz-cyber.github.io/FastVEM/.

2 Related Work

Boundary-Conforming Fluid Simulation. Accurate conformity between fluid motion and complex immersed geometry is crucial for physically plausible flows. Existing boundary-conforming fluid solvers can be broadly categorized into three classes. The first class of methods enhances boundary fidelity by locally adapting grid resolution on top of a Cartesian grid. Representative approaches include adaptive octree structures that resolve fine-scale details [Losasso et al. 2004], power-diagram–based methods that avoid T-junction artifacts while achieving second-order accurate pressure projection near boundaries [Aanjaneya et al. 2017], and surface-adaptive octree frameworks designed for efficient and robust liquid simulation [Ando and Batty 2020]. Within this category, other variants increase local resolution by overlaying high-resolution grids on a coarse background [Zhang et al. 2016], or by coupling boundary-guided multiresolution grids with parametric boundary treatments to better resolve turbulent flows near solid surfaces [Liu and Liu 2023]. More recently, overset grid techniques have been extended to support moving, multi-resolution domains with explicit boundary layer control, enabling more accurate two-phase fluid–rigid interactions [Xiao et al. 2025]. The second class of methods preserves the underlying Cartesian grid and incorporates boundary information directly into the pressure solve by modifying the discretization stencils used to assemble the system matrix. Representative examples include ghost-fluid formulations that achieve second-order accurate free-surface pressure conditions by altering finite-difference gradient stencils [Enright et al. 2003], as well as variational approaches that handle irregular boundaries through divergence stencils augmented with volume fractions [Batty et al. 2007]. Subsequent improvements replace volume fractions with area fractions in finite-volume cut-cell discretizations to achieve second-order pressure accuracy [Ng et al. 2009], and further improve efficiency by integrating multigrid solvers [Weber et al. 2015]. Despite their simplicity and compatibility with Cartesian grids, these methods remain limited in accurately resolving complex boundary geometries. The third class of methods comprises cut-cell–based fluid solvers, which construct body-fitted grids by explicitly cutting a Cartesian grid with a boundary mesh, thereby enabling discretizations that naturally conform to complex geometry. Early work employs fixed tetrahedral meshes near boundaries to improve geometric fidelity [Batty et al. 2010]. Subsequent approaches introduce adaptive discontinuous Galerkin schemes on body-fitted grids to capture detailed free-surface dynamics even on coarse grids [Edwards and Bridson 2014]. Finite-volume cut-cell formulations further advance this line of work by improving pressure projection and velocity interpolation near thin boundaries [Azevedo et al. 2016], as well as enabling robust two-way coupling through positive-definite system construction [Zarifi and Batty 2017]. By assigning multiple virtual pressure samples per cell with a single physical degree of freedom, this formulation extends to support sub-grid free-surface structures, achieving second-order accuracy for pressure values while preserving the symmetric positive definite structure [Chen et al. 2020]. Related ideas have also been explored in the lattice Boltzmann framework, where sample-based cut-cell techniques are used to resolve sub-voxel geometric features [Lyu et al. 2021]. More recent work, VEMPIC [Tao et al. 2022], combines Mandoline [Tao et al. 2019] for constructing body-fitted grids with a non-conforming virtual element method for pressure projection, enabling robust fluid interaction with complex boundaries at a high computational cost. To enforce incompressibility and boundary conditions, our work likewise employs the virtual element method, but adopts a conforming formulation to avoid discontinuities in the pressure gradient across cell interfaces that are inherent to non-conforming schemes.

Multigrid Methods. Enforcing incompressibility entails solving large linear systems, which typically dominate the computational cost of split-based fluid solvers. To mitigate this bottleneck, multigrid methods are widely adopted and have a long history of efficiently solving the Poisson equations arising in fluid simulation, initially on regular Cartesian grids via geometric multigrid preconditioning of conjugate gradient solvers [McAdams et al. 2010]. This line of work was subsequently extended to support irregular boundaries by accelerating volume-fraction–based variational formulations [Chentanez and Mueller-Fischer 2012]. Later efforts adapted multigrid solvers to hybrid Cartesian discretizations, including tall-grid representations [Chentanez and Müller 2011] and sparsely populated grids with octree-style adaptive refinement [Setaluri et al. 2014]. More specialized developments further tailor multigrid techniques to fraction-based discretizations [Weber et al. 2015], introduce topology-aware coarsening to improve convergence in the presence of thin solid features [Dick et al. 2016], or design dedicated smoothers for efficiently solving viscosity systems [Aanjaneya et al. 2019], and introduce adaptive smoothers to enable multiresolution adaptive grids for pressure correction in two-phase flows with large density contrasts [Braun et al. 2025]. Despite these advances, existing geometric multigrid methods are not designed to operate directly on body-fitted grids. In contrast, algebraic multigrid methods make no assumptions about geometric structure, allowing them to accelerate sparse linear systems arising from discretizations on body-fitted grids [Demidov 2020; Shao et al. 2022; Zarifi 2020]. In the context of elastic simulation, Galerkin multigrid methods have been widely applied to tetrahedral mesh–based discretizations [Briggs et al. 2000]. For example, Tiantian et al. [Xian et al. 2019] define prolongation operators using piecewise constant interpolation to avoid dense coarse-level matrices on tetrahedral meshes, while Liu et al. [Lu et al. 2025] combine the Full Approximation Scheme with block Jacobi smoothers to construct efficient multigrid solvers for such discretizations. Despite their success, these methods cannot be directly applied to body-fitted fluid simulation. Beyond graphics, two works are most closely related to ours. The first proposes a p-multigrid method for VEM discretizations [Antonietti et al. 2018], performing coarse-grid correction by switching to lower-order VEM spaces rather than coarsening the volumetric grid. The second develops a geometric multigrid framework for first-order two-dimensional VEM [Antonietti et al. 2023], defining the prolongation operator directly from the VEM degrees of freedom and providing theoretical convergence guarantees. To the best of our knowledge, we present the first geometric multigrid method for second-order VEM in three dimensions.

Numerical Discretization on Irregular Grids. Irregular grids offer a flexible representation for complex boundary geometries, motivating extensive research on solving PDEs over arbitrarily shaped cells. Finite volume methods [Eymard et al. 2000] naturally extend to general polyhedral grids, but often involve complex stencils and may lose symmetry or positive definiteness, leading to reduced accuracy on highly irregular grids. Discontinuity-based methods, such as Discontinuous Galerkin [Cockburn et al. 2009], Weak Galerkin [Wang and Ye 2013], and Hybrid High-Order methods [Pietro et al. 2014], relax inter-element continuity and enable robust discretizations on general polyhedral grids through the use of discontinuous skeletal structures. Polyhedral finite element methods [Sukumar and Tabarraei 2004] generalize classical finite elements to polygonal and polyhedral cells, but rely on explicit rational basis functions whose evaluation requires high-order numerical quadrature, resulting in significant computational cost. The Mimetic Finite Difference (MFD) method [Brezzi et al. 2009] bridges finite differences and finite elements by constructing discrete variational formulations directly on degrees of freedom, enabling stable and consistent discretizations on general grids without explicit shape functions. Building on this mimetic philosophy, the Virtual Element Method (VEM) [Beirão da Veiga et al. 2013] introduces a Galerkin framework with well-defined approximation spaces and projector-based bilinear forms, overcoming key limitations of MFD in nonlinear problems and theoretical analysis while retaining flexibility on general polyhedral grids.

ALGORITHM 1: FastVEM

Precomputation stage:

1. Construct a body-fitted grid $\mathcal{P}^\ell$ from the given boundary mesh (Section 5.1);
2. Build a grid hierarchy $\{\mathcal{P}^0, \ldots, \mathcal{P}^{\ell-1}\}$ for the proposed multigrid method (Section 5.2);
3. Assemble the prolongation operators $\mathbf{P}$ and construct the multilevel system matrices via $\mathbf{A}_c = \mathbf{R}\,\mathbf{A}_f\,\mathbf{P}$ (Section 4);

A simulation step:

4. *Particle-to-grid transfer:* Velocity degrees of freedom at vertices of P^l are computed by accumulating contributions from particles in the local neighborhood (Section 3.2);
5. *VEM-based pressure projection:* Assemble and solve the VEM Poisson system to obtain the pressure field (Section 3.1);
6. *Grid-to-particle transfer and particle advection:* Apply the pressure field to the polynomial velocity space, update particle velocities, and advect particles (Section 3.3);

3 VEMFLUID

We approximately solve the incompressible, inviscid fluid governed by the following Euler equations:

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p &= \mathbf{f} && \text{in } \Omega, \\ \nabla\cdot\mathbf{u} &= 0 && \text{in } \Omega, \\ \mathbf{u}\cdot\mathbf{n} &= 0 && \text{on } \partial\Omega, \end{aligned} \tag{1}$$

where $\mathbf{u}$ denotes the velocity field, p the pressure, $\mathbf{f}$ the external force, $\mathbf{n}$ the outward unit normal on $\partial\Omega$, and Ω the fluid domain. The fluid density is set to one. Enforcing incompressibility and boundary conditions on body-fitted grids requires a discretization that remains stable and consistent over general polyhedral grids. We employ a second-order Virtual Element Method (VEM) for pressure projection, as it provides a theoretically grounded and practically mature framework for discretizing elliptic operators on such grids [Beirão da Veiga et al. 2013]. In contrast to [Tao et al. 2022], we adopt a conforming formulation to avoid inter-cell discontinuities and to facilitate efficient multigrid solver design. Conforming and non-conforming VEM differ fundamentally in their degrees of freedom (DOFs): the former enforces inter-element continuity through nodal, edge, face,

and volume DOFs, yielding globally continuous approximations, whereas the latter relies on face- and volume-based moment DOFs that only weakly enforce continuity without guaranteeing pointwise consistency. This difference allows conforming VEM to avoid inter-cell discontinuities in the pressure field and to achieve more consistent velocity behavior across cell interfaces. As shown in Section 4, conforming VEM also yields a DOF structure that supports the design of efficient diffusion-free multigrid solvers. For a detailed discussion of conforming and non-conforming VEM, see [Cangiani et al. 2016]. The material derivative $D\mathbf{u}/Dt$ is discretized using a standard FLIP formulation to reduce numerical dissipation, in which velocities are stored on particles and advected through a grid-based velocity field. Throughout the paper, for a scalar field f, we use f_h to denote its virtual element approximation and $\vec{f}$ to represent the vector of Lagrange-type coefficients associated with f_h at DOFs. The same notation applies analogously to vector fields.

3.1 VEM-Based Pressure Projection Preliminaries

To keep the paper self-contained, this section briefly summarizes the VEM formulation used for pressure projection; more detailed discussions can be found in [Beirão Da Veiga et al. 2023].

The pressure projection can be formulated as a Poisson problem with Neumann boundary conditions:

$$\begin{aligned} \Delta p &= \frac{1}{\Delta t}\,\nabla\cdot\mathbf{u}' && \text{in } \Omega,\\ \frac{\partial p}{\partial \mathbf{n}} &= \frac{1}{\Delta t}\,\mathbf{u}'\cdot\mathbf{n} && \text{on } \partial\Omega. \end{aligned} \tag{2}$$

where $\mathbf{u}'$ denotes the intermediate velocity field. The corresponding variational formulation is: find p such that

$$\int_\Omega \nabla p\cdot\nabla v\,\mathrm{d}x = \frac{1}{\Delta t}\int_\Omega \mathbf{u}'\cdot\nabla v\,\mathrm{d}x, \qquad \forall v\in V, \tag{3}$$

where V is a suitable function space. Let V_k be the virtual element space of order k, spanned by the basis functions $\{\varphi_i^k\}_{i=1}^{N_{\mathrm{dof}}^k}$, which are implicitly defined through the degrees of freedom (DOFs) $\{D_i^k\}$ by the duality condition:

$$D_j^k(\varphi_i^k) = \delta_{ij}, \qquad i,j = 1,\dots,N_{\mathrm{dof}}^k. \tag{4}$$

Here, N_{dof}^k denotes the total number of DOFs associated with the virtual element space V_k. As a consequence, any function $v_h^k\in V_k$ admits a Lagrange-type interpolation representation,

$$v_h^k = \sum_{i=1}^{N_{\mathrm{dof}}^k} D_i^k(v_h^k)\,\varphi_i^k. \tag{5}$$

When no ambiguity arises, we omit the superscript k indicating the order of the virtual element space. To more accurately capture the interaction between the fluid and the boundary, FastVEM employs second-order pressure to enforce both the boundary conditions and the incompressibility of a linear velocity field. This leads us to seek the pressure solution in the V_2 space, resulting in the following discrete problem: find $p_h\in V_2$ such that

$$\int_\Omega \nabla p_h\cdot\nabla\varphi_i\,\mathrm{d}x = \frac{1}{\Delta t}\int_\Omega \mathbf{u}_h'\cdot\nabla\varphi_i\,\mathrm{d}x, \qquad i=1,\dots,N_{\mathrm{dof}}, \tag{6}$$

where $\{\varphi_i\}_{i=1}^{N_{\mathrm{dof}}}$ are the basis functions spanning V_2.

To solve the discrete variational problem, the global stiffness matrix K and the right-hand-side vector $\vec{r}$ need to be computed:

$$K_{ij} = \int_\Omega \nabla\varphi_i\cdot\nabla\varphi_j\,\mathrm{d}x, \qquad i,j=1,\dots,N_{\mathrm{dof}}, \tag{7}$$

$$\vec{r}_i = \frac{1}{\Delta t}\int_\Omega \mathbf{u}_h'\cdot\nabla\varphi_i\,\mathrm{d}x, \qquad i=1,\dots,N_{\mathrm{dof}}. \tag{8}$$

The global stiffness matrix K is assembled from the element-wise stiffness matrices K^E, and the right-hand-side vector $\vec{r}$ is computed as described in Section 3.2. Once K and $\vec{r}$ are assembled, the DOF vector $\vec{p}$ of the discrete pressure field p_h is obtained by solving

$$K\,\vec{p} = \vec{r}. \tag{9}$$

In the finite element method (FEM), the element stiffness matrix K^E is computed by integrating explicitly defined polynomial basis functions over each element. VEM extends FEM to support general polyhedral grids by introducing non-polynomial functions in the local virtual element spaces. As a consequence, the basis functions cannot be constructed explicitly. Rather than evaluating the stiffness matrix via explicitly constructed basis functions, VEM is formulated so that, for any functions $v_h, u_h\in V_k$, the bilinear form $\int_E \nabla v_h\cdot\nabla u_h\,\mathrm{d}x$ can be computed directly from the DOFs of v_h and u_h.

The key idea behind computing K_{ij}^E in VEM is to project virtual basis functions onto a polynomial space. Specifically, for a polyhedral cell E, let $V_k(E)$ denote the local virtual element space of order k on E. VEM defines the projection operator $\Pi_k^\nabla : V_k(E)\to\mathbb{P}_k(E)\subset V_k(E)$, where $\mathbb{P}_k(E)$ is the space of three-dimensional polynomials of degree at most k on each element E. The operator Π_k^∇ is defined element-wise for any $v_h\in V_k(E)$ by the orthogonality conditions

$$\int_E \nabla p\cdot\nabla\left(\Pi_k^\nabla v_h - v_h\right)\mathrm{d}x = 0, \qquad \forall\, p\in\mathbb{P}_k(E), \tag{10}$$

along with the additional constraint

$$P_0\left(\Pi_k^\nabla v_h - v_h\right) = 0. \tag{11}$$

The operator P_0 fixes the null space of the orthogonality conditions and is defined as

$$P_0 v_h := \begin{cases} \dfrac{1}{N_D}\displaystyle\sum_{i=1}^{N_D} D_i(v_h), & k=1,\\[2ex] \dfrac{1}{|E|}\displaystyle\int_E v_h\,\mathrm{d}x, & k\geq 2, \end{cases} \tag{12}$$

where $D_i(v_h)$ denotes the i-th degree of freedom of v_h. By definition of the projection, $\Pi_k^\nabla v_h$ admits a polynomial expansion

$$\Pi_k^\nabla v_h = \sum_{\alpha=1}^{N_k} s_\alpha^h\, m_\alpha, \tag{13}$$

where $\{m_\alpha\}$ is a set of three-dimensional polynomials of degree less than or equal to k, and $N_k = \dim\mathbb{P}_k(E)$. The construction of the polynomial basis $\{m_\alpha\}$ is detailed in Appendix A.

Projecting the virtual basis functions onto the polynomial space amounts to computing the coefficients s_α^h in Eq. 13. Substituting

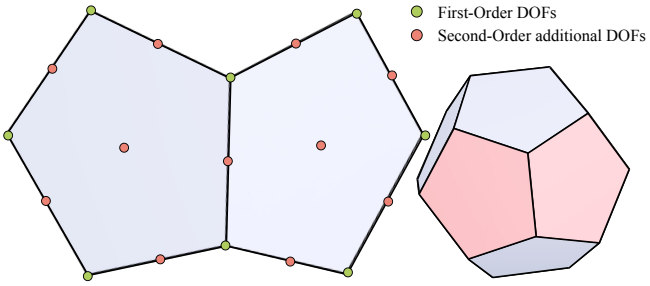


Fig. 2. In the linear setting, VEM places degrees of freedom at grid vertices; in the second-order formulation, additional degrees of freedom are introduced on edges, faces, and within cells.

this expansion into the orthogonality conditions in Eq. 10 and the constraint in Eq. 11 yields the following linear system:

$$\sum_{\alpha=1}^{N_k} s_\alpha^h \int_E \nabla m_\alpha \cdot \nabla m_\beta \,\mathrm{d}x = \int_E \nabla v_h \cdot \nabla m_\beta \,\mathrm{d}x, \quad \beta = 1, \ldots, N_k,$$
$$\sum_{\alpha=1}^{N_k} s_\alpha^h P_0(m_\alpha) = P_0(v_h). \tag{14}$$

The resulting linear system involves only polynomial quantities on the left-hand side and can therefore be evaluated exactly. To render the right-hand side computable without constructing v_h, the virtual element method defines the DOFs of $v_h \in V_k(E)$ as follows:

(1) *Vertex values*: the values of v_h at the vertices of the polyhedral element E.
(2) *Edge values*: on each edge $e \subset \partial E$, the values of v_h at the $k-1$ internal points of the $(k+1)$-point Gauss–Lobatto quadrature.
(3) *Face moments*: for each face $f \subset \partial E$, the moments of v_h over f up to order $k-2$,
$$\frac{1}{|f|}\int_f v_h \, m_\alpha^f \,\mathrm{d}s, \qquad \alpha = 1, \ldots, N_{k-2}^f.$$
(4) *Cell moments*: the moments of v_h over E up to order $k-2$,
$$\frac{1}{|E|}\int_E v_h \, m_\alpha \,\mathrm{d}x, \qquad \alpha = 1, \ldots, N_{k-2}.$$

Here, $|f|$ and $|E|$ denote the area of f and the volume of E, respectively, and m_α^f denotes the two-dimensional counterpart of m_α. Moreover, N_{k-2} and N_{k-2}^f denote the numbers of basis polynomials of degree at most $k-2$ in the three-dimensional and two-dimensional settings, respectively. An intuitive illustration of the DOF placement is shown in Fig. 2. With the above definition of DOFs, the right-hand side terms can either be directly queried from the DOFs of v_h or computed using the DOFs via integration by parts. Consequently, the element-wise linear system corresponding to Eq. (14) can be established and solved to obtain the projection operator Π_k^∇, which maps virtual functions onto the polynomial space.

Using the projector Π_k^∇, the basis function φ_i is decomposed as

$$\varphi_i = \Pi_k^\nabla \varphi_i + (I - \Pi_k^\nabla)\varphi_i. \tag{15}$$

The element stiffness matrix can then be written as

$$K_{ij}^E = \int_E \nabla\Pi_k^\nabla \varphi_i \cdot \nabla\Pi_k^\nabla \varphi_j \,\mathrm{d}x + \int_E \nabla(I-\Pi_k^\nabla)\varphi_i \cdot \nabla(I-\Pi_k^\nabla)\varphi_j \,\mathrm{d}x. \tag{16}$$

where the cross terms vanish by the definition of Π_k^∇. The first term ensures consistency and, since it involves only integrals with known polynomial functions, can be computed exactly. The second term provides stability and, following [Beirão da Veiga et al. 2013], is approximated using the DOFs:

$$\sum_{i=1}^{N_{\mathrm{dof}}} D_i\Big((I-\Pi_k^\nabla)\varphi_i\Big)\; D_i\Big((I-\Pi_k^\nabla)\varphi_j\Big). \tag{17}$$

Once K^E is computed, the global stiffness matrix K can be assembled.

3.2 Particle-to-Grid Transfer

In our formulation, the particle-to-grid (P2G) step builds the DOFs of the first-order velocity field and assembles the right-hand-side vector $\vec{r}$, with entries

$$\vec{r}_i = \frac{1}{\Delta t}\int_\Omega \mathbf{u}_h' \cdot \nabla\varphi_i \,\mathrm{d}x. \tag{18}$$

As discussed in Section 3.1, for a discrete scalar field, the first-order DOFs correspond to its values at vertices. Therefore, we adopt a PIC-style transfer to reconstruct velocity DOFs from particles.

The coefficient of the i-th degree of freedom of the discrete scalar field f_h is computed via normalized radial kernel averaging:

$$\vec{f}_i = \frac{\sum_p f_p \, w_{ip}}{\sum_p w_{ip}}, \tag{19}$$

where f_p denotes the value of the scalar field carried by particle p. For the weights w_{ip}, we use a radial kernel function given by

$$w_{ip} = k\Big(\|\mathbf{x}_p - \mathbf{x}_i\|^2/R^2\Big), \qquad k(q) = \max\big(0, (1-q)^3\big), \tag{20}$$

where $\mathbf{x}_i$ denotes the position of the vertex associated with the i-th degree of freedom, and the kernel radius R is set to the edge length of the Cartesian grid embedding the boundary. After P2G, the DOF vector $\vec{\mathbf{u}}$ of the velocity field $\mathbf{u}_h$ is obtained, from which the intermediate velocity $\mathbf{u}_h'$ is computed by applying external forces,

$$\mathbf{u}_h' = \mathbf{u}_h + \mathbf{f}_h \,\Delta t. \tag{21}$$

Since both $\mathbf{u}_h$ and $\mathbf{f}_h$ lie in the same linear space V_1, the force contribution is added directly at the DOF vector level, with $\mathbf{f}_h$ computed using the Boussinesq buoyancy model.

Given the intermediate velocity DOF vector $\vec{\mathbf{u}}'$, the right-hand-side term is evaluated via polynomial projection. Specifically, replacing $\mathbf{u}_h'$ with its projected polynomial $\Pi_1^\nabla \mathbf{u}_h'$ in Eq. (18) yields:

$$\frac{1}{\Delta t}\int_\Omega \mathbf{u}_h' \cdot \nabla v_i \,\mathrm{d}x \;\approx\; \frac{1}{\Delta t}\sum_{E\subset\Omega}\int_E \Pi_1^\nabla \mathbf{u}_h' \cdot \nabla v_i \,\mathrm{d}x. \tag{22}$$

Then, by the definition of the projection Π_k^∇, we have

$$\frac{1}{\Delta t}\int_E \Pi_1^\nabla \mathbf{u}_h' \cdot \nabla v_i \,\mathrm{d}x = \frac{1}{\Delta t}\int_E \Pi_1^\nabla \mathbf{u}_h' \cdot \Pi_2^\nabla(\nabla v_i) \,\mathrm{d}x, \tag{23}$$

which reduces the evaluation to a polynomial integral over each element and makes the right-hand-side vector $\vec{r}$ directly computable.

3.3 Grid-to-Particle Transfer

Leveraging polynomial projection, we generalize the FLIP formulation to the conforming VEM scheme, resulting in improved preservation of fine-scale smoke details. Within each element, the projected pressure field admits a quadratic polynomial expansion,

$$\Pi_2^\nabla p_h^2 = \sum_{\alpha=1}^{N_2} s_\alpha^p \, m_\alpha. \tag{24}$$

Similarly, the projected intermediate velocity field is represented using first-order polynomials as

$$\Pi_1^\nabla \mathbf{u}_h'^1 = \sum_{\alpha=1}^{N_1} \mathbf{s}_\alpha^{\mathbf{u}'} \, m_\alpha. \tag{25}$$

A divergence-free velocity field is obtained by updating the polynomial representation of the velocity,

$$\Pi_1^\nabla \mathbf{u}_h^{\text{new}} = \Pi_1^\nabla \mathbf{u}_h'^1 - \nabla \Pi_2^\nabla p_h^2. \tag{26}$$

Since $\nabla \Pi_2^\nabla p_h^2$ is itself a first-order polynomial, its coefficients can be directly combined with those of $\Pi_1^\nabla \mathbf{u}_h'^1$. After obtaining the updated velocity field in the polynomial space, we compute the velocity increment as:

$$\Pi_1^\nabla \Delta \mathbf{u}_h = \Pi_1^\nabla \mathbf{u}_h^{\text{new}} - \Pi_1^\nabla \mathbf{u}_h. \tag{27}$$

The resulting velocity increment is transferred to particles using the standard FLIP scheme [Zhu and Bridson 2005], with particle velocities updated as:

$$\mathbf{u}_p^{\text{new}} = \alpha_{\text{FLIP}} \, \mathbf{u}_p^{\text{old}} + \Pi_1^\nabla \Delta \mathbf{u}_h(\mathbf{x}_p) + (1 - \alpha_{\text{FLIP}}) \, \Pi_1^\nabla \mathbf{u}_h^{\text{new}}(\mathbf{x}_p), \tag{28}$$

where $\mathbf{u}_p^{\text{old}}$ and $\mathbf{u}_p^{\text{new}}$ denote the particle velocity before and after the update, respectively, $\mathbf{x}_p$ denotes the particle position, and $\alpha_{\text{FLIP}} \in [0, 1]$ controls the balance between FLIP and PIC updates. Following the velocity update, particle advection is performed using a second-order Runge–Kutta scheme. During advection, an AABB tree is employed for collision detection, and mirror reflection is applied to project particles back into the fluid domain.

4 Galerkin Geometric Multigrid for VEM

Designing Galerkin multigrid methods for VEM Poisson problems on general polyhedral grids is particularly challenging, as Galerkin projection on general polyhedral grids often induces significant connectivity proliferation among DOFs, leading to dense coarse-level system matrices [Xian et al. 2019]. Here, connectivity refers to the presence of nonzero entries in the system matrix coupling pairs of DOFs. This issue is exacerbated in second-order VEM on cut-cell grids for two main reasons. First, second-order VEM exhibits dense local connectivity: vertex, edge, face, and volume DOFs within each cell are all strongly coupled, thereby amplifying the propagation of connectivity compared to first-order schemes. Second, cut-cell grids are highly irregular, with some cells containing a large number of vertices, edges, and faces, leading to substantially larger local stiffness matrices and making the diffusion effect even more pronounced. To prevent connectivity proliferation from degrading multigrid efficiency, we introduce a *VEM-specific, diffusion-free prolongation operator* that prevents cross-cell DOF connectivity, thereby avoiding dense coarse-level system matrices.

4.1 Multigrid Overview

Multigrid methods are theoretically optimal and practically efficient solvers for large sparse linear systems. Their effectiveness stems from a divide-and-conquer strategy that combines two complementary processes: relaxation and coarse-grid correction. Relaxation applies a smoother to eliminate high-frequency errors on the fine grid. Under the standard assumption that low-frequency errors on a fine grid appear as relatively higher-frequency errors on a coarser grid, coarse-grid correction transfers these errors to a coarser level via *restriction*, applies smoothing there, and then propagates the correction back to the fine grid through *prolongation*.

Our method belongs to the class of geometric multigrid methods with Galerkin coarse-grid approximation [Briggs et al. 2000]. Unlike purely geometry-based multigrid schemes that re-discretize the governing equations on coarser grids, both prolongation and restriction operators in our method are constructed through the Galerkin framework. Consequently, the efficiency of the solver critically depends on the construction of the grid hierarchy and the design of inter-grid transfer operators. Galerkin multigrid constructs the coarse-level system matrix A_c from the fine-level matrix A_f as

$$A_c = R A_f P, \tag{29}$$

where R and P denote the restriction and prolongation operators, respectively. To ensure variational consistency, Galerkin multigrid typically sets $R = P^\mathsf{T}$, making the design of the prolongation operator crucial for the solver's efficiency. In this setting, the prolongation operator must accurately interpolate smooth functions to achieve rapid convergence, while the prolongation matrix P should remain sparse to prevent densification of coarse-level matrices.

4.2 Naive VEM Prolongation Operator

Assuming a hierarchy of polyhedral grids $\{\mathcal{P}^\ell\}_{\ell=1}^L$, ordered from coarse to fine, with corresponding VEM spaces $\{V^\ell\}_{\ell=1}^L$, prolongation maps a function $v_h^{\ell-1} \in V^{\ell-1}$ on the coarse level to a function $v_h^\ell \in V^\ell$ on the fine level. We first attempt to generalize the definition of the prolongation operator used for first-order 2D VEM in [Antonietti et al. 2023] to the second-order 3D VEM, resulting in the *naive VEM prolongation operator*. This operator reconstructs the fine-level function from coarse-level degrees of freedom by directly transferring shared vertex DOFs and evaluating the remaining fine-level DOFs from the projected coarse-level function, i.e.,

$$v_h^\ell := \sum_{i \in \mathcal{N}(V^{\ell-1})} D_i\left(v_h^{\ell-1}\right) \varphi_i^\ell + \sum_{i \in \mathcal{N}(V^\ell \setminus V^{\ell-1})} D_i\left(\Pi_2^\nabla v_h^{\ell-1}\right) \varphi_i^\ell, \tag{30}$$

where $\mathcal{N}(V^\ell)$ denotes the index set of DOFs in V^ℓ. This prolongation requires computing the face- and volume-moments of a fine-level polyhedral cell, which, in turn, necessitates a *nested* polyhedral grid hierarchy to ensure their efficient and consistent evaluation. Here, nested indicates that each fine-level polyhedral cell is fully contained within a single coarse-level cell.

Through the experiments, we found that the naive VEM prolongation operator induces a diffusion effect, leading to extremely dense coarse-level system matrices and poor efficiency when used either as a standalone solver or as a preconditioner. This diffusion effect is

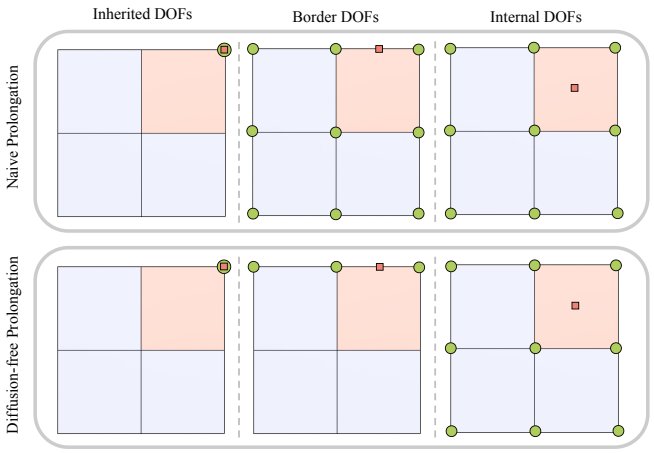


Fig. 3. The large square represents a coarse-level cell, with green circles indicating coarse-level DOFs. The orange square denotes a fine-level cell, with orange markers indicating fine-level DOFs. To prolong error corrections from the coarse level to the fine level, coarse-level DOFs are used as interpolation points; the figure illustrates which coarse-level DOFs are involved in the prolongation of different categories of fine-level DOFs. Compared to the naive prolongation operator, our diffusion-free prolongation modifies the treatment of boundary DOFs (middle), effectively preventing the spurious spread of connectivity induced by boundary DOFs.

illustrated experimentally in Fig. 9, and a theoretical analysis of the phenomenon is provided in Appendix B.

4.3 Diffusion-Free VEM Prolongation Operator

To prevent connectivity between DOFs across cells at coarse levels, we propose a *diffusion-free VEM prolongation operator*, defined as

$$\begin{aligned} v_h^\ell := & \sum_{i\in\mathcal{N}(V^{\ell-1})} D_i\left(v_h^{\ell-1}\right)\,\varphi_i^\ell \\ & + \sum_{i\in\mathcal{N}_e(V^\ell\setminus V\ell-1)} D_i\left(\Pi_{2,e}^\nabla v_h^{\ell-1}\right)\,\varphi_i^\ell \\ & + \sum_{i\in\mathcal{N}_f(V^\ell\setminus V\ell-1)} D_i\left(\Pi_{2,f}^\nabla v_h^{\ell-1}\right)\,\varphi_i^\ell \\ & + \sum_{i\in\mathcal{N}_v(V^\ell\setminus V\ell-1)} D_i\left(\Pi_2^\nabla v_h^{\ell-1}\right)\,\varphi_i^\ell. \end{aligned} \tag{31}$$

Here, newly introduced fine-level DOFs are partitioned according to their physical meaning. Specifically, $\mathcal{N}_e$, $\mathcal{N}_f$, and $\mathcal{N}_v$ denote the sets of edge, face, and volume DOFs, respectively. These sets form a disjoint decomposition of the newly introduced fine-level DOFs:

$$\mathcal{N}_e^\ell \cup \mathcal{N}_f^\ell \cup \mathcal{N}_v^\ell = \mathcal{N}(V^\ell) \setminus \mathcal{N}(V^{\ell-1}), \qquad \mathcal{N}_e^\ell \cap \mathcal{N}_f^\ell \cap \mathcal{N}_v^\ell = \emptyset.$$

The operators $\Pi_{2,e}^\nabla$ and $\Pi_{2,f}^\nabla$ denote quadratic polynomial projections constructed via one- and two-dimensional VEM formulations on coarse-level edges and faces, respectively. These projections depend only on the DOFs associated with the corresponding edges or faces, rather than on all DOFs of the polyhedral element, thereby ensuring a diffusion-free prolongation. Under the proposed prolongation operator, the resulting coarse-level matrix satisfies $A_c(i,j) \neq 0$ if and only if DOFs i and j belong to the same polyhedral element at the coarse level. The one- and two-dimensional VEM formulations are obtained by straightforward dimensional reduction of the three-dimensional formulation described in Section 3.1.

An intuitive comparison between the naive prolongation and the proposed diffusion-free prolongation is shown in Fig. 3. With the prolongation operator defined, the multigrid hierarchy matrices are constructed according to Eq. 29. For high-frequency error reduction, we employ a Chebyshev smoother with default parameters, implemented in [Demidov 2020]. Compared to Jacobi smoothing, it is better suited to VEM formulations and, unlike Gauss–Seidel smoothers, is straightforward to parallelize on GPUs.

5 Multilevel Body-Fitted Grid Construction

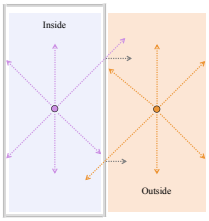
To support the diffusion-free geometric multigrid method described above, a multilevel hierarchy of nested, body-fitted polyhedral grids is required. This section describes the construction of body-fitted polyhedral grids and their associated multilevel hierarchy. Specifically, Section 5.1 presents the generation of a body-fitted grid from a given boundary mesh, and Section 5.2 introduces the construction of the nested multilevel grid hierarchy. A illustrative example is provided in Fig. 4 to facilitate understanding of the algorithm.

5.1 Binary Space Cut-Cell

Inspired by [Pan et al. 2025; Zhang et al. 2024], we propose a simple and robust binary space partitioning-based cut-cell strategy for constructing body-fitted polyhedral grids. Starting from a background Cartesian grid, each boundary triangle is first associated with all grid cells it intersects. This association is determined by uniformly sampling points on each triangle and assigning the triangle to any grid cell containing at least one sample.

For each grid cell, the associated triangles are ordered according to the number of intersections between their supporting planes and other triangles, with those exhibiting fewer intersections prioritized for half-space partitioning. This ordering reduces the number of cut cells and, consequently, the size of the resulting VEM stiffness matrix. Following this order, half-space partitioning is applied independently within each grid cell: the supporting plane of the selected triangle subdivides the current polyhedral cell into sub-cells, and the remaining triangles are propagated to the corresponding sub-cells based on their spatial relationship to the cutting plane. This procedure continues until all relevant boundary triangles have been processed, yielding a body-fitted polyhedral grid. During body-fitted grid construction, partitions that produce elements with volumes below 10^{-12} are discarded to improve numerical robustness.

Each resulting polyhedron is subsequently classified as interior or exterior with respect to the boundary mesh using ray casting. Rays are emitted from the centroid of polyhedron and intersected with the boundary mesh; a polyhedron is classified as interior if more than half of the rays intersect boundary triangles whose normals form an angle smaller than 90° with the ray direction. An two-dimensional illustrative ray casting example is shown in right. Based on this classification, only



exterior polyhedra are retained for VEM discretization, thereby naturally embedding boundary information into the grid construction. Ray casting is accelerated using an AABB tree, with all intersection queries handled by CGAL.

5.2 Multilevel Grid Construction

As discussed in Section 4, our multigrid method requires a nested grid hierarchy capable of representing fine-level low-frequency errors on the coarser grids. This implies that coarse grids should remain as well aligned with the boundary geometry as possible, so that smooth errors near boundaries can be effectively represented through well-posed placement of coarse-level DOFs.

To satisfy these requirements, we propose a *nested, boundary-aware* grid hierarchy construction algorithm. Building on the cut-cell procedure described in Section 5.1, each initial regular cell is treated as the root of a binary tree that records the sequence of half-space partitions applied during grid construction. To enable boundary-aware hierarchy construction, each partition is assigned an importance weight w that quantifies the contribution of the corresponding cut to boundary embedding. Since a cut may introduce a polygonal face that is only partially aligned with the boundary geometry, the importance weight w is used to quantify the boundary-covered portion of that face. It is defined as the area of the boundary-aligned portion divided by the total area of the boundary mesh. These weights then guide a grid coarsening process, yielding a boundary-aware grid hierarchy.

Coarsening is performed for $L-1$ iterations, yielding coarse-level polyhedral grids $\{\mathcal{P}^\ell\}_{\ell=1}^{L-1}$. The transition from $\mathcal{P}^\ell$ to $\mathcal{P}^{\ell-1}$ proceeds in two stages. First, regular cubic cells that do not contain cuts are aggregated following standard geometric multigrid practice, effectively halving the grid resolution. Second, for irregular grid cells introduced by cuts, a greedy strategy is applied: removable partitions with the smallest importance weights are iteratively eliminated until the accumulated removed weight exceeds $1/L$, where L denotes the total number of levels. A partition is considered removable if its two child cells correspond to leaf nodes; removing it merges the two cells into a single cell and coarsens the grid locally.

After coarsening, the grid $\mathcal{P}^{\ell-1}$ is extracted from the remaining cells. A cell is included in $\mathcal{P}^{\ell-1}$ if it intersects the region enclosed by the boundary mesh. Since each coarse cell can be expressed as the union of a set of finest-level cells, this test is evaluated efficiently by checking whether any of the corresponding finest cells is classified as interior. The interior/exterior classification at the finest level is obtained via ray casting, as described in Section 5.1.

By construction, the resulting grid hierarchy is nested: for any $c^\ell \in \mathcal{P}^\ell$, there exists a cell $c^{\ell-1} \in \mathcal{P}^{\ell-1}$ such that $c^\ell \subset c^{\ell-1}$, and each coarse cell contains at least one finer cell. This property prevents rank-deficient restriction operators and avoids the introduction of spurious nullspaces in coarse-level system matrices.

6 Results

We evaluate FastVEM on 14 scenes spanning a broad range of complex boundary configurations; the corresponding simulation settings and timings are summarized in Table 1. In all cases, the boundary surface meshes are embedded into a 128^3 Cartesian background grid to construct body-fitted grids, and a five-level hierarchy is built for the multigrid solver. FLIP particles are initialized according to the background grid resolution, with eight particles placed per grid cell. Particles located outside the simulation domain Ω (i.e., inside the boundary mesh) are marked as ghost particles to reduce P2G bias, and are excluded from both G2P transfer and advection. In our setup, one time step corresponds to one frame. During advection, we employ substepping with a CFL number of 1 and grid spacing $\Delta x = 1/N$, where N denotes the background grid resolution. The Advection column in Table 1 reports the total advection cost per time step, including all substeps.

For each scene, we provide two complementary visualizations: a conventional smoke rendering and a particle trajectory–based velocity visualization. The latter encodes particle speed using a colormap, with blue indicating higher velocities, and additionally highlights particles traveling near solid boundaries using red-colored particles to emphasize near-boundary flow behavior. Green/blue pathlines trace 500–1000 randomly selected smoke particles, while red pathlines trace 100–300 particles that remain near solid boundaries for more than 50 consecutive frames. All experiments were conducted on a machine equipped with an Intel i9-13900K CPU, 64 GB of RAM, and an RTX 3090 GPU. Exact intersection computations required for cut-cell are handled using CGAL [Fabri and Pion 2009].

6.1 Complex Boundary Handling

FastVEM robustly resolves sub-grid boundary features that are challenging for Cartesian grid methods, including thin structures (Fig. 5), narrow gaps (Fig. 1; Fig. 13), and highly curved surfaces (Fig. 12). To more closely examine this advantage, we compare FastVEM against a Cartesian grid–based fluid solver on the *Brush* and *Plane* scenes, both of which contain thin structures and sub-grid geometric details. For this comparison, we use Blender's built-in smoke simulator, Mantaflow [Thuerey and Pfaff 2018], configured with a grid resolution of 256^3 to provide a more challenging reference.

As shown in Fig. 6, the smoke motion produced by Mantaflow exhibits limited sensitivity to sub-grid boundary features. In the *Brush* scene, Mantaflow fails to capture the influence of thin, rod-like structures on the flow, whereas FastVEM generates filamentary smoke patterns that accurately reflect the flow perturbations induced by these structures. The *Plane* scene provides an even more illustrative comparison. We set the plane thickness to 10^{-5} times the side length of the cubic simulation domain, a regime that Mantaflow is essentially unable to handle: the smoke passes through the plane with little apparent interaction, as if the boundary were absent. In contrast, FastVEM produces boundary-conforming flow behavior, with the fluid being deflected by the plane while portions of the flow pass through the narrow gaps. For reference, the thickness of a real sheet of paper is on the order of 0.1 mm, indicating that FastVEM can resolve the influence of paper-scale boundaries within simulation domains on the order of $10\,\mathrm{m} \times 10\,\mathrm{m} \times 10\,\mathrm{m}$.

To further quantitatively evaluate the boundary-handling accuracy of FastVEM, we measure the geometric error between the generated cut-cell mesh and the input surface mesh, as reported in Table 1. The error is quantified using the bidirectional Chamfer Distance (CD), computed on meshes after normalized scaling to remove

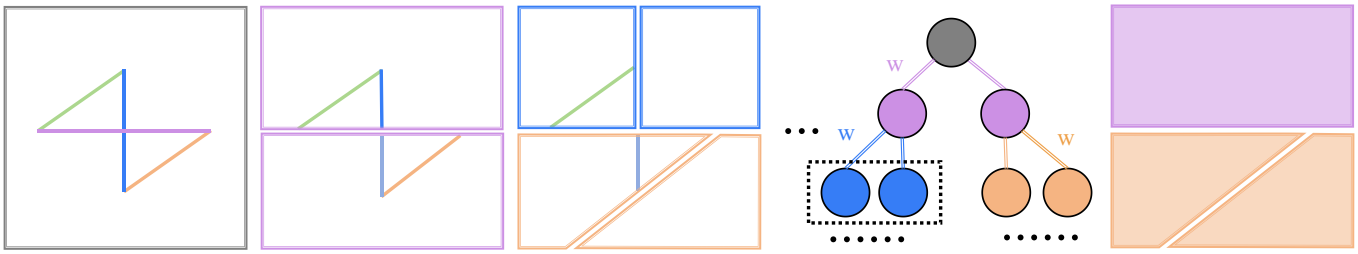


Fig. 4. Illustration of a Binary Space Cut-Cell procedure and the corresponding multilevel grid construction. Starting from a regular gray cell, planar boundaries contained within the cell iteratively partition it through half-space cuts, resulting in body-fitted grids. This process is recorded as a binary tree, where the importance of each partition is quantified by an importance weight w. Based on these weights, a greedy strategy removes cuts that are least beneficial for boundary embedding, yielding a coarsened grid. In this example, removing the cut highlighted by the dashed box produces the grid shown on the right.

Table 1. Statistics of the 14 scenes used in our experiments. *Faces* denotes the number of triangles in the input boundary mesh. *Grid Construction* reports the time required to build the corresponding body-fitted grid. *NNZs* indicates the number of nonzero entries in the resulting stiffness matrix. *CD* denotes the bidirectional Chamfer distance between the extracted cut mesh and the input surface, measuring boundary approximation error. *Precompute* reports the additional time required to construct the geometric multigrid hierarchy.

Scene	Features	Faces	Grid Construction	NNZs	CD	Precompute	P2G	Project	Advection
Bunny	Manifold, Closed	69451	93.6s	$8.9e^8$	$4.4e^{-5}$	42.3s	9.9s	41.6s	8.9s
Dragon	Self-Isects	100000	201.3s	$9.4e^8$	$4.7e^{-5}$	49.1s	11.6s	41.2s	9.3s
Hand	Manifold, Closed	10856	13.7s	$6.5e^8$	$2.8e^{-5}$	38.0s	6.3s	20.6s	8.2s
Rose	Self-Isects, Thin features	35905	105.4s	$7.5e^8$	$3.2e^{-5}$	31.9s	7.1s	25.4s	8.4s
Skull	Self-Isects, Thin features	24808	43.6s	$7.1e^8$	$3.6e^{-5}$	41.6s	8.7s	23.1s	8.0s
Ball	Manifold, Closed	16380	42.3s	$6.4e^8$	$2.2e^{-5}$	37.9s	7.9s	33.5s	7.8s
Hilbert	High genus	31060	36.9s	$8.1e^8$	$1.9e^{-5}$	38.7s	10.0s	37.6s	8.6s
Short	Thin features	9168	15.0s	$6.7e^8$	$1.8e^{-5}$	29.4s	5.3s	34.2s	7.1s
Lucy	Manifold, Closed	99970	128.5s	$9.0e^8$	$3.9e^{-5}$	43.2s	9.7s	39.2s	8.8s
Hair Brush	Self-Isects, Thin features	17442	27.3s	$6.8e^8$	$2.3e^{-5}$	33.5s	11.2s	31.8s	9.3s
Conical Pipe	Non-Manifold, Thin features	15762	2.9s	$6.7e^8$	$8.5e^{-6}$	31.0s	6.8s	25.3s	7.5s
Plane	Thin features	172	3.8s	$6.1e^8$	$3.5e^{-6}$	27.1s	6.1s	23.4s	7.7s
Panel	Thin features	32571	5.7s	$6.3e^8$	$4.1e^{-6}$	35.8s	7.5s	28.3s	8.1s
Building	Non-Manifold, Self-Isects, Open	1908013	235.7s	$6.2e^8$	$2.9e^{-3}$	29.3s	7.7s	17.3s	6.3s

Table 2. Comparison with existing cut-cell–based fluid simulators in terms of runtime for the pressure projection stage. We implement a first-order pressure solver (denoted as Ours*) for comparison with [Azevedo et al. 2016].*Grid* indicates the resolution of the Cartesian background grid used for embedding the boundary mesh.

Method	Pressure Order	Grid	Project
[Azevedo et al. 2016]	1	128^3	47.2 s
[Tao et al. 2022]	2	50^3	457.6 s
Ours*	1	128^3	0.4 s
Ours	2	50^3	3.7 s

global scale differences. Here, the cut-cell mesh refers to the surface mesh extracted from the interior volumetric grid via ray casting, following the procedure described in Section 5. The results show that our cut-cell strategy achieves errors on the order of 10^{-5} to 10^{-6}, indicating that it can reliably capture extremely fine sub-grid boundary features. Fig. 7 further provides a visual illustration of the body-fitted grid and the boundary embedding error.

6.2 Computational Performance Comparison

This subsection evaluates the performance of FastVEM through a series of controlled comparisons. We first benchmark FastVEM against representative boundary-conforming fluid solvers under matched simulation settings, where it achieves speedups exceeding two orders of magnitude in the most time-consuming pressure projection stage. We then analyze the sources of this acceleration via an ablation-style study, comparing the proposed simulation-friendly cut-cell strategy with Mandoline [Tao et al. 2019] and quantitatively assessing the performance gains provided by the multigrid solver.

Comparison with boundary-conforming fluid solvers. Table 2 compares the computational efficiency of FastVEM with closely related cut-cell–based boundary-conforming fluid solvers under comparable experimental settings. We report the runtime of each method in the pressure projection stage. All methods use the same boundary mesh (*Bunny*), and the reported results are taken directly from their published papers. To ensure a fair comparison with the first-order pressure formulation of Batty et al. [Azevedo et al. 2016],

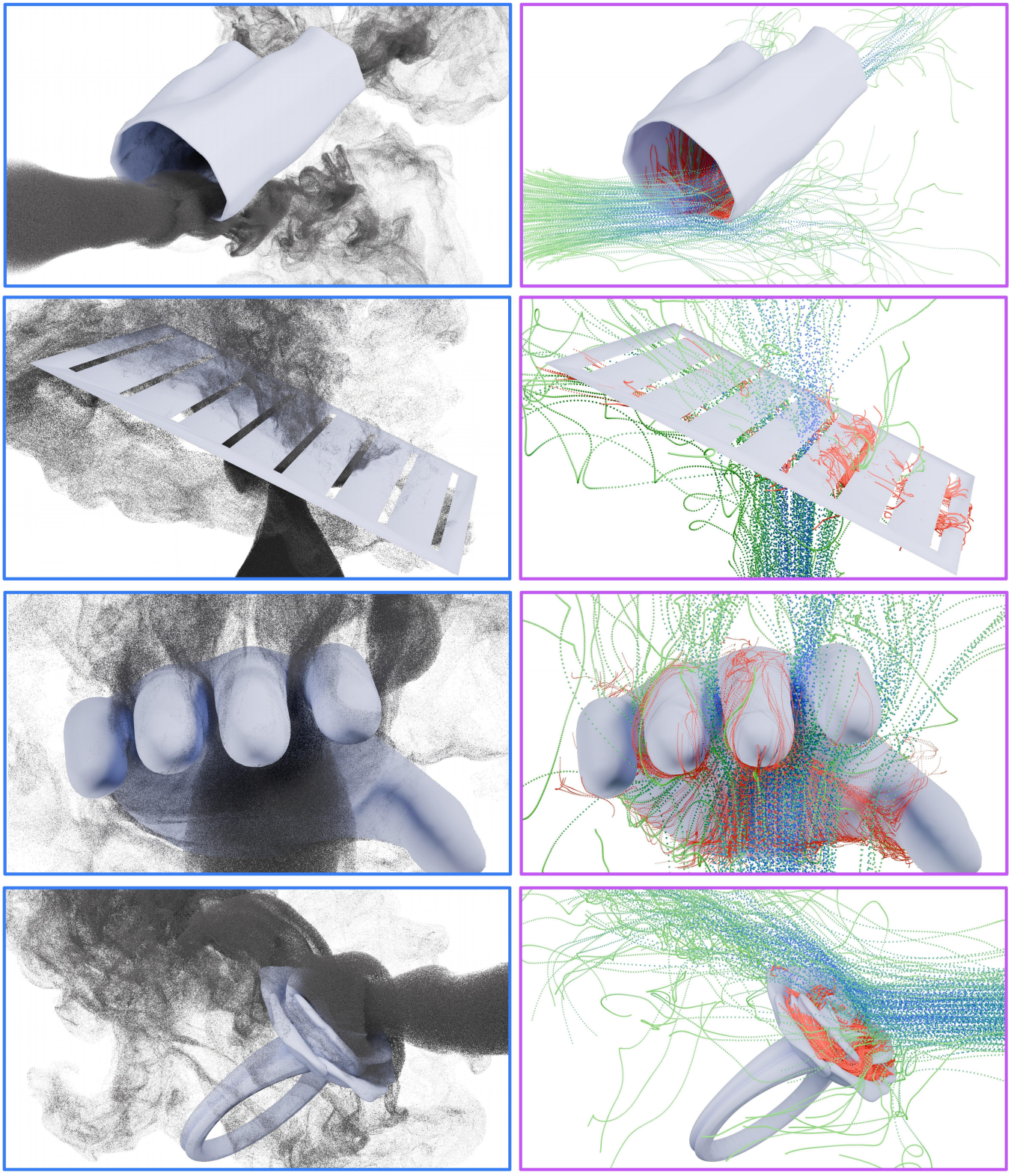


Fig. 5. Smoke simulation results (left) and smoke particle trajectory visualizations colored by velocity magnitude (right), where blue indicates higher speeds. Near-surface flow behavior is visualized using red particles (right). From top to bottom: *Short*, *Plane*, *Hand*, and *Rose*.

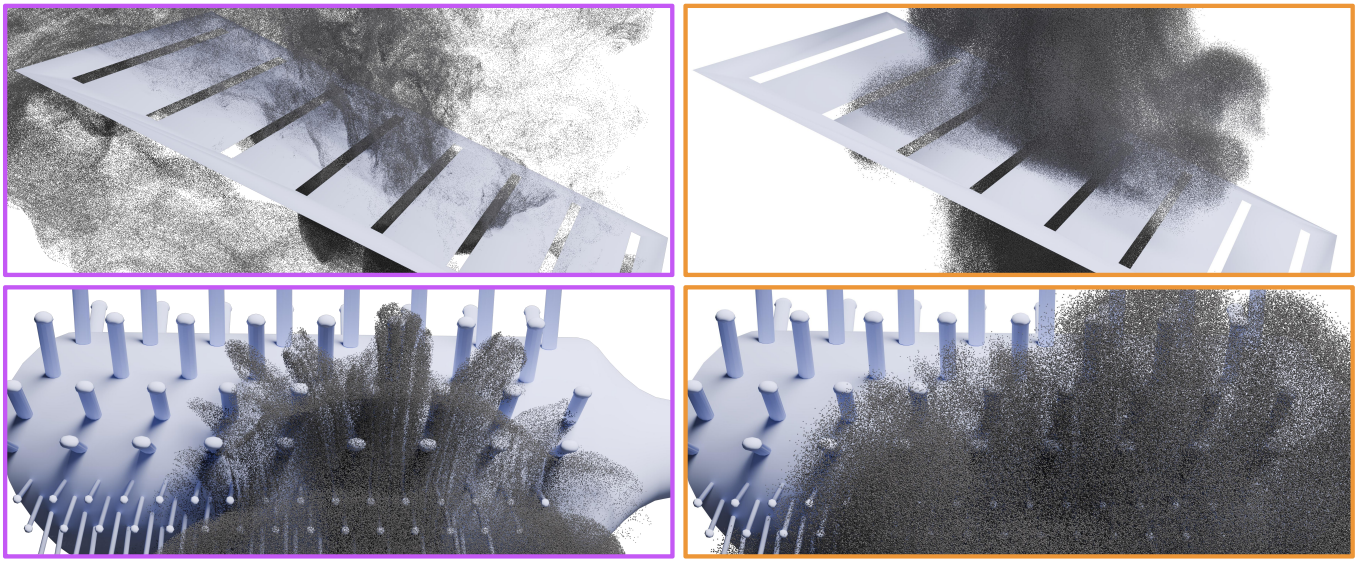


Fig. 6. Comparison with Mantaflow. FastVEM more accurately resolves boundary interactions involving thin structures and narrow gaps (top), as well as sub-grid curved features (bottom), whereas Mantaflow fails to resolve these fine-scale geometric details.

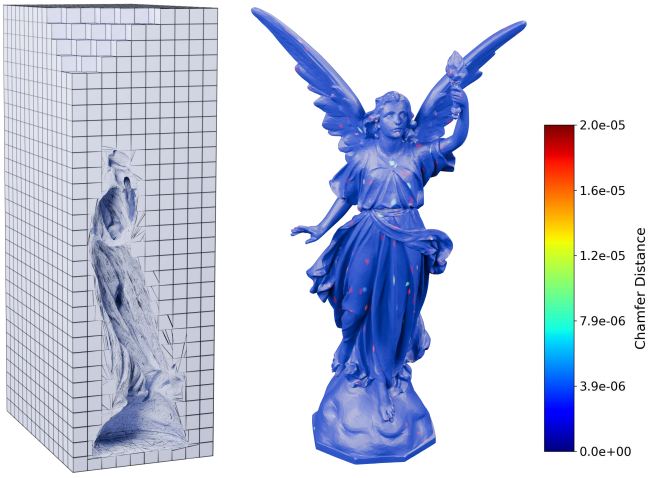


Fig. 7. Embedding of the Lucy model into a Cartesian grid using the proposed Binary Space Cut-Cell method. The Chamfer distance between the embedded boundary and the original surface is visualized on the right, demonstrating the high geometric accuracy of the embedding.

we implement a first-order VEM pressure solver within our framework. As shown in the table, FastVEM demonstrates a substantial efficiency advantage, achieving speedups of approximately two orders of magnitude over both first-order and second-order boundary-conforming pressure solvers.

Comparison with Mandoline. Compared to Mandoline, the proposed binary space partitioning–based cut-cell strategy offers two key advantages. First, in our method, all generated grid cells are guaranteed to be convex, avoiding non–star-shaped elements; this improves grid quality and, in turn, leads to better conditioning of the resulting VEM stiffness matrices. Second, the strategy prevents the accumulation of boundary faces within individual grid cells, yielding a more favorable local degree-of-freedom structure and, consequently, a sparser global stiffness matrix.

Following [Sorgente et al. 2024], we assess the quality of the body-fitted grids using a VEM-specific element quality indicator. For each grid cell E, the overall quality measure is defined as

$$\rho^{3D}(E) = \frac{\sqrt{\rho_1^{3D}(E)\,\rho_2^{3D}(E) + \rho_1^{3D}(E)\,\rho_3^{3D}(E)}}{2}, \tag{32}$$

which combines three complementary factors to characterize the geometric regularity and topological complexity of the generated grids. Detailed definitions of the individual terms are provided in Appendix C. Based on the element-wise quality measure, the overall quality of a body-fitted polyhedral grid $\mathcal{P}$ is defined as the root-mean value of the quality measures over all cut cells:

$$\rho(\mathcal{P}) = \sqrt{\frac{1}{\#\{E^c \in \mathcal{P}\}} \sum_{E^c \in \mathcal{P}} \rho^{3D}(E^c)}. \tag{33}$$

Table 3 reports quantitative measures of body-fitted grid quality, the number of nonzeros in the corresponding stiffness matrix, the runtime of the pressure projection stage when simulating fluid flow on the resulting grids, and the cut-cell runtime for both the proposed cut-cell strategy and Mandoline. In this comparison, the boundary mesh is embedded into a moderate 50^3 background grid to ensure that the pressure projection cost on Mandoline-generated grids remains tractable. Since Mandoline does not provide a grid hierarchy, pressure projection on its grids is solved using UA-AMGPCG.

As shown in the first three columns of Table 3, FastVEM consistently produces higher-quality body-fitted grids and sparser stiffness

Table 3. Quantitative comparison of cut-cell grid quality and performance between the proposed method and Mandoline [Tao et al. 2019]. We report grid quality (defined in Eq. 32), global stiffness matrix nonzeros (NNZs), pressure projection time, and cut-cell runtime. For each entry, values are reported in the form *Ours / Mandoline*.

Scene	Quality	NNZs	Project	Runtime
Bunny	0.78 / 0.63	$4.7e^8$ / $7.1e^8$	7.9s / 127.1s	84.2s / 4.6s
Dargon	0.77 / 0.48	$6.5e^8$ / $7.4e^8$	12.3s / 148.5s	187.1s / 5.4s
Hand	0.79 / 0.63	$1.1e^8$ / $1.9e^8$	6.2s / 15.8s	10.3s / 1.8s
Rose	0.77 / 0.62	$2.3e^8$ / $4.7e^8$	5.5s / failed	93.9s / 6.5s
Skull	0.78 / 0.62	$1.9e^8$ / $2.7e^8$	4.7s / 80.5s	38.9s / 6.7s
Ball	0.78 / 0.64	$1.0e^8$ / $2.0e^8$	1.7s / 221.8s	41.1s / 1.6s
Hilbert	0.80 / 0.65	$2.4e^8$ / $3.8e^8$	7.2s / 103.2s	29.6s / 4.7s
Short	0.79 / 0.74	$9.4e^7$ / $1.5e^8$	3.1s / 5.9s	11.8s / 2.1s
Lucy	0.77 / 0.52	$7.4e^8$ / $8.3e^8$	11.5s / 254.1s	178.9s / 5.1s
Hair Brush	0.79 / 0.52	$1.3e^8$ / $2.3e^8$	3.5s / failed	23.3s / 8.9s
Conical Pipe	0.79 / 0.74	$1.2e^8$ / $2.4e^8$	8.8s / 21.2s	16.4s / 3.0s
Plane	0.85 / 0.75	$4.6e^7$ / $1.2e^8$	8.3s / 9.1s	1.6s / 0.4s
Panel	0.81 / 0.61	$8.3e^7$ / $3.8e^8$	6.7s / failed	3.4s / 2.9s
Building	0.85 / -	$6.1e^7$ / -	2.5s / -	218.2s / failed

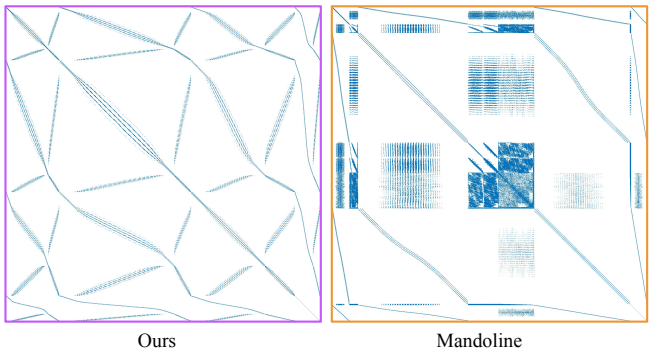


Fig. 8. Compared to body-fitted grids constructed with Mandoline for VEM discretization, our cut-cell strategy produces significantly sparser stiffness matrices and avoids irregular dense substructures. This is a direct consequence of our binary space partitioning strategy, which fundamentally prevents excessive accumulation of boundary faces within individual cells.

matrices, which together lead to faster pressure projection. Fig. 8 provides a direct visual comparison of the resulting stiffness matrices, illustrating that Mandoline generates locally dense matrix blocks, whereas our method yields a more favorable sparsity pattern.

Compared to Mandoline, our cut-cell strategy incurs a higher preprocessing cost due to the use of exact geometric predicates for robust and accurate body-fitted grid generation, as well as a currently non-parallel implementation. This preprocessing, however, is performed only once and can be reused across simulations with different physical parameters; consequently, its cost accounts for only a small fraction of the overall simulation time.

Multigrid Method Evaluation. Table 4 summarizes the performance of our geometric multigrid preconditioned conjugate gradient (GMGPCG) solver and several alternatives across multiple scenes,

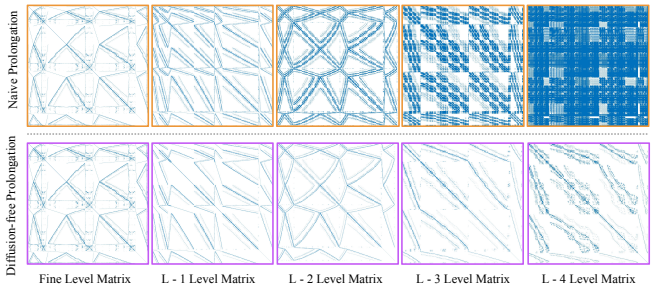


Fig. 9. Compared to the naive prolongation operator obtained by directly extending a two-dimensional first-order VEM multigrid method to three-dimensional second-order settings, the proposed diffusion-free prolongation operator effectively avoids coarse-level matrix densification.

including Diagonal Preconditioned Conjugate Gradient (DPCG), Incomplete Cholesky Preconditioned Conjugate Gradient (ICPCG) using Eigen's implementation [Guennebaud et al. 2010], and Unsmoothed Aggregation Algebraic Multigrid Preconditioned Conjugate Gradient (UA-AMGPCG) provided by AMGCL [Demidov 2020]. In all experiments, we construct five multigrid levels for our solver and set the maximum number of levels in UA-AMGPCG to five as well. Under these matched settings, FastVEM achieves an approximate 4× speedup over the second-best method, UA-AMGPCG. Notably, UA-AMGPCG exhibits unstable performance—particularly in scenes with complex boundaries, where convergence degrades significantly—whereas the proposed GMGPCG solver remains robust due to the boundary-aware grid hierarchy construction.

We further perform ablation studies to assess the impact of two key components of the proposed multigrid framework: the nested, boundary-aware LOD hierarchy construction and the diffusion-free prolongation operator. As reported in Table 4 (column $Ours^{nn}$), replacing our hierarchy construction with the strategy of [Pan et al. 2025], which neither enforces nestedness nor explicitly accounts for boundary geometry, results in a substantial increase in the number of iterations required for convergence and, consequently, higher solution costs. Table 4 (column $Ours^{np}$) further examines the effect of naively extending the prolongation operator proposed in [Antonietti et al. 2023], originally designed for two-dimensional first-order VEM, to three-dimensional second-order pressure solves, as defined in Eq. (30). This extension leads to coarse-level matrices with significantly fill-in, as illustrated in Fig. 9. Although it requires slightly fewer conjugate gradient iterations, the substantially higher per-iteration cost—caused by the reduced sparsity of coarse-level matrices—ultimately renders this approach much less efficient than the proposed diffusion-free prolongation–based multigrid method.

Convergence Tests. We evaluate the accuracy and convergence behavior of the second-order conforming VEM scheme by employing it as a Poisson solver for problems of the form

$$\begin{cases} \nabla^2 f(x) = g(x), & x \in \mathcal{D}, \\ \frac{\partial f}{\partial n}(x) = h(x), & x \in \partial\mathcal{D}, \end{cases} \tag{34}$$

where non-homogeneous Neumann boundary conditions are prescribed, consistent with pressure projection in incompressible flows.

Table 4. Performance comparison for solving Poisson systems arising from VEM discretization. We compare DPCG and ICPCG using Eigen's implementations, and UA-AMGPCG using AMGCL. For both UA-AMGPCG and our GMGPCG, the Chebyshev smoother in AMGCL is used with default parameters.

Scene	DPCG	ICPCG	UA-AMGPCG	Ours^{nn}	Ours^{np}	Ours
Bunny	6975 iters / 3856.6s	503 iters / 1532.0s	140 iters / 111.0s	102 iters / 301.2s	13 iters / 107.5s	16 iters / 41.6s
Dargon	7214 iters / 4725.3s	602 iters / 2566.7	186 iters / 176.9s	96 iters / 283.2s	14 iters / 153.0s	15 iters / 41.2s
Hand	4632 iters / 2475.5s	426 iters / 1231.2s	75 iters / 58.8s	111iters / 348.9s	12 iters / 101.9s	11 iters / 32.6s
Rose	3998 iters / 2474.1s	325 iters / 1487.2s	92 iters / 81.4s	105 iters / 273.8s	9 iters / 75.6s	9 iters / 25.4s
Skull	6325 iters / 4313.2s	459 iters / 2629.1s	168iters / 153.7s	88 iters / 188.9s	11iters / 99.6s	13iters / 31.9s
Ball	6322 iters / 3952.4s	512 iters / 2240.4s	85 iters / 60.6s	97 iters / 285.3s	12 iters / 108.4s	11 iters / 33.5s
Hilbert	5326 iters / 3105.7s	487 iters / 1831.3s	110 iters / 90.1s	129 iters / 287.5s	13 iters / 149.0s	16 iters / 37.6s
Short	2316 iters / 1326.1s	319 iters / 1021.6s	167 iters / 104.6s	71iters / 193.0s	12 iters / 72.4s	13 iters / 34.2s
Lucy	7013 iters / 4366.0s	501 iters / 2593.2s	196iters / 183.0s	130 iters / 88.7s	16iters / 201.7s	15iters / 39.2s
Hair Brush	1721 iters / 1088.2s	206 iters / 781.3s	131 iters / 96.3s	198 iters / 392.2s	12 iters / 89.3s	14 iters / 31.8s
Conical Pipe	3135 iters / 1750.8s	399 iters / 1386.3s	74 iters / 52.3s	67 iters / 237.8s	11 iters / 81.1s	11 iters / 25.3s
Plane	3845 iters / 2160.3s	412 iters / 1620.2s	217 iters / 129.4s	17 iters / 41.0s	8 iters / 68.2s	8 iters / 23.4s
Panel	2296 iters / 1421.9s	267 iters / 1125.8s	108 iters / 64.4s	39 iters / 78.7s	8 iters / 78.0s	10 iters / 28.3s
Building	2356 iters / 1517.5s	231 iters / 916.9s	52 iters / 42.6s	68 iters / 153.2s	7 iters / 61.1s	7 iters / 17.3s

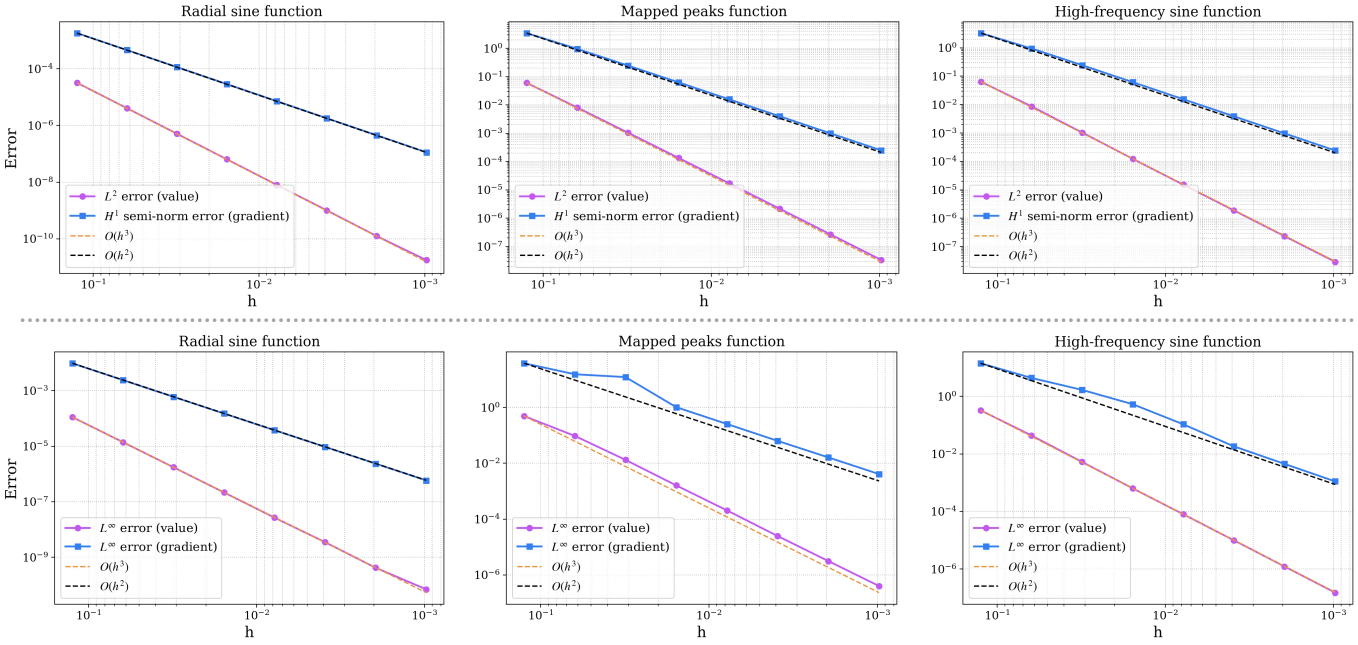


Fig. 10. Convergence study of second-order conforming VEM on three test functions (radial sine, mapped peaks, and high-frequency sine). Errors are plotted against the grid resolution parameter h in log–log scale, where h is defined as the inverse of the grid resolution (e.g., $h = 1/N$ for an $N \times N$ discretization). We report L^2 and L^∞ errors for the solution, and the H^1 semi-norm and L^∞ errors for its gradient. The slopes indicate approximately third-order convergence in the solution and second-order convergence in the gradient.

Convergence tests are conducted on a 1×1 2D domain using three representative functions. A circular obstacle centered at $(0.5, 0.5)$ with radius 0.2, approximated by a piecewise-linear curve with 256 segments, is embedded into the Cartesian background grid. The test functions include a radial function $f(x, y) = r \sin r$ with $r = \sqrt{x^2 + y^2}$; a mapped peaks function defined by evaluating the standard MATLAB peaks function under the transformation $X = 6x - 3$, $Y = 6y - 3$; and a high-frequency function $f(x, y) = \sin(kx)\sin(ky)$ with $k = 5\pi$. As shown in Fig. 10, the second-order VEM exhibits approximately third-order convergence in the solution and second-order convergence in its gradient, in agreement with the theoretical convergence rates [Beirão da Veiga et al. 2013].

Further Applications. Our cut-cell strategy can be combined with planar detection techniques to robustly handle extremely complex and noisy reconstructed meshes, for which Mandoline fails to produce valid outputs. As shown in Fig. 11, FastVEM successfully

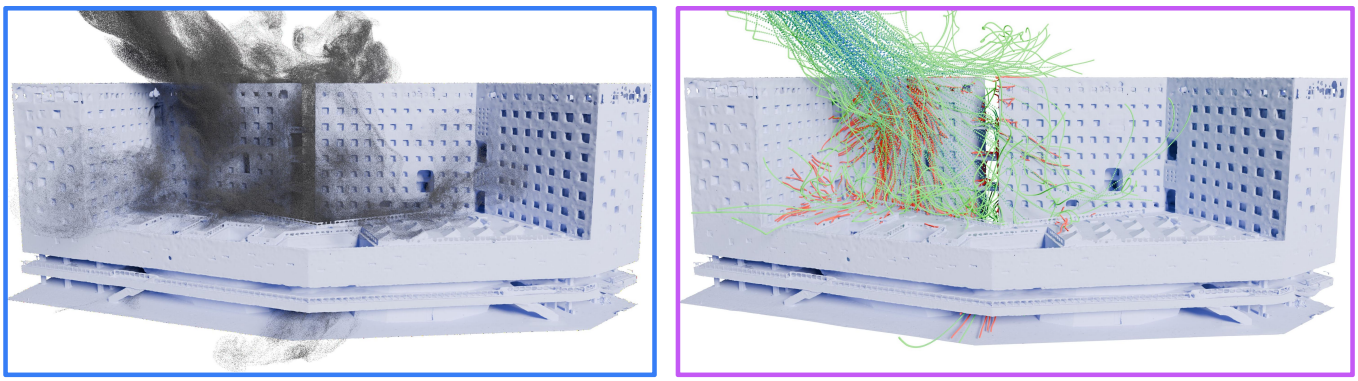


Fig. 11. FastVEM supports fluid interaction with extremely complex and noisy models reconstructed from drone scans.

simulates smoke interacting with a highly complex building model reconstructed from drone scans containing nearly two million faces. More interestingly, as reported in Table 3, the resulting body-fitted grids exhibit high element quality, which in turn leads to reduced computational cost. Despite relying on planar detection–based approximations, the representation accuracy remains on the order of 10^{-3}, as shown in Table 1. The resulting simulations accurately capture boundary-induced flow behavior: smoke is deflected by complex architectural features, passes through narrow gaps, and naturally emerges near the base of the structure.

7 Disscussion and Future Works

We present *FastVEM*, a boundary-conforming fluid simulator on body-fitted grids that speeds up the computationally dominant pressure projection stage by up to 100× over existing cut-cell–based fluid simulators. FastVEM combines a generalized FLIP formulation for advection with a first-order velocity and second-order pressure VEM discretization to robustly enforce incompressibility and boundary conditions. Complementing this discretization, we introduce a simple yet robust binary space partitioning–based cut-cell strategy for constructing simulation-friendly body-fitted grids. This strategy offers two key advantages: it guarantees convex grid cells, improving numerical stability, and it avoids excessive clustering of boundary facets within individual cells, thereby preventing overly dense local stiffness matrices. To further accelerate pressure projection, we develop a Galerkin geometric multigrid solver with a diffusion-free prolongation operator that prevents coarse-level matrix densification, together with a nested, boundary-aware grid hierarchy that improves convergence. Extensive experiments demonstrate that FastVEM robustly handles a wide range of complex boundaries while maintaining practical computational cost.

Our work is subject to several limitations. First, despite robust and efficient handling of complex boundary geometries, FastVEM is currently limited to static boundaries. Applying our framework to deforming boundaries or liquid simulations requires frequent grid updates to track evolving surfaces, incurring tens of seconds of overhead per timestep. This overhead stems from the lack of hierarchical body-fitted grid construction algorithms that preserve both mesh quality and efficiency, which remains an open problem. A highly parallel, floating-point–based binary space cut-cell strategy, instead of our current serial implementation relying on CGAL's exact intersection computations, could potentially mitigate this cost. Second, as shown in Table 1, second-order VEM discretization produces up to 9.4×10^8 nonzeros at 128^3, requiring 14 GB of GPU memory for the solver, which limits scalability to higher resolutions. Further exploration of more efficient stiffness matrix storage for regular-grid regions within the body-fitted grid may alleviate this limitation. Third, while FastVEM can handle boundaries with open surfaces by implicitly sealing them during the ray-casting stage, this design choice limits its ability to support intentionally open surface features. Nevertheless, interactions with objects typically modeled as open surfaces, such as cloth or planar sheets, can be simulated by representing them as thin shells with finite thickness, as illustrated in Fig. 5. Extending FastVEM to directly support intentionally open surfaces could broaden the range of fluid–boundary interactions, but would require specialized partitioning of degrees of freedom near open boundaries and is therefore left for future investigation.

Looking ahead, FastVEM opens several promising directions for future research. First, extending the FastVEM framework to support dynamic boundaries and, more broadly, fluid–solid coupling could enable substantially richer visual effects. Second, as demonstrated by our experiments, constructing simulation-friendly body-fitted grids is far from trivial and leaves considerable room for further exploration; for example, investigating how the ordering of space partitioning in our cut-cell strategy affects grid quality could lead to improved performance. Third, extending our VEM-specific multigrid design to other discretizations is a promising direction. VEM provides locally redundant DOFs with respect to the underlying polynomial space, enabling reduced global coupling in the prolongation operator while preserving interpolation quality, thereby supporting a diffusion-free construction while preserving V-cycle efficiency. Extending this design to other discretizations may require introducing auxiliary DOFs to balance coarse-level sparsity and V-cycle performance. Fourth, FastVEM adopts a Chebyshev smoother rather than a Jacobi smoother because, for second-order VEM discretizations of the Poisson system, naive Jacobi smoothers fail to converge for both our GMGPCG and UA-AMGPCG solvers. This behavior likely arises because different diagonal entries encode

distinct physical meanings and therefore exhibit disparate scales and spectral characteristics. Developing VEM-specific, highly parallel smoothers may thus be key to further improving the efficiency of boundary-conforming fluid simulators. Finally, many applications could benefit from explicitly embedding boundaries into volumetric grids, including fluidic system design and boundary-layer modeling, and we look forward to exploring such integrations in future work.

Acknowledgments

This work was supported by the Natural Science Foundation of China (62595772, 62272245), and the Fundamental Research Funds for the Central Universities (Nankai University, 63263248).

References

Mridul Aanjaneya, Ming Gao, Haixiang Liu, Christopher Batty, and Eftychios Sifakis. 2017. Power diagrams and sparse paged grids for high resolution adaptive liquids. *ACM Trans. on Graphics* 36, 4, Article 140 (July 2017), 12 pages.

Mridul Aanjaneya, Chengguizi Han, Ryan Goldade, and Christopher Batty. 2019. An Efficient Geometric Multigrid Solver for Viscous Liquids. *Proc. ACM Comput. Graph. Interact. Tech.* 2, 2, Article 14 (July 2019), 21 pages.

Ryoichi Ando and Christopher Batty. 2020. A practical octree liquid simulator with adaptive surface resolution. *ACM Trans. on Graphics* 39, 4, Article 32 (Aug. 2020), 17 pages.

Paola F. Antonietti, Stefano Berrone, Martina Busetto, and Marco Verani. 2023. Agglomeration-Based Geometric Multigrid Schemes for the Virtual Element Method. *SIAM J. Numer. Anal.* 61, 1 (2023), 223–249.

Paola F. Antonietti, Lorenzo Mascotto, and Marco Verani. 2018. A multigrid algorithm for the p-version of the virtual element method. *ESAIM: Mathematical Modelling and Numerical Analysis* 52, 1 (2018), 337–364.

Vinicius C. Azevedo, Christopher Batty, and Manuel M. Oliveira. 2016. Preserving geometry and topology for fluid flows with thin obstacles and narrow gaps. *ACM Trans. on Graphics* 35, 4, Article 97 (July 2016), 12 pages.

Christopher Batty, Florence Bertails, and Robert Bridson. 2007. A fast variational framework for accurate solid-fluid coupling. *ACM Trans. on Graphics* 26, 3 (2007), 100–es.

Christopher Batty, Stefan Xenos, and Ben Houston. 2010. Tetrahedral Embedded Boundary Methods for Accurate and Flexible Adaptive Fluids. *Computer Graphics Forum* 29, 2 (2010), 695–704.

Lourenco Beirão da Veiga, Franco Brezzi, Andrea Cangiani, Gianmarco Manzini, L Donatella Marini, and Alessandro Russo. 2013. Basic principles of virtual element methods. *Mathematical Models and Methods in Applied Sciences* 23, 01 (2013), 199–214.

Lourenço Beirão Da Veiga, Franco Brezzi, L. Donatella Marini, and Alessandro Russo. 2023. The virtual element method. *Acta Numerica* 32 (2023), 123–202.

Bernhard Braun, Jan Bender, and Nils Thuerey. 2025. Adaptive Phase-Field-FLIP for Very Large Scale Two-Phase Fluid Simulation. *ACM Trans. on Graphics* 44, 4, Article 42 (July 2025), 23 pages.

Franco Brezzi, Annalisa Buffa, and Konstantin Lipnikov. 2009. Mimetic finite differences for elliptic problems. *ESAIM: Modélisation mathématique et analyse numérique* 43, 2 (2009), 277–295.

William L. Briggs, Van Emden Henson, and Steve F. McCormick. 2000. *A multigrid tutorial (2nd ed.).* Society for Industrial and Applied Mathematics.

Andrea Cangiani, Gianmarco Manzini, and Oliver J. Sutton. 2016. Conforming and nonconforming virtual element methods for elliptic problems. *IMA J. Numer. Anal.* 37, 3 (08 2016), 1317–1354.

Yi-Lu Chen, Jonathan Meier, Barbara Solenthaler, and Vinicius C. Azevedo. 2020. An extended cut-cell method for sub-grid liquids tracking with surface tension. *ACM Trans. on Graphics* 39, 6, Article 169 (Nov. 2020), 13 pages.

Nuttapong Chentanez and Matthias Mueller-Fischer. 2012. A Multigrid Fluid Pressure Solver Handling Separating Solid Boundary Conditions. *IEEE Trans. Visualization & Computer Graphics* 18, 8 (2012), 1191–1201.

Nuttapong Chentanez and Matthias Müller. 2011. Real-time Eulerian water simulation using a restricted tall cell grid. *ACM Trans. on Graphics* 30, 4, Article 82 (July 2011), 10 pages.

Bernardo Cockburn, Jayadeep Gopalakrishnan, and Raytcho Lazarov. 2009. Unified Hybridization of Discontinuous Galerkin, Mixed, and Continuous Galerkin Methods for Second Order Elliptic Problems. *SIAM J. Numer. Anal.* 47, 2 (2009), 1319–1365.

Denis Demidov. 2020. AMGCL —A C++ library for efficient solution of large sparse linear systems. *Software Impacts* 6 (2020), 100037.

Christian Dick, Marcus Rogowsky, and Rüdiger Westermann. 2016. Solving the Fluid Pressure Poisson Equation Using Multigrid—Evaluation and Improvements. *IEEE Trans. Visualization & Computer Graphics* 22 (2016), 2480–2492.

Essex Edwards and Robert Bridson. 2014. Detailed water with coarse grids: combining surface meshes and adaptive discontinuous Galerkin. *ACM Trans. on Graphics* 33, 4, Article 136 (July 2014), 9 pages.

Doug Enright, Duc Nguyen, Frederic Gibou, and Ron Fedkiw. 2003. Using the particle level set method and a second order accurate pressure boundary condition for free surface flows. In *Fluids Engineering Division Summer Meeting*, Vol. 36975. 337–342.

Robert Eymard, Thierry Gallouët, and Raphaèle Herbin. 2000. Finite volume methods. *Handbook of numerical analysis* 7 (2000), 713–1018.

Andreas Fabri and Sylvain Pion. 2009. CGAL: The computational geometry algorithms library. In *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*. 538–539.

Gaël Guennebaud, Benoît Jacob, et al. 2010. Eigen. https://libeigen.gitlab.io.

Mengyun Liu and Xiaopei Liu. 2023. A Parametric Kinetic Solver for Simulating Boundary-Dominated Turbulent Flow Phenomena. *ACM Trans. on Graphics* 42, 6, Article 189 (Dec. 2023), 20 pages.

Frank Losasso, Frédéric Gibou, and Ron Fedkiw. 2004. Simulating water and smoke with an octree data structure. *ACM Trans. on Graphics* 23, 3 (Aug. 2004), 457–462.

Jia-Ming Lu, Tailing Yuan, Zhe-Han Mo, and Shi-Min Hu. 2025. Fast Galerkin Multigrid Method for Unstructured Meshes. *ACM Trans. on Graphics* 44, 6, Article 179 (Dec. 2025), 16 pages.

Chaoyang Lyu, Wei Li, Mathieu Desbrun, and Xiaopei Liu. 2021. Fast and versatile fluid-solid coupling for turbulent flow simulation. *ACM Trans. on Graphics* 40, 6, Article 201 (Dec. 2021), 18 pages.

A. McAdams, E. Sifakis, and J. Teran. 2010. A parallel multigrid Poisson solver for fluids simulation on large grids. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '10)*. 65–74.

Yen Ting Ng, Chohong Min, and Frédéric Gibou. 2009. An efficient fluid–solid coupling algorithm for single-phase flows. *J. Comput. Phys.* 228, 23 (2009), 8807–8829.

Shanshan Pan, Runze Zhang, Yilin Liu, Minglun Gong, and Hui Huang. 2025. Building LOD representation for 3D urban scenes. *ISPRS Journal of Photogrammetry and Remote Sensing* 226 (2025), 16–32.

Daniele A. Di Pietro, Alexandre Ern, and Simon Lemaire. 2014. An Arbitrary-Order and Compact-Stencil Discretization of Diffusion on General Meshes Based on Local Reconstruction Operators. *Computational Methods in Applied Mathematics* 14, 4 (2014), 461–472.

Rajsekhar Setaluri, Mridul Aanjaneya, Sean Bauer, and Eftychios Sifakis. 2014. SPGrid: a sparse paged grid structure applied to adaptive smoke simulation. *ACM Trans. on Graphics* 33, 6, Article 205 (Nov. 2014), 12 pages.

Han Shao, Libo Huang, and Dominik L. Michels. 2022. A fast unsmoothed aggregation algebraic multigrid framework for the large-scale simulation of incompressible flow. *ACM Trans. on Graphics* 41, 4, Article 49 (July 2022), 18 pages.

Tommaso Sorgente, Fabio Vicini, Daniela Cabiddu, Silvia Biasotti, Michela Spagnuolo, Gianmarco Manzini, and Stefano Berrone. 2024. Mesh Quality Meets The Virtual Element Method. In *SIGGRAPH Asia 2024 Courses (SA Courses '24)*. Article 9, 93 pages.

N. Sukumar and A. Tabarraei. 2004. Conforming polygonal finite elements. *Internat. J. Numer. Methods Engrg.* 61, 12 (2004), 2045–2066.

Michael Tao, Christopher Batty, Mirela Ben-Chen, Eugene Fiume, and David I. W. Levin. 2022. VEMPIC: particle-in-polyhedron fluid simulation for intricate solid boundaries. *ACM Trans. on Graphics* 41, 4, Article 115 (July 2022), 22 pages.

Michael Tao, Christopher Batty, Eugene Fiume, and David I. W. Levin. 2019. Mandoline: robust cut-cell generation for arbitrary triangle meshes. *ACM Trans. on Graphics* 38, 6, Article 179 (Nov. 2019), 17 pages.

Nils Thuerey and Tobias Pfaff. 2018. MantaFlow. *URL: http://mantaflow. com* (2018).

Junping Wang and Xiu Ye. 2013. A weak Galerkin finite element method for second-order elliptic problems. *J. Comput. Appl. Math.* 241 (2013), 103–115.

Daniel Weber, Johannes Mueller-Roemer, André Stork, and Dieter Fellner. 2015. A Cut-Cell Geometric Multigrid Poisson Solver for Fluid Simulation. *Computer Graphics Forum* 34, 2 (2015), 481–491.

Zangyueyang Xian, Xin Tong, and Tiantian Liu. 2019. A scalable galerkin multigrid method for real-time simulation of deformable objects. *ACM Trans. on Graphics* 38, 6, Article 162 (Nov. 2019), 13 pages.

Xiaoyu Xiao, Ding Lin, Yiheng Wu, Kai Bai, and Xiaopei Liu. 2025. Simulating Two-Phase Fluid-Rigid Interactions With an Overset-Grid Kinetic Solver. *IEEE Trans. Visualization & Computer Graphics* 31, 10 (2025), 8397–8412.

Omar Zarifi. 2020. Sparse Smoke Simulations in Houdini. In *ACM SIGGRAPH 2020 Talks (SIGGRAPH '20)*. Article 3, 2 pages.

Omar Zarifi and Christopher Batty. 2017. A positive-definite cut-cell method for strong two-way coupling between fluids and deformable bodies. In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation (SCA '17)*. Article 7, 11 pages.

Runze Zhang, Shanshan Pan, Chenlei Lv, Minglun Gong, and Hui Huang. 2024. Architectural Co-LOD Generation. *ACM Trans. on Graphics* 43, 6, Article 193 (Nov. 2024), 16 pages.

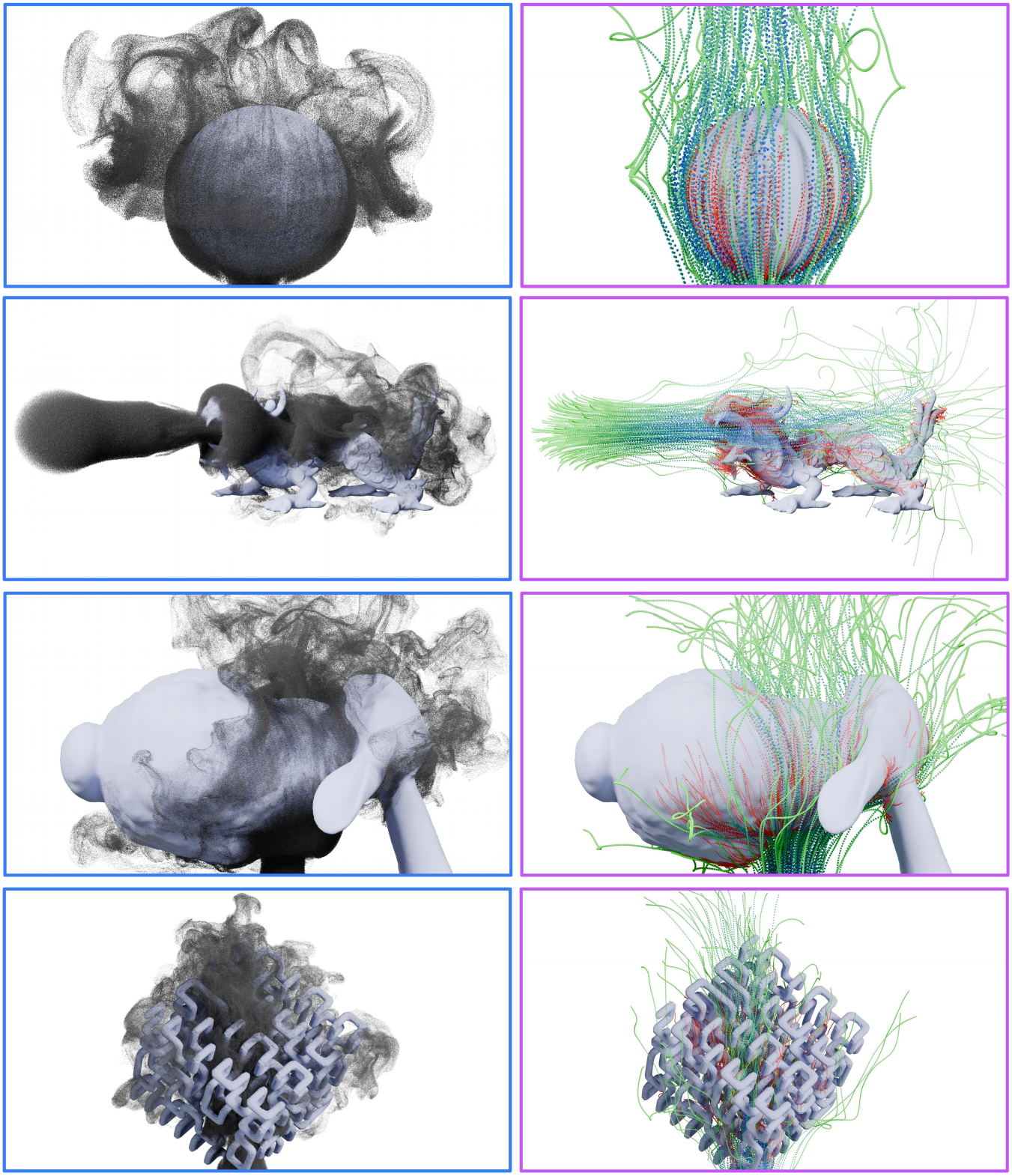


Fig. 12. Smoke simulation results (left) and smoke particle trajectory visualizations colored by velocity magnitude (right), where blue indicates higher speeds. Near-surface flow behavior is visualized using red particles (right). From top to bottom: *Ball*, *Dragon*, *Bunny*, and *Hilbert*.

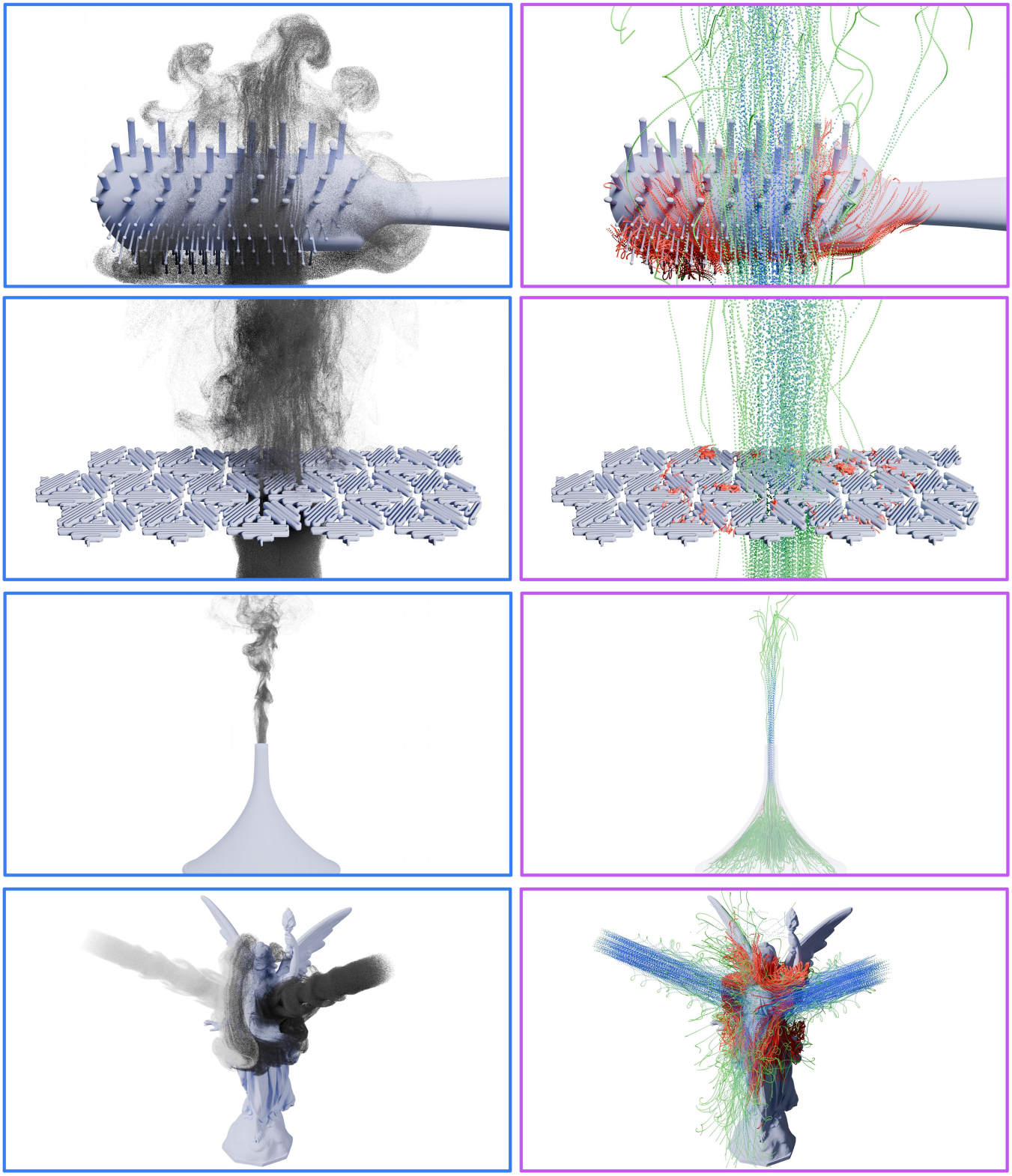


Fig. 13. Smoke simulation results (left) and smoke particle trajectory visualizations colored by velocity magnitude (right), where blue indicates higher speeds. Near-surface flow behavior is visualized using red particles (right). From top to bottom: *Brush*, *Panel*, *Conical Pipe*, and *Lucy*.

Xinxin Zhang, Minchen Li, and Robert Bridson. 2016. Resolving fluid boundary layers with particle strength exchange and weak adaptivity. *ACM Trans. on Graphics* 35, 4, Article 76 (July 2016), 8 pages.
Yongning Zhu and Robert Bridson. 2005. Animating sand as a fluid. *ACM Trans. on Graphics* 24, 3 (July 2005), 965–972.

A Construction of Polynomial Basis

Following standard choice [Beirão da Veiga et al. 2013], we use a basis of scaled monomials for all polynomial spaces, defined as

$$m_{\boldsymbol{\alpha}}(\mathbf{x}) := \left(\frac{x - x_E}{h_E}\right)^{\alpha_1} \left(\frac{y - y_E}{h_E}\right)^{\alpha_2} \left(\frac{z - z_E}{h_E}\right)^{\alpha_3}, \qquad |\boldsymbol{\alpha}| \le k. \tag{35}$$

Here, $\mathbf{x}_E$ denotes the centroid of element E, and h_E is the diameter, defined as the maximum distance between any two vertices of E. The mapping between α and $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \alpha_3)$ is defined as

$$1 \leftrightarrow (0,0,0), \quad 2 \leftrightarrow (1,0,0), \quad 3 \leftrightarrow (0,1,0), \quad 4 \leftrightarrow (0,0,1), \ \ldots$$

B Diffusion Analysis of Naive Prolongation Operators

To analyze and mitigate the diffusion effect of naive prolongation operator, the following sets are defined for a given prolongation matrix P:

$$\begin{aligned} \mathcal{I}(c) &:= \{\, i \mid P(c,i) \neq 0 \,\}, \\ \mathcal{I}'(i) &:= \{\, c \mid P^{\mathsf{T}}(c,i) \neq 0 \,\}, \\ \mathrm{Link}(i) &:= \{\, j \mid A_f(i,j) \neq 0 \,\}. \end{aligned} \tag{36}$$

Here, c denotes a coarse-level DOF index, while i and j denote fine-level DOF indices. The set $\mathcal{I}(c)$ collects fine-level DOFs influenced by the coarse DOF c, $\mathcal{I}'(i)$ collects coarse-level DOFs contributing to the fine DOF i, and $\mathrm{Link}(i)$ encodes the connectivity induced by the fine-level system matrix A_f. It then follows that a coarse-level matrix entry $A_c(i,j) \neq 0$ if and only if

$$j \in \bigcup_{k \in \mathcal{I}(c_i)} \mathcal{I}'\big(\mathrm{Link}(k)\big).$$

This implies that, for the naive prolongation operator, an entry $A_c(i,j)$ becomes nonzero whenever the corresponding DOFs i and j are associated with polyhedral elements that are k-ring neighbors at level $\ell - k$. Such diffusion leads to a rapid growth of nonzero entries on coarse levels.

C Quality Indicator for VEM Cells

The first term,

$$\rho_1^{\mathrm{3D}}(E) = \frac{|k(E)|}{|E|} \prod_{f \in \partial E} \frac{|k(f)|}{|f|}, \tag{37}$$

quantifies the alignment between the cell geometry and its kernel, penalizing degenerate or strongly non–star-shaped elements. The kernel of a element E consists of all interior points from which E is entirely visible. The second term accounts for local geometric scale variations:

$$\rho_2^{\mathrm{3D}}(E) = \frac{\min\left(\sqrt[3]{|E|},\ \min\limits_{f \in \partial E} h_f\right)}{h_E} + \frac{1}{\#f_E} \sum_{f \in \partial E} \frac{\min\left(\sqrt[3]{|f|},\ \min\limits_{e \in \partial f} |e|\right)}{h_f}, \tag{38}$$

which measures the consistency between characteristic length scales across volumetric, face, and edge elements, thereby penalizing strong geometric imbalance within a cell. Finally, the third term captures topological complexity:

$$\rho_3^{\mathrm{3D}}(E) = \frac{4}{\#f_E} + \frac{1}{\#f_E} \sum_{f \in \partial E} \frac{3}{\#e_f}, \tag{39}$$

penalizing cells with an excessive number of faces or faces with many edges. For all three terms above, $\#(\cdot)$ denotes the cardinality of the corresponding set, and $|\cdot|$ denotes the length, area, or volume of the associated geometric entity, as appropriate.