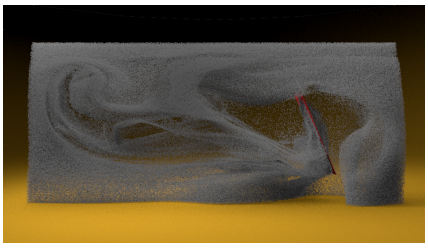
Preserving Geometry and Topology for Fluid Flows with Thin Obstacles and Narrow Gaps

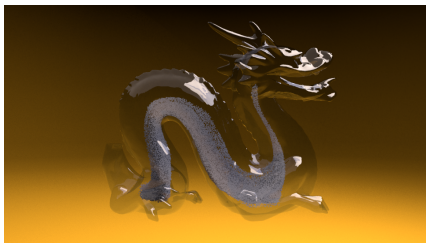
Vinicius C. Azevedo
Instituto de Informática - UFRGS

Christopher Batty
University of Waterloo

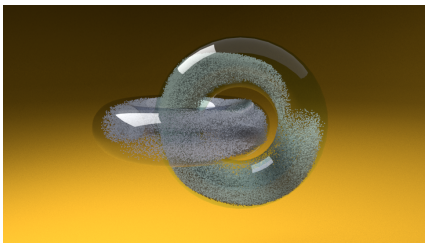
Manuel M. Oliveira
Instituto de Informática - UFRGS



Rotating Paddle: $23 \times 10 \times 6$ grid



Hollow Dragon: $11 \times 9 \times 6$ grid



Linked Tori: $7 \times 7 \times 7$ grid

Figure 1: *Our geometry- and topology-aware boundary treatment supports simulating smooth flows in the presence of thin solid geometry and narrow gaps on* very *coarse grids.*

Abstract

Fluid animation methods based on Eulerian grids have long struggled to resolve flows involving narrow gaps and thin solid features. Past approaches have artificially inflated or voxelized boundaries, although this sacrifices the correct geometry and topology of the fluid domain and prevents flow through narrow regions. We present a *boundary-respecting* fluid simulator that overcomes these challenges. Our solution is to intersect the solid boundary geometry with the cells of a background regular grid to generate a topologically correct, boundary-conforming cut-cell mesh. We extend both pressure projection and velocity advection to support this enhanced grid structure. For pressure projection, we introduce a general graph-based scheme that properly preserves discrete incompressibility even in thin and topologically complex flow regions, while nevertheless yielding symmetric positive definite linear systems. For advection, we exploit polyhedral interpolation to improve the degree to which the flow conforms to irregular and possibly non-convex cell boundaries, and propose a modified PIC/FLIP advection scheme to eliminate the need to inaccurately reinitialize invalid cells that are swept over by moving boundaries. The method naturally extends the standard Eulerian fluid simulation framework, and while we focus on thin boundaries, our contributions are beneficial for volumetric solids as well. Our results demonstrate successful one-way fluid-solid coupling in the presence of thin objects and narrow flow regions even on very coarse grids.

Keywords: fluids, coupling, cut-cell, boundaries, thin solids

Concepts: •**Computing methodologies** → **Physical simulation;**

SIGGRAPH '16 Technical Paper,, July 24-28, 2016, Anaheim, CA,
ISBN: 978-1-4503-4279-7/16/07
DOI: http://dx.doi.org/10.1145/2897824.2925919

1 Introduction

Compelling fluid animations often result from interactions with moving solid boundaries. However, standard grid-based discretizations face difficulties when either the boundaries themselves or the spaces between the boundaries are thin relative to the grid resolution. For narrow flow regions, the challenge is that a typical voxelized view of the domain simply cannot capture them correctly, either topologically or geometrically. For thin boundaries, the same difficulties are exacerbated by the need to prevent flow on one side of an impermeable boundary from erroneously interfering with flow on the opposing side. While in principle one could continually increase the grid resolution until the thin feature or region is fully resolved, this is tremendously expensive and impractical for most animation scenarios. The poor scaling of volumetric simulation has motivated recent efforts to capture as much detail as possible at *free surface boundaries* while using much lower resolution underlying simulation grids [Kim et al. 2009; Wojtan et al. 2010; Bojsen-Hansen and Wojtan 2013; Edwards and Bridson 2014].

Our goal is philosophically similar: we seek to enhance the ability of coarse grid-based Eulerian fluid simulators to resolve interesting flows, but focus instead on *solid boundaries* which may be moving, irregularly shaped, arbitrarily thin, and in close mutual proximity.

We take an embedded boundary or *cut-cell* approach: at each time step, grid cells intersected by the triangle mesh representing the solid boundary are clipped against it, potentially yielding multiple distinct polyhedral sub-cells. The resulting hybrid simulation mesh closely conforms to the geometry of the solid boundary and reduces to a regular grid away from the boundary. Crucially, and in contrast to existing fluid animation methods using regular grids, *our approach preserves the topology of the fluid domain*, including thin solids and slender narrow gaps between nearby solids. The practical advantage this offers is a sharp reduction in the unnecessary coupling between grid resolution and solid boundary topology present in previous work; that is, fluid grid resolution can be artistically adjusted solely to achieve the desired balance of fluid detail and computational cost, without concern for whether an inaccurate solid discretization will inadvertently disconnect or merge flow regions in the process.

We first introduce a topologically-accurate, graph-based discretization for the pressure projection on the cut-cell mesh which can resolve flows in difficult regions. Furthermore, it offers greater fidelity than prior work on thin solids: it better accounts for the sub-

grid geometry of the boundary, correctly recovers free-slip boundary conditions, and is consistent with existing cut-cell approaches for *volumetric* objects (e.g., [Batty et al. 2007; Ng et al. 2009]).

Secondly, to improve the handling of advection near boundaries we develop a *conforming* velocity interpolant on arbitrarily-shaped polyhedral cut-cells by relying on spherical barycentric coordinates (SBC) [Langer et al. 2006]. This allows flow characteristics to more closely respect boundary geometry than is possible with standard, boundary-oblivious linear or cubic interpolation schemes, particularly in narrow regions.

Lastly, we augment our approach with a tailored PIC/FLIP [Zhu and Bridson 2005] advection scheme. Beyond its usual ability to reduce numerical dissipation, this resolves a lingering difficulty with semi-Lagrangian advection in the context of thin moving boundaries. Specifically, grid cells swept over by solids lack valid velocity information after semi-Lagrangian advection [Guendelman et al. 2005], and must be filled back in by extrapolating from valid cells that may be arbitrarily far away. Our use of Lagrangian particles ensures that velocity data flows coherently with the boundaries themselves, so that extrapolation is not required.

These enhancements substantially improve the detail that can be achieved when simulating fluids interacting with solid boundaries, while readily integrating into the dominant Eulerian staggered grid fluid pipeline. Figure 1 shows examples of fluid animations containing thin obstacles and gaps created with our technique on very coarse grids. On the left, a very thin paddle successfully stirs smoke. The image in the center shows smoke propagating through a narrow tunnel in a low-resolution grid. On the right, we see flow through two linked tori without mutual interaction.

To summarize, the **technical contributions** of our work include:

- The identification of key limitations of existing thin solid and thin gap treatments, due to voxelized geometry and standard interpolation strategies;
- A symmetric, graph-based cut-cell pressure projection method that preserves the domain topology (Section 4). It is the first to properly handle both thin obstacles and thin gaps between obstacles within coarse 3-D grid cells, allowing the use of less costly grids to animate flows in difficult geometries;
- An improved velocity interpolation scheme in polyhedral cut-cells based on spherical barycentric coordinates (Section 5), allowing flows to better respect irregular solid boundaries;
- A technique to improve velocity advection near thin moving obstacles (Section 6). By combining Lagrangian PIC/FLIP particles with our cut-cell scheme, velocity information is correctly propagated despite the presence of moving geometry.

1.1 Overview

The structure of our technique is outlined in Algorithm 1, which generally follows the hybrid particle-grid approach of Zhu and Bridson [2005]. After reviewing related work, we describe our cut-cell mesh structure (Section 3) and explain how pressure projection (Section 4), interpolation (Section 5) and advection (Section 6) are modified to best exploit it. We describe our algorithm in 3-D; the 2-D analogy is straightforward.

Algorithm 1 Main Loop

```
while simulating do
    Advect particles (Section 5) and advance solid position
    Generate cut-cell mesh (Section 3)
    Transfer particle velocities to the mesh (Section 6)
    Add external forces to the mesh
    Perform pressure projection on the mesh (Section 4)
    Update particle velocities from the mesh (Section 6)
end while
```

2 Previous Work

2.1 Thin Solid Boundaries

While thin boundaries often arise in two-way coupling, we focus our review on aspects relevant to the one-way (solid-to-fluid) coupling problem addressed by our work.

The coupling of fluid to thin boundaries in computer graphics was first addressed by Guendelman et al. [2005]. Their approach voxelized the geometry of thin shells onto the regular grid, and used a one-sided extension of trilinear interpolation based on raycasting to avoid mixing data from the opposite side of a boundary. They also proposed an extrapolation approach to fill in data for fluid regions that are swept over and invalidated by moving boundaries. Later work by Robinson-Mosher et al. [2008; 2009] adopted essentially the same one-sided interpolation mechanism. A similar raycasting strategy has been applied to compressible flows in computational fluid dynamics [Wang et al. 2012].

Robinson-Mosher et al. [2008] used a mass-lumping technique for two-way coupling of thin shells to fluid on a regular grid. This scheme sacrifices free-slip velocities even in the inviscid limit, so the same authors proposed the use of ghost-velocities and a constraint-based formulation to restrict only the normal component of velocity [Robinson-Mosher et al. 2009]. Both methods use a voxelized boundary approximation, and thus the topology of the fluid domain used by the pressure solver is often incorrect in tight configurations. Voxelization also leads the solid boundary velocity constraints to be applied at grid face centres rather than on the actual boundary itself. Qiu et al. [2015] proposed a two-way rigid-body-fluid coupling scheme that extends the voxelized approach to thin gaps using lower-dimensional advection and extra degrees of freedom, though it does not consider thin objects.

Boundary-conforming Eulerian tetrahedral meshes (e.g., [Feldman et al. 2005; Klingner et al. 2006; Elcott et al. 2007]) could *potentially* simplify the treatment of thin boundaries during pressure projection, at the cost of repeated and potentially costly remeshing, but to our knowledge this has not been explicitly considered. The closest is the work of Chentanez et al. [2006], who simulated the coupling of fluid to deformable shells of modest thickness discretized with tetrahedra using a conforming mesh approach. To reduce meshing costs for liquid animation, Chentanez et al. [2007] later relied on the efficiency of isosurface stuffing [Labelle and Shewchuk 2007]; however, isosurface stuffing conforms to an approximate isosurface rather than the exact solid geometry. In general, while conforming meshes simplify the pressure projection, their use in Eulerian schemes does not inherently resolve interpolation and advection issues near thin boundaries. In contrast to Eulerian methods, purely Lagrangian methods that rely on conforming tetrahedralizations of both fluid and solid are also possible [Misztal et al. 2010; Clausen et al. 2013], and may better avoid these issues; again, this does not appear to have been studied.

While beyond the scope of this work, thin objects have also been coupled to SPH simulations (e.g., [Lenaerts and Dutré 2008]). Another interesting alternative uses history-based forces to approximate the effects of fluid on submerged cloth [Ozgen et al. 2010]; this does not extend to scenarios where the fluid motion itself is also of interest.

2.2 Embedded Boundary Methods

Roble et al. [2005] proposed a two-dimensional finite volume-like technique for irregular static boundaries, similar to much earlier work by Purvis and Burkhalter [1979], in which the usual Poisson stencil is augmented with per-face weights that account for the fluid fraction of each face. Batty et al. [2007] presented a closely related, variational technique that enabled stable two-way coupling in 3-D with irregular volumetric rigid bodies. Ng et al. [2009] showed that the finite volume variant of this scheme yields second-order accurate pressures and first-order velocities. (This complements the ghost fluid method for *free surfaces* [Enright et al. 2003] which achieves the same.) In essence, *our discretization applies and generalizes the work of Ng et al. to thin boundaries and thin gaps*.

Colella and collaborators [Johansen and Colella 1998; Schwartz et al. 2006] developed a similar cut-cell method that additionally interpolates velocities to lie at the centroids of partial faces. This achieves second-order accurate *velocities* at the expense of more complex stencils; however, these stencils yield non-symmetric systems and cannot be applied in narrow regions. Day et al. presented an interesting partial extension of this idea to thin boundaries in two dimensions, through the use of a more general graph structure and extra ghost samples on the grid [Day et al. 1998]. Our pressure projection draws inspiration from this method, but differs in a few key respects. *We achieve a symmetric positive-definite system, provide a direct extension to three dimensions, and support an arbitrary number of disjoint components per cell*.

Crockett et al. [2011] discussed the related idea of "multi-cells" which arise during coarsening steps of a multigrid scheme for Poisson problems on irregular domains; work by Dick et al. [2016] is similar in spirit. Weber et al. [2015] developed a multigrid solver for the scheme of Ng et al., ensuring consistent discretization across grid levels, but did not consider multi-cells or the treatment of thin solids. Hellrung et al. [2012] presented a more complex virtual node discretization for 3-D Poisson problems with discontinuities, assuming a level-set description of domain boundaries which effectively restricts the method to closed regions that do not possess thin boundaries or gaps.

Ferstl et al. [2014] used a cut-cell tetrahedra-based finite-element scheme with a multigrid solver, and similarly preserved the free surface topology during coarsening, though solid boundaries were treated as voxelized. Edwards et al. [2014] proposed an adaptive discontinuous Galerkin scheme on cut cell meshes to handle detailed free surface flow on coarse grids, potentially involving multiple disjoint liquid components per original cell; they did not discuss thin solids or thin gaps.

Outside of the fluid setting, topology-aware strategies have been applied to simulate the dynamics of elastic deformable objects possessing multiple distinct deforming components inside a single finite element [Teran et al. 2005; Nesme et al. 2009]. Though conceptually related, they are inapplicable to the problem we consider.

2.3 Velocity Reconstruction and Interpolation

Staggered-grid projection methods recover only the face-normal components of velocity rather than full vectors; this slightly complicates interpolation and advection. In the regular grid case, multilinear interpolation can be applied on each velocity component independently. However, for more general unstructured or polyhedral meshes, full velocities must first be reconstructed before interpolating, typically through least squares fitting, as done by previous work on tetrahedral and Voronoi grids [Feldman et al. 2005; Klingner et al. 2006; Elcott et al. 2007; Sin et al. 2009; Batty et al. 2010; Brochu et al. 2010]. We use a least squares fit to recover nodal velocities from face fluxes on our polyhedral cells, which simplifies velocity interpolation during advection.

Various velocity interpolation schemes have been proposed for use during the advection step, the most common being simple bi/trilinear interpolation on a regular grid [Stam 1999]. Higher-order extensions have been used to improve the retention of vorticity [Fedkiw et al. 2001; Selle et al. 2008].

Guendelman et al. [2005] were the first to directly address the interpolation issues raised by thin boundaries. Subsequent related work by Robinson-Mosher et al. [2008; 2009] relied on the same interpolation technique. Raycasting is used to determine visibility between an interpolation point and the position of a velocity sample it would depend on; one-sided interpolation can then be performed using only the visible data to robustly avoid polluting the result with data from the opposite side of a thin boundary. However, since basic trilinear or tricubic interpolation do not possess knowledge of the solid position, fluid trajectories typically still cross boundaries; this necessitates the frequent use of collision-processing during advection to prevent data crossing over.

On staggered unstructured tetrahedral meshes, the velocity reconstruction approaches are first used to determine velocities at desired nodal points; these can then be applied within a mesh-based barycentric interpolant. Given the velocities at tetrahedra centres (i.e., Voronoi vertices), generalized barycentric interpolation is applied over the convex Voronoi elements [Klingner et al. 2006; Elcott et al. 2007].

In the above methods, polyhedral interpolants were used for the purpose of avoiding oversmoothing velocities, as compared to interpolating over tetrahedra. By contrast, our primary motivation for using polyhedral interpolation is that it enables the interpolated velocity to closely conform to the geometry of solid boundaries. Rosatti et al. [2005] presented a related two-dimensional technique that fits boundary-respecting linear velocity fields to the triangular, trapezoidal, and pentagonal cells resulting from the usual marching-squares cases applied to an implicit representation of the solid boundary. *Our approach clips the regular grid against the solid boundary triangle mesh, yielding arbitrary polyhedral cells*. We can then use an interpolant that handles non-convex polyhedra, i.e., spherical barycentric coordinates [Langer et al. 2006].

3 The Cut-Cell Mesh

Given a triangle mesh representing the geometry of the solid boundary, we perform clipping on all cells intersected by this boundary. Each affected original grid cell may give rise to one or more boundary-conforming *polyhedral* sub-cells. Clipping with triangle meshes is a well-studied problem (e.g., [Aftosmis et al. 1998; Sifakis et al. 2007; Wang et al. 2014]), most recently used by Edwards and Bridson [2014] to support detailed liquid free surfaces. We therefore refer the reader to these works for implementation details on generating cut-cell meshes, and simply summarize the properties of the resulting mesh.

A principal difference between our cut-cell meshes and those used by Edwards and Bridson is that we retain sub-cells on both sides of the triangle mesh geometry. The solid geometry is also not required to be a "closed" surface, and therefore the triangle mesh may cut only partway through a cell. In this case, we subdivide the faces through which it crosses, but do not partition the cell itself. We will refer to the resulting faces as *dangling interior faces*. We will refer to mesh faces that connect two fluid (sub-)cells as *fluid faces*; these will always be axis-aligned. New faces produced by clipping against the solid boundary will be called *solid faces*. We do not tetrahedralize the resulting polyhedra, so cell faces may be general

planar polygons. Cells that are not intersected by the geometry are left untouched, so as to be efficiently and conveniently treated with standard methods.

Figure 2 illustrates these cut-cell concepts in 2-D, for two infinitesimally thin solid boundaries with fluid on either side. In (a), the thin boundaries are represented by polylines, shown in green with bright green nodes. The original regular grid is shown in gray. Part (b) illustrates the boundaries (in red) resulting from the raycasting or voxelized view used by previous work [Guendelman et al. 2005; Robinson-Mosher et al. 2008; Robinson-Mosher et al. 2009]. Both the geometry and topology of the fluid domain are sacrificed: the gap between the two solid boundaries has been entirely collapsed away. Part (c) illustrates our cut-cell mesh with the new vertices created during clipping (shown in black). Under our cut-cell view, both the thin gap and the detailed geometry of the boundary are maintained. Part (d) uses a graph (blue) to illustrate the neighbour relationships between the resulting sub-cells. The segments in the partially cut upper-left and lower-left cells are examples of *dangling interior faces*; notice that, as illustrated in the graph view, these partially cut cells are assigned only a single pressure sample although their faces are subdivided.

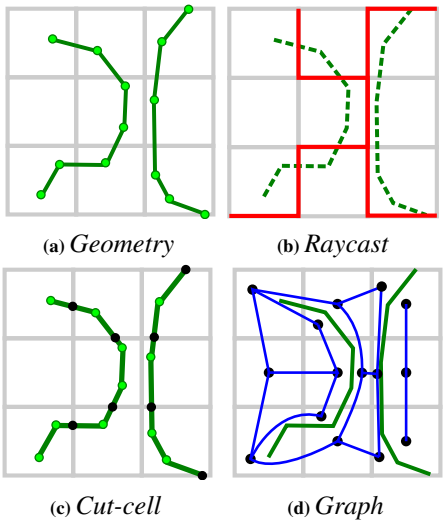


Figure 2: *(a) Sub-grid thin boundaries (green) are represented by a polyline mesh in 2-D. (b) Voxelization/raycasting yields inaccurate axis-aligned boundaries (red). (c) Clipping the grid against the solid boundary mesh instead yields a cut-cell mesh with multiple distinct sub-cells, with new mesh nodes shown in black. (d) The connectivity relationships between sub-cells can be visualized as a graph (blue).*

This cut-cell mesh, with its support for multiple disjoint solid components and multiple sub-cells in each grid cell, forms the infrastructure with which we will handle narrow gaps and thin solids. An example is shown for the Dragon in Figure 3.

4 Graph-Based Pressure Projection

Cut-Cell Pressure Projection The standard pressure projection step solves the Poisson problem $\frac{\Delta t}{\rho}\nabla \cdot \nabla p = \nabla \cdot \mathbf{u}^*$, in order to find the pressure field that will correctly convert the intermediate velocity field, $\mathbf{u}^*$, into the nearest incompressible field, $\mathbf{u}$. Having found the pressure field p, its gradient is subtracted from the velocity field: $\mathbf{u} = \mathbf{u}^* - \frac{\Delta t}{\rho}\nabla p$.

Our approach to discretizing this problem on the cut-cell mesh extends previous variational [Batty et al. 2007] and finite volume cut-cell [Roble et al. 2005; Ng et al. 2009; Batty et al. 2010] techniques for *volumetric* solids, which account for the flow through each face of a given grid cell adjusted for the area of the faces that are blocked by a solid obstacle.

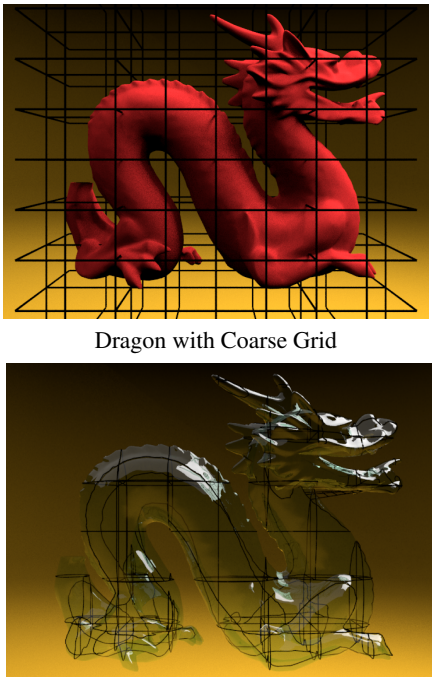


Figure 3: *(Top) The dragon solid geometry, shown with the regular grid superimposed. (Bottom) The network of curves generated by intersecting the two, with the dragon rendered transparent.*

In particular, we begin with the scheme of Ng et al. [2009] as the basis of our approach as it *yields symmetric positive definite linear systems and pressure solutions that converge with second-order spatial accuracy*. The associated discrete divergence measure is:

$$\nabla \cdot \mathbf{u} \approx \frac{\sum_i A_i(\mathbf{u}\cdot\mathbf{n})_i + \sum_j A_j(\mathbf{u}_{solid}\cdot\mathbf{n}_{solid})_j}{V_{cell}}, \qquad (1)$$

where the index i runs over all fluid faces of a cell, and j runs over all solid faces. A_k indicates the area of the k-th face, $\mathbf{u}$ is the fluid velocity, $\mathbf{u}_{solid}$ is the solid velocity, $\mathbf{n}$ is the fluid face normal vector, $\mathbf{n}_{solid}$ is the solid face normal vector, and V_{cell} indicates the volume of the cell. Face normals are assumed to be oriented outwards. As usual, scaling each row of the discrete Poisson problem by its corresponding cell volume cancels volume terms in the system; we require only face areas (e.g., [Losasso et al. 2004]). Following Guendelman et al. [2005] one must also take care to set the velocity for the solid boundary condition to be the *effective* velocity computed over the subsequent timestep rather than its instantaneous/analytical velocity, to ensure that the resulting end-of-step velocities sync with the motion of the solid during advection on the next step; their paper provides further discussion.

Pressure gradients are computed using standard centered differences between *cell-centered pressures*, $\nabla p \approx \frac{p_{i+1}-p_i}{\Delta x}$, *even near cut-cell faces*. Given these discrete divergence and gradient operators, the Poisson problem can be directly discretized on the usual staggered grid. Perhaps surprisingly, Ng et al. clearly show that this projection scheme correctly converges even though the geometric centers of cells and the midpoints of faces often lie outside the actual fluid domain (see Figure 4, top-left). This feature conveniently

preserves many of the benefits of the structured regular grid (symmetry, positive-definiteness, primal-dual orthogonality, second-order accurate pressures) as we extend it to more general topologies below.

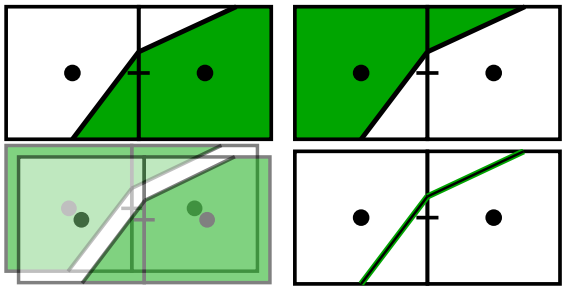


Figure 4: *Top-left: The method of Ng et al. for embedded volumetric solid boundaries (green) converges despite using active face midpoints (black dash) and cell centers (black disks) lying outside the fluid domain (white). Top-right: A complementary dual geometry, created by swapping fluid and solid domains, can also be easily simulated with Ng's method. Bottom-left: By conceptually superimposing the top two scenarios and duplicating the required degrees of freedom, a pressure projection can be performed on the thin solid, shown at the bottom-right, without interference across it.*

Topology-Aware Pressure Projection The method of Ng et al. implies a few restrictions. It assumes a level set description of the geometry, which limits it to volumetric objects and fluid regions larger than a grid cell width to guarantee a faithful topological description of the domain. The strictly regular underlying grid structure also means that each cell contains only a single active region and corresponding pressure . We seek to relieve these restrictions.

To extend this strategy to multiple distinct flow regions within a single regular grid cell, as produced by our mesh clipping strategy, we take inspiration from recent virtual node [Molino et al. 2004; Hellrung et al. 2012] and topology-preserving [Teran et al. 2005; Nesme et al. 2009] schemes. We allow multiple disjoint active *sub-cells* within a single original cell, with additional pressure and velocity degrees of freedom that conceptually coincide for consistency with Ng's discretization (see Figure 4). We assign one pressure to each sub-cell, placing it at the original cell center's location (i.e., *not* at sub-cell centroids). Each original fluid face of the grid has multiple fluid sub-faces which connect sub-cells of adjacent regular cells together; each sub-face is assigned a velocity degree of freedom that is geometrically positioned at the regular cell face midpoint (i.e., *not* at the sub-face midpoint). This yields a more general graph structure (see Figure 2d) on which we can perform the pressure projection, yet the gradient and divergence operators remain axis-aligned. Table 1 gives the explicit matrix representation of our discrete Poisson equation for the small 2-D scenario shown in Figure 5. In large examples, most of the mesh will exhibit the usual banded Poisson matrix structure, with a few additional unstructured entries to treat regions involving cut geometry.

Primal-Dual Orthogonality As highlighted by Batty et al. [2010], orthogonality of the discrete gradient estimates with respect to the face-normal velocity components is key to preserving accuracy in staggered finite volume approaches. For example, staggered octree schemes [Losasso et al. 2004] can lose accuracy at T-junctions due to non-orthogonal gradients between large and small neighbor cells, without a more careful treatment [Losasso et al. 2005]. By contrast, the gradients we use between sub-cells are always computed between the *geometric centers* of their original grid cells, and therefore preserve orthogonality with respect to the grid faces (see Figure 6). This property is key to both our method and that of Ng et al.

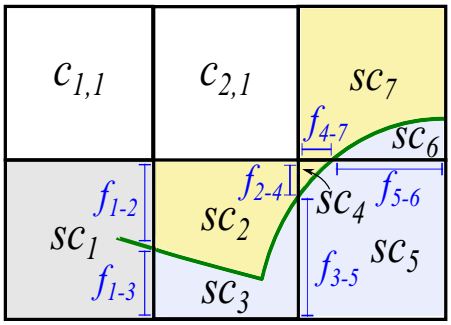


Figure 5: *Geometry and notation used in our 2-D Poisson matrix example (Table 1). The solid thin boundary is shown in green. $c_{i,j}$ is a regular grid cell at row j, column i. sc_k is sub-cell k. $f_{a\text{-}b}$ is the fraction of the fluid edge shared by sub-cells a and b.*

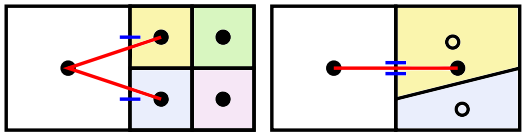


Figure 6: *Left: Naïve octree discretizations yield face fluxes (blue) and pressure gradients (red) that are not aligned. Right: Following Ng, our cut-cell discretization co-locates all sub-cell pressures at grid cell centers (filled black circle) rather than sub-cell centroids (empty black circles). Thus our "T-junction-like" branching preserves orthogonality and avoids artifacts. Similarly, face fluxes are conceptually stored at original face midpoints (blue), rather than at sub-face midpoints.*

5 Conforming Interpolation on Cut-Cells

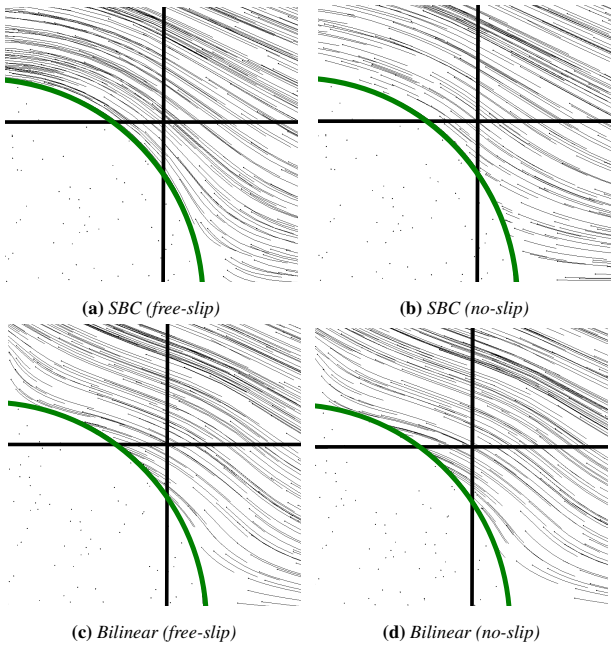
Both PIC/FLIP and semi-Lagrangian advection schemes rely on the ability to interpolate velocities at arbitrary points in the fluid domain. The interpolants used to estimate pointwise velocity values in grid-based methods typically rely on simple piecewise linear or cubic approximations [Fedkiw et al. 2001]. Though effective in free flowing regions, such interpolants are fundamentally oblivious to the domain geometry, regardless of either the order of the interpolant or the accuracy of the pressure projection. As a result, the interpolated fluid velocities do not necessarily satisfy the desired no-penetration boundary condition $\mathbf{u}_{fluid} \cdot \mathbf{n} = \mathbf{u}_{solid} \cdot \mathbf{n}$, but are instead often directed *towards and through* the boundary, a fact which is particularly problematic for thin solids. The most common treatment is to apply collision detection to directly clip particle trajectories against solids (e.g., [Fedkiw et al. 2001; Guendelman et al. 2005]), although this can exacerbate artificial clumping of particles and other data [Rasmussen et al. 2004].

We instead aim to construct an improved interpolant so that the fluid velocities themselves better respect the solid geometry, and reliance on explicit collision-processing can be reduced. Figure 7 uses a streamline visualization to compare one-sided bilinear interpolation [Guendelman et al. 2005; Robinson-Mosher et al. 2009] against our interpolation method. Although both approaches carefully avoid mixing data from the wrong side of the thin boundary, the latter yields a velocity field that conforms more closely to the boundary, reduces grid-dependence, and can easily be set to satisfy either a free-slip or no-slip condition as desired. By contrast, the results for bilinear interpolation do not align with the boundary and exhibit nearly identical results under free-slip and no-slip conditions.

We describe our interpolation approach below, and use it during the

$$
\begin{array}{c}
 \\
c_{1,1} \\
c_{2,1} \\
sc_1 \\
sc_2 \\
sc_3 \\
sc_4 \\
sc_5 \\
sc_6 \\
sc_7
\end{array}
\begin{array}{c}
\begin{array}{ccccccccc}
c_{1,1} & c_{2,1} & sc_1 & sc_2 & sc_3 & sc_4 & sc_5 & sc_6 & sc_7
\end{array} \\
\begin{pmatrix}
2 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 3 & 0 & -1 & 0 & 0 & 0 & 0 & -1 \\
-1 & 0 & \sum f_1 & -f_{1\text{-}2} & -f_{1\text{-}3} & 0 & 0 & 0 & 0 \\
0 & -1 & -f_{1\text{-}2} & \sum f_2 & 0 & -f_{2\text{-}4} & 0 & 0 & 0 \\
0 & 0 & -f_{1\text{-}3} & 0 & \sum f_3 & 0 & -f_{3\text{-}5} & 0 & 0 \\
0 & 0 & 0 & -f_{2\text{-}4} & 0 & \sum f_4 & 0 & 0 & -f_{4\text{-}7} \\
0 & 0 & 0 & 0 & -f_{3\text{-}5} & 0 & \sum f_5 & -f_{5\text{-}6} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -f_{5\text{-}6} & \sum f_6 & 0 \\
0 & -1 & 0 & 0 & 0 & -f_{4\text{-}7} & 0 & 0 & \sum f_7
\end{pmatrix}
\end{array}
\begin{pmatrix}
p_{1,1} \\ p_{2,1} \\ p_{sc_1} \\ p_{sc_2} \\ p_{sc_3} \\ p_{sc_4} \\ p_{sc_5} \\ p_{sc_6} \\ p_{sc_7}
\end{pmatrix}
=
\begin{pmatrix}
d_{1,1} \\ d_{2,1} \\ d_{sc_1} \\ d_{sc_2} \\ d_{sc_3} \\ d_{sc_4} \\ d_{sc_5} \\ d_{sc_6} \\ d_{sc_7}
\end{pmatrix}
$$

Table 1: *The symmetric cut-cell pressure projection matrix that results from the 2-D configuration shown in Figure 5, assuming Neumann boundary conditions on the domain perimeter. $c_{i,j}$ represents a regular grid cell at row j, column i. sc_k is sub-cell k. $f_{a\text{-}b}$ is the fraction of the fluid edge shared by sub-cells a and b. $\sum f_k$ is the sum of all fluid-edge fractions shared by sub-cell sc_k. $p_{i,j}$ and $d_{i,j}$ are, respectively, the pressure and divergence at grid cell $c_{i,j}$. p_{sc_k} and d_{sc_k} are the pressure and divergence at sub-cell sc_k.*



(a) *SBC (free-slip)* **(b)** *SBC (no-slip)*

(c) *Bilinear (free-slip)* **(d)** *Bilinear (no-slip)*

Figure 7: *Streamlines of a velocity field obtained using different one-sided interpolation schemes, with a stationary flow on one side. (a) SBC interpolation from nodal velocities using* free-slip *boundary conditions leads to streamlines that flow smoothly along the boundary. (b) SBC with* no-slip *likewise conforms to the boundary, but both tangential and normal velocities drop to zero precisely at the boundary. (c) and (d) show one-sided bilinear interpolation from values stored at the regular grid corners for both free-slip and no-slip. The bilinear results exhibit grid-dependence, and do not differ appreciably from one another. Moreover, since the interpolant is non-conforming, the relevant velocity components do not drop to zero on the boundary curve.*

particle advection step of Section 6.

5.1 Polyhedral Cut-Cell Interpolation

Away from solid boundaries, we apply standard trilinear interpolation to the regular grid velocities. For polyhedral (sub-)cells abutting the solid geometry, we will first reconstruct nodal velocity values (Section 5.2), and use these values for interpolation.

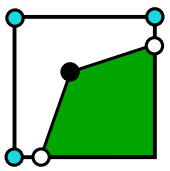
To perform interpolation over potentially non-convex polyhedra with general planar polygonal faces, we make use of spherical barycentric coordinates (SBC) [Langer et al. 2006], which provide a convenient generalization of standard barycentric coordinates to this case. We select SBC over the more widely-known mean value coordinates [Ju et al. 2005], because SBC supports polygonal rather than triangular faces. This method is effective for the vast majority of cut cells, and the result is a nicely conforming interpolant.

Unfortunately, SBC cannot be readily applied to the comparatively small set of cells containing dangling interior faces, as this represents a degenerate configuration (essentially two coincident but oppositely oriented faces). The simplest way to treat this is to just thicken or extrude the input geometry into a very slim volume before simulating. This naturally eliminates problematic dangling interior faces, allowing SBC to work as usual. However, while the geometry is thin it is no longer of infinitesimal width.

The alternative is to try to construct an interpolant that is effective in the presence of the problematic dangling faces. We experimented with various approaches, including visibility-aware SPH interpolation, visibility-aware Shepard [1968] (i.e., inverse distance-weighting with a raycast check of mutual visibility) interpolation, or simply ignoring the dangling geometry altogether and reverting to trilinear interpolation similar to previous work. None of these choices is entirely satisfactory, because none ensure a velocity field that is consistent with interpolation on neighbouring cells, conforms to the interior geometry, and avoids mixing data from opposing sides of the geometry. For examples involving truly infinitesimal width geometry, we used the Shepard interpolation approach; we highlight this as an interesting challenge for future work.

5.2 Velocity Reconstruction

Free-Slip Case We distinguish two cases for reconstructing nodal velocities on cut-cells: *free-slip* and *no-slip*. We consider free-slip first, in which the solid velocity determines the normal component, and fluid velocities must dictate tangential components.



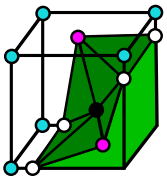
We define three types of cut-cell nodes, illustrated to the right. *Fluid nodes* (cyan) are the original nodes of the regular grid outside the object, for which all incident faces are axis-aligned fluid faces. *Solid nodes* (black) are nodes on the solid for which all incident faces are solid faces. Finally, *mixed nodes* (white) are nodes incident on both solid and fluid faces generated by the

clipping process. Recovering a full velocity at fluid nodes is trivially done by averaging from the staggered data on the incident fluid faces.

For mixed nodes, we apply a weighted least squares fit to the normal velocity components corresponding to the incident faces (solid and fluid) on the same side of the solid boundary, similar to previous work [Feldman et al. 2005]. We use the (inverse) distances from the node to the face centre as the weights. The system can occasionally be underdetermined if a mixed node has nearly co-planar faces; however, this can be compensated for by incorporating extra face-normal velocity samples from additional nearby faces.

If all of a node's incident faces are solid faces, we extrapolate from the nearby mixed nodes for which a valid velocity has been reconstructed as described above; let us call these *valid nodes*. We call *invalid nodes* the solids nodes for which we have yet to assign a velocity. In 2-D, we perform this extrapolation by simply linear interpolating along the solid boundary curve inside the cell.

In 3-D, the geometry is a set of surface triangles rather than a piecewise linear curve, and we still have solid nodes (black) and fluid nodes (cyan). However, we also distinguish two types of mixed nodes: *edge-mixed* nodes (white), which lie on edges of the original grid, and *face-mixed* nodes (magenta), which lie on faces of the original grid. Starting from reconstructed velocity data at edge-mixed nodes, we linearly interpolate along the solid boundary curves so that every face-mixed node on the boundary curve has valid data. We label the internal (solid) nodes of this surface patch to be initially invalid, and perform a simple iterative averaging approach to extrapolate into the interior solid geometry: all invalid nodes with a valid neighbour are set to the average of the valid neighbours, and then marked as valid. This process is iterated until no invalid nodes are left. Once all interior nodes have valid data assigned, we perform a few additional iterations of repeated averaging to smooth the velocities towards a steady distribution. To avoid the damping of velocities introduced by this averaging process, we perform this step for vector magnitudes and directions separately, re-combining them at the end.



We further improve the degree to which the interpolated velocity remains tangent to the solid by directly projecting out the normal component of the relative velocity between the solid and the fluid for mixed and solid nodes (similar in spirit to *constrained velocity extrapolation* [Houston et al. 2003; Rasmussen et al. 2004]). This amounts to computing a new fluid velocity as

$$\mathbf{u}'_{fluid} = \mathbf{u}_{fluid} - ((\mathbf{u}_{fluid} - \mathbf{u}_{solid}) \cdot \mathbf{n}_{solid})\mathbf{n}_{solid}. \quad (2)$$

We observe that even if nodal velocities are projected to be orthogonal to vertex normals, the interpolated flow may *still* clearly cross boundaries if the geometry is sharply concave. In 2-D, we further handled this by treating sharp corners with a no-slip condition, and smooth curves with the preceding approach; this can be observed in our results. However, in 3-D such a treatment is not straightforward, and we have not pursued it.

We do not reconstruct full velocities at face-mixed nodes, as they may lack enough data to reconstruct a full 3-D fluid velocity. However, there are cases where a boundary loop consists of only face-mixed nodes, and no edge-mixed nodes, e.g. a cylindrical tube with diameter less than a grid cell width cutting horizontally through a cell face. The face-mixed nodes on the boundary of the cylinder have only one fluid face component, and the solid normal contribution may only reliably determine one of the remaining two axes. At present, we are therefore limited to scenarios in which each such a solid geometry patch includes at least one edge-mixed node on its boundary. (A reasonable approach in severely under-resolved cases would be to reconstruct the available dimensions, and set the the remaining dimension to zero. In the cylindrical example, the missing dimension would correspond to circular rotations around the cylinder's dominant axis, which is not provided by the fluxes along the axis or the solid normal velocities perpendicular to it.)

No-Slip Case If no-slip interpolation is preferred for visual purposes, fluid nodes are again treated using least squares, but all mixed and solid nodes are directly assigned the solid velocity, including its tangential component. However, note that no-slip conditions will tend to rapidly damp out relative tangential flow in very slender gaps. This is because fluid nodes are the only nodes that no-slip conditions do not effect, and narrow gaps may contain relatively few such nodes. Hence, velocities in these sub-cells will be totally dominated by the solid, and will halt the flow entirely near static solids; an example can be seen in Figure 10(d). (No-slip can also lead to extra particle clumping, because it is fundamentally inconsistent with the inviscid free-slip condition inherent in the pressure solve.)

Figure 8 illustrates the case of parallel line segments aligned with the flow direction. Free-slip (top) allows the flow to pass undisturbed, while no-slip (bottom) causes some drag and deflection in the velocity field. For free-slip, no velocity dissipation happens, regardless of the number of parallel lines per cell, as long as a sufficient number of particles is available (Figure 8 top). Table 4 shows, for a fixed 2D grid resolution (16×10), performance figures for 1, 8, and 64 lines per cell over 5 cells (Figure 8 top). In such a scenario, as the percentage of cut cells relative to the regular 160 grid cells increases from 5% to 123.12% ($24.62\times$), the cut-cell generation time increases $38\times$. Even for the 64-line case, the total time is still dominated by pressure projection.

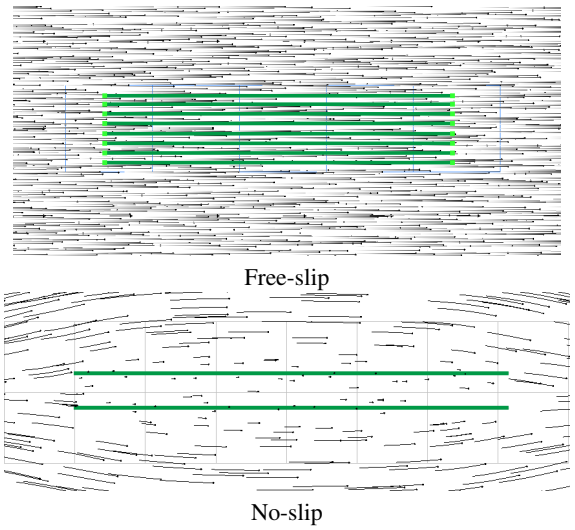
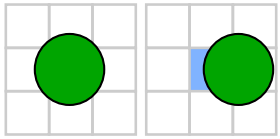


Figure 8: *Horizontal straight segments aligned with the flow. (Top) Free-slip allows the flow to continue undisturbed. (Bottom) No-slip causes drag on the fluid and a deflection in the flow.*

6 Advection on Cut Cells

Having described our cut-cell interpolation method, we can use it to trace out particle trajectories using forward Euler. This allows us to construct a cut-cell-aware particle-based advection

scheme that addresses an issue faced by previous Eulerian and semi-Lagrangian schemes in the presence of moving boundaries. Specifically, solid boundaries that sweep over the stationary Eulerian grid leave behind cells lacking velocity data. Consider the inset example: a volumetric circular solid (green) translates past to reveal the formerly inactive central cell (light blue). It suddenly becomes active again and its faces must be assigned valid velocity data before continuing the simulation. Mittal and Iaccarino [2005] dub these "freshly-cleared" cells. While physically, fluid velocities would simply advect along with the object and fill in the missing data, this is not true in the discrete case for Eulerian and semi-Lagrangian schemes.



For large volumetric solids, semi-Lagrangian advection suffices, since the velocities previously extrapolated into the solid are often reasonable. However, for relatively thin boundaries, (sub-)cell regions often go from one side of a moving solid boundary to another *in a single step*, and there is no extrapolated velocity already present "inside" the object. Using semi-Lagrangian advection, the end-of-trajectory velocity value which one would ordinarily use to begin backtracing *from* lies on the *wrong side of the boundary*, where the fluid may be flowing in entirely the wrong direction.

Guendelman et al. [2005] observe this issue, and instead reinitialize these cells by using an iterative averaging procedure to extrapolate data from nearby fluid cells on the same side which were not swept over, and hence contain valid data. However, this choice has two shortcomings illustrated in (Figure 9): in the case of closed invalid regions, data must be created from scratch, or it may result in data being extrapolated quite long distances (e.g., in narrow regions).

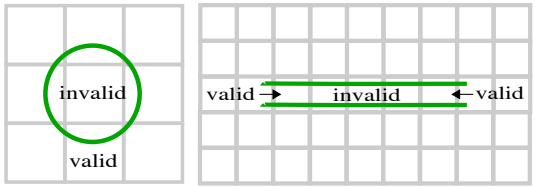


Figure 9: *Failure cases for standard velocity extrapolation from valid into invalid (uninitialized) cells. Left: Closed regions cannot be extrapolated into. Right: Long narrow regions may require extrapolation across arbitrary distances.*

We observe that with an appropriate PIC/FLIP implementation, this issue does not arise. Given a sufficient sampling of Lagrangian particles, there is no need for backtracing or extrapolation; velocity data carried by the particles travels *forwards* in tandem with the solid boundary to fill the freshly cleared cells, much like in the physical world. We maintain a good sampling of particles throughout by placing a user-defined lower and upper bound on the number of particles *per unit area*, and reseeding on each *sub-cell* independently. To complete our advection scheme, we just need mechanisms to transfer particle velocity data to and from the cut-cell mesh.

Transfer to Mesh We transfer velocity information carried by the particles onto the simulation mesh nodes by setting each nodal velocity to be a weighted average of the velocity of all particles in the cells incident on that node. We use SPH kernels throughout, with one minor twist in presence of solid geometry. We use a raycast to check whether the node in question is visible from the particle, and if not, we discard its contribution to that node. When face-normal velocity components are needed by the pressure solve, they are computed by interpolating the velocity vector from the cell nodes and taking the dot-product with the corresponding face normal. We use an SPH kernel radius of twice the grid cell width, although one can safely use smaller kernels provided they cover one full cell.

Transfer to Particles Interpolation of data from the simulation mesh back to the particles is performed using our enhanced interpolation scheme. This allows for a standard PIC or FLIP update to be performed. We found that using FLIP in cut cells leads to instability, particularly for small cells; we conjecture that the inherent instabilities in the FLIP scheme are exacerbated in the presence of such small cells. Therefore, *we revert to a pure PIC approach* in cut cells, using FLIP only in the broader domain.

As we discuss later in Section 8, truly ideal boundary-respecting trajectories are infeasible with discrete time-integrators due to numerical error; however, for the remaining particle trajectories that do cross the boundary, it is necessary to rely on direct collision detection and response as in previous work [Guendelman et al. 2005]. Nevertheless, because of our conforming velocity fields, the majority of particles do not cross over or collide.

7 Results

We refer the reader to our supplemental video for various results in both 2-D and 3-D, which we summarize below. Timings and settings are given in Table 2; all our results are computed using a single core of an Intel i7-2600 CPU at 3.4 GHz with 8GB memory. The robust intersection processing needed for mesh generation is handled with the CGAL library [CGAL 2016]. We simulated a single timestep per frame.

While we focus primarily on coarse grids, Table 3 shows how our technique scales with increasing grid resolution. It provides some cut-cell statistics and performance numbers for a fluid flow around a Bunny model (5,002 triangles) with grid resolutions ranging from 8^3 up to 256^3. Timings were obtained by averaging five measurements with identical settings. As expected, as the number of regular cubic cells increases by a factor of 2^3 from one grid resolution to the next, the number of cut cells only increases by a factor of 2^2. Consequently, the *percentage* of cut cells (relative to the number of grid cells) reduces by a factor of 2; that is, for any given solid geometry, the *relative* overhead associated with handling irregular cells decreases as grid resolution increases. The cut-cell generation time is dominated by CGAL operations (ranging from 80% to 90% of the time for resolutions up to 128^3). The identification of all faces (both mesh and grid ones) incident to each mixed node is currently performed in a brute force and unoptimized fashion, and thus its cost (column *Mixed Nodes* in Table 3) increases by a factor of roughly 15 from one resolution to the next; this rapid growth ultimately causes it to reduce CGAL's relative contribution to the overall time at the 256^3 grid level. We expect that optimizing this procedure will significantly accelerate cut-cell generation at high resolutions. The rightmost columns of Table 3 show the advection and projection times involved in simulating a single time step.

Our simulations do not apply vorticity confinement or other artificial turbulence-creation mechanisms, although these could be easily added. Our 3-D examples use 64 or 128 PIC/FLIP particles per cell, while our 2-D examples use 32 or 64 particles; these counts are higher than normal because we want to ensure that even small or skinny cells are sufficiently well-sampled. A smart adaptive particle sampling scheme would likely bring these values down with minimal impact on the results.

Example	Advection Time	Projection Time	Total Time	Meshing	Grid Dims	No. Cut-Cells
Linked Tori - Free Slip	0.372	0.040	0.597	0.443 (once)	$7 \times 7 \times 7$	88
Bunny - No Slip	0.343	0.004	0.516	1.527 (once)	$7 \times 7 \times 7$	78
Bunny - Free Slip	0.490	0.004	0.881	1.469 (once)	$7 \times 7 \times 7$	78
Dragon (fine) - Free Slip	1.124	0.301	2.686	7.984 (once)	$19 \times 14 \times 11$	1298
Dragon (coarse) - Free Slip	0.880	0.172	1.986	4.446 (once)	$11 \times 9 \times 6$	326
Disk - Free Slip	0.319	0.118	0.63	0.197 (per frame)	$25 \times 21 \times 13$	137
Disk - No Slip	0.296	0.105	0.554	0.200 (per frame)	$25 \times 21 \times 13$	137
Rotating Paddle - No Slip	0.290	0.148	0.597	0.153 (per frame)	$23 \times 10 \times 6$	67

Table 2: *Timing and parameters for 3-D simulations. Timing information is in seconds per frame, and is computed as an average over the first 3-4 frames. Total time excludes meshing, listed separately. For moving geometry, the cut-cell count is given for the first frame. For static meshes, meshing occurs only once at the start. All examples use 64 particles per cell, except the Linked Tori with 128.*

Grid Resolution	# Cut-Cells	% Cut-Cells	# Polygons	CC Generat. Time (sec)	CGAL (sec)	CGAL (%)	Mixed Nodes (sec)	Advect (sec)	Project (sec)
$8 \times 8 \times 8$	90	17.58	7,810	0.82	0.68	82.06	0.0004	0.54	0.006
$16 \times 16 \times 16$	316	7.71	10,636	1.31	1.15	87.95	0.0012	0.71	0.045
$32 \times 32 \times 32$	1,359	4.15	17,502	2.81	2.55	90.43	0.0168	1.24	0.956
$64 \times 64 \times 64$	5,260	2.01	32,342	7.85	6.62	84.36	0.2030	2.57	10.094
$128 \times 128 \times 128$	20,943	1.00	70,034	29.20	23.37	80.06	3.0822	7.80	47.218
$256 \times 256 \times 256$	83,518	0.50	176,584	172.33	111.12	64.48	49.0320	17.09	224.981

Table 3: *Statistics for a fluid simulation around the Bunny model (5,002 triangles) on grids of various resolutions. For each grid resolution, the table provides the number of cut cells (# Cut-Cells), its* percentage *with respect to the total number of cubic cells in the regular grid (% Cut-Cells), the number of triangles in the Bunny mesh after intersecting with the grid (# Polygons), the total time in seconds required to generate the cut cells (CC Generat. Time), the subset of the cut-cell generation time spent on CGAL operations (CGAL (sec)), the* percentage *of the cut-cell generation time corresponding to CGAL operations (CGAL (%)), the time spent by our meshing algorithm in the key step of finding all faces incident on each mixed node (Mixed Nodes), and the advection and projection times for simulating one time step.*

7.1 Flow through narrow regions

Branching Tube Figure 10 shows a challenging example of flow in a thin gap: a narrow tube, whose width is less than that of a grid cell, turns and branches, creating numerous cut cells and a rich topology. This illustrates the ability of our technique to handle flows through complex regions defined by closely spaced thin boundaries. Results are shown for interpolation using both free-slip (top) and no-slip (bottom) rules. Here and elsewhere free-slip is generally preferred for its superior behavior, but we show various examples with no-slip for completeness.

Ghost in Maze A similar but more elaborate example in 2-D features a complex maze structure, with a "ghost" character traveling through. Again the fluid follows the solid geometry well on the coarse grid.

Dragon and Linked Tori The teaser (Figure 1, middle and right) shows two 3-D examples of flow through quite narrow regions discretized with few cells: a passage through the dragon mesh (grid dimensions: $11 \times 9 \times 6$), and two linked tori (grid dimensions: $7 \times 7 \times 7$). The forcing is provided by setting a velocity boundary condition on a few faces at one end of the thin region. We ran a second version of the dragon at approximately twice the grid resolution in each dimension; while the flow is indeed smoother at finer resolution, it flows in essentially the same fashion as its less well-resolved counterpart. It is this tradeoff of quality and cost, independent of the geometry, that we seek to provide the user.

7.2 Flow around irregular objects

Three Circles We demonstrate that our method is also effective for standard solid *volumetric* objects in close proximity by simulating flow through thin gaps between a set of three circles in 2-D.

Bunny Figure 11 illustrates the improved quality of flow around coarse 3-D objects (5,002 triangles) as well. Conforming polyhedral interpolation ensures a reasonable motion that follows the curves of the bunny even on a $7 \times 7 \times 7$ grid. Trading computational cost for quality, Figure 12 shows how the level of turbulent detail increases with higher resolution grids.

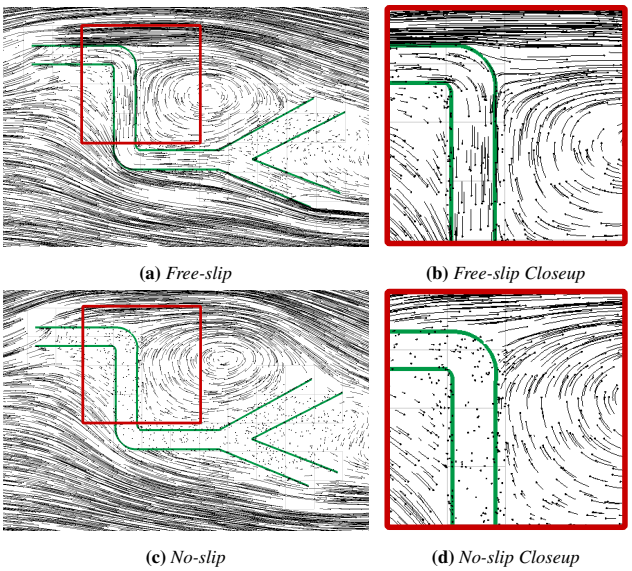
7.3 Flow around thin objects

Oscillating Lines A sequence of examples in our video feature two vertical line segments oscillating back and forth horizontally. At the closest point of their trajectories, the segments occupy the same column of grid cells, dividing them into three sub-cells; even in this extreme case the flow behaves naturally. We can see all our contributions in action: the cut-cell pressure projection yields proper fluxes, our PIC/FLIP particles ensure coherent motion without velocity extrapolation for swept-over regions, and our modified interpolation yields particle motion that conforms closely to segments. A close-up illustrates that with free-slip interpolation, the fluid flows vertically even as it is squeezed out of the slender sub-cell narrow gap at the closest position. With no-slip, the interpolated vertical velocities in the gap drop to zero when the segments enter the same grid column, since they are forced to match that the solid.

Diagonal Line We further illustrate the accuracy of the ideal free-slip conditions with a 2-D example of a diagonal solid line embedded in a perfectly tangential flow, showing that it does not disrupt the flow unless it rotates.

Stirring Line To test free-slip in a related scenario where the object is moving and the flow is stationary, our video includes a 2-D example with a diagonal solid line translating tangentially without disrupting the flow; later it begins to rotate and stir the surrounding fluid. By contrast, no-slip conditions immediately induce flow.

Disk Slicing Smoke Extending the above scenario to 3-D, we reproduce a test proposed by Robinson-Mosher et al. [2009] in which a disk with infinitesimal thickness slices tangentially through a block of stationary smoke (Figure 13). With ideal free-slip, the



(a) *Free-slip* **(b)** *Free-slip Closeup*

(c) *No-slip* **(d)** *No-slip Closeup*

Figure 10: *Flow simulation on a turning and branching tube whose width is smaller than a grid cell width. (a) Free-slip flows smoothly. (c) No-slip halts in the tube. (b) and (d) show closeup views of the highlighted regions.*

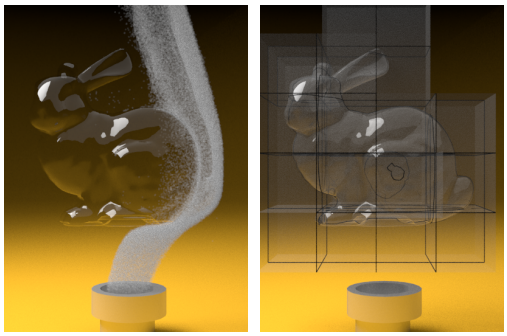


Figure 11: *Left: The fluid flow conforms to the irregular bunny mesh due to our use of conforming polyhedral interpolation. Right: The same bunny with black curves illustrating the coarse grid.*

smoke should remain perfectly stationary as the disk slips through edge-on; when it passes through a second time while rotating, the smoke should be disturbed. The accuracy of the cut-cell pressure solver allows our simulator to pass this stress test, in contrast to the results in previous work.

Rotating Line and Paddle Our 2-D example of a rotating line with no-slip conditions shows that the fluid is able to faithfully react to and follow the moving geometry. In the 3-D variant of this example shown in the teaser (Figure 1, left), a rotating thin paddle translates back and forth through a fluid domain stirring up a volume of smoke on a grid with dimensions $23 \times 10 \times 6$. This example is modeled after one proposed by Klingner et al. [2006] and used by Batty et al. [2007], with the exception that our paddle is extremely thin relative to the grid resolution. This compares favourably to the work of Batty et al. who used as their lowest resolution a grid of $40 \times 20 \times 30$ in order to ensure that their much thicker paddle was adequately resolved at the grid scale. In this example, to avoid issues caused by inadequate treatment of dangling interior faces provided by Shepard interpolation, we assigned the paddle a finite but very small thickness which ensures that SBC is used.

Lines	# CC	% CC	CC Gen.	Adv.	Proj.	Total
1	8	5.00	0.001	0.0016	0.023	0.0256
8	29	18.12	0.003	0.0028	0.035	0.0408
64	197	123.12	0.038	0.0120	0.063	0.1130

Table 4: *For one step at a fixed 2D grid resolution* (16×10)*, 1, 8, and 64 lines per cell crossing over* 5 *cells (as in Figure 8 top). # of Cut-Cells (CC), percentage of CC relative to the regular 160 grid cells. Times (sec) for: CC generation, advection, projection, and total time.*

8 Discussion and Conclusion

We have presented a method to improve the handling of moving irregular solid boundaries in regular grid-based fluid animation, particularly when objects are thin or in close proximity on coarse grids. While the additional processing is non-trivial, it need only be done immediately around objects. Our approach should be extensible to free surface flow and two-way coupling, and it would be natural to incorporate sub-grid turbulence to eke out even greater apparent detail on coarse grids.

Our pressure discretization ignores dangling interior solid faces arising from *partially* cut cells, as in previous regular grid schemes for thin boundaries (e.g., [Day et al. 1998; Guendelman et al. 2005; Robinson-Mosher et al. 2009]). Precisely accounting for this geometry would require generating a fully unstructured *conforming* mesh within the cell. While coupling a regular grid MAC scheme to a full FEM scheme is possible (e.g., [Zheng et al. 2015]), it would sacrifice the numerical and implementation benefits of our chosen *nearly-regular* grid discretization.

When tunnels between solid geometry within a single cell become *very* small or labyrinthine, a sufficient number of particles may fail to flow into or through gaps. Hence, truly extreme scenarios, such as flow through stacked pages of a book or pores of a sponge, remain impractical; porous flow or homogenization schemes may be preferable.

Although our interpolation method improves the interpolated velocities and resulting trajectories, it cannot *guarantee* trajectories never cross, due to truncation error in time integration. For free-slip conditions on cells with sharp corners, the interpolated velocity may also still have trajectories that do not satisfy $\mathbf{u}_{fluid} \cdot \mathbf{n} = \mathbf{u}_{solid} \cdot \mathbf{n}$ at all points along the perimeter of the cell. More broadly, our interpolants cannot ensure *pointwise* divergence-free velocity fields, which can lead to uneven particle distributions or poor flows near high-frequency geometry. However, for open regular grid regions, even standard trilinear and tricubic interpolation do not yield pointwise divergence-free fields. This highlights an interesting challenge for future work: can one construct interpolants which simultaneously (a) respect the discrete face velocities from the pressure solve, (b) accurately conform to boundaries, and (c) satisfy pointwise incompressibility? This holds out the potential to produce significantly improved visual results even in extremely under-resolved regions.

Acknowledgments

This work was supported by grants from NSERC (RGPIN-04360-2014) and CNPq (306196/2014-0, 482271/2012-4, 140811/2014-1, 201481/2014-6).

References

AFTOSMIS, M. J., BERGER, M. J., AND MELTON, J. E. 1998. Robust and efficient Cartesian mesh generation for component-based geometry. *AIAA Journal 36*, 6, 952–960.

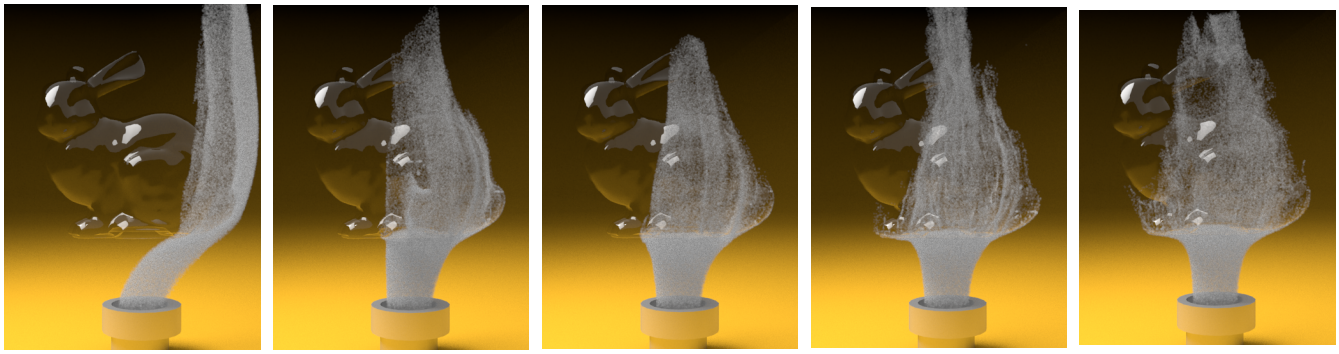


Figure 12: *Simulation of fluid flow around the Bunny model (free-slip, same time step) using grids of various resolutions. While the level of turbulent detail naturally increases at higher resolutions, the flow still respects the geometry even at extremely coarse resolutions.*

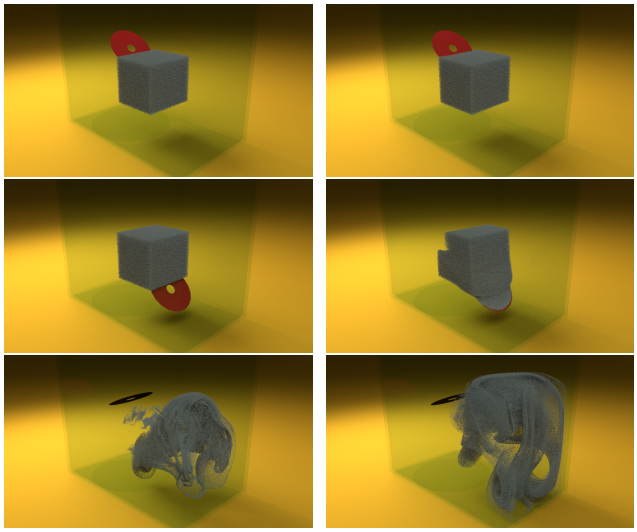


Figure 13: *A disk passing through smoke, first tangentially (2nd row), then while rotating (3rd row). Left column: Free-slip case. The smoke is undisturbed after the first tangential slice through. Right column: No-slip case. The smoke is disturbed immediately.*

BATTY, C., BERTAILS, F., AND BRIDSON, R. 2007. A fast variational framework for accurate solid-fluid coupling. *ACM Trans. Graph. 26*, 3, 100.

BATTY, C., XENOS, S., AND HOUSTON, B. 2010. Tetrahedral embedded boundary methods for accurate and flexible adaptive fluids. *Computer Graphics Forum (Eurographics) 29*, 2, 695–704.

BOJSEN-HANSEN, M., AND WOJTAN, C. 2013. Liquid surface tracking with error compensation. *ACM Trans. Graph. 32*, 4, 79:1–79:10.

BROCHU, T., BATTY, C., AND BRIDSON, R. 2010. Matching fluid simulation elements to surface geometry and topology. *ACM Trans. Graph. 29*, 4, 47.

CGAL. 2016. *CGAL User and Reference Manual*, 4.8 ed. CGAL Editorial Board.

CHENTANEZ, N., GOKTEKIN, T. G., FELDMAN, B. E., AND O'BRIEN, J. F. 2006. Simultaneous coupling of fluids and deformable bodies. In *Symp. on Computer Animation*, 83–89.

CHENTANEZ, N., FELDMAN, B. E., LABELLE, F., O'BRIEN, J. F., AND SHEWCHUK, J. R. 2007. Liquid simulation on lattice-based tetrahedral meshes. In *Symp. on Computer Animation*, 219–228.

CLAUSEN, P., WICKE, M., SHEWCHUK, J. R., AND O'BRIEN, J. F. 2013. Simulating liquids and solid-liquid interactions with Lagrangian meshes. *ACM Trans. Graph. 32*, 2, 17.

CROCKETT, R. K., COLELLA, P., AND GRAVES, D. T. 2011. A Cartesian grid embedded boundary method for solving the Poisson and heat equations with discontinuous coefficients in three dimensions. *J. Comp. Phys. 230*, 7, 2451–2469.

DAY, M., COLELLA, P., LIJEWSKI, M., RENDLEMAN, C., AND MARCUS, D. 1998. Embedded boundary algorithms for solving the Poisson equation on Complex Domains. Tech. rep., LBNL.

DICK, C., ROGOWSKY, M., AND WESTERMANN, R. 2016. Solving the fluid pressure Poisson equation using multigrid - evaluation and improvements. *IEEE TVCG*.

EDWARDS, E., AND BRIDSON, R. 2014. Detailed water with coarse grids: Combining surface meshes and adaptive discontinuous Galerkin. *ACM Trans. Graph. 33*, 4, 136:1–136:9.

ELCOTT, S., TONG, Y., KANSO, E., SCHRÖDER, P., AND DESBRUN, M. 2007. Stable, circulation-preserving, simplicial fluids. *ACM Trans. Graph. 26*, 1, 4.

ENRIGHT, D., NGUYEN, D., GIBOU, F., AND FEDKIW, R. 2003. Using the particle level set method and a second order accurate pressure boundary condition for free surface flows. In *Proceedings of the 4th ASME-JSME Joint Fluids Engineering Conference*, ASME, 337–342.

FEDKIW, R., STAM, J., AND JENSEN, H. W. 2001. Visual simulation of smoke. In *SIGGRAPH*, 15–22.

FELDMAN, B. E., O'BRIEN, J. F., AND KLINGNER, B. M. 2005. Animating gases with hybrid meshes. *ACM Trans. Graph. 24*, 3 (jul), 904–909.

FERSTL, F., WESTERMANN, R., AND DICK, C. 2014. Large-scale liquid simulation on adaptive octree grids. *IEEE TVCG Preprint*.

GUENDELMAN, E., SELLE, A., LOSASSO, F., AND FEDKIW, R. 2005. Coupling water and smoke to thin deformable and rigid shells. *ACM Trans. Graph. 24*, 3, 973–981.

HELLRUNG, J., WANG, L., SIFAKIS, E., AND TERAN, J. 2012. A second order virtual node method for elliptic problems with interfaces and irregular domains. *J. Comp. Phys. 231*, 4, 2015–2048.

HOUSTON, B., BOND, C., AND WIEBE, M. 2003. A unified approach for modeling complex occlusions in fluid simulations. In *SIGGRAPH Sketches*.

JOHANSEN, H., AND COLELLA, P. 1998. A Cartesian grid embedded boundary method for Poisson's equation on irregular domains. *J. Comp. Phys. 147*, 1 (nov), 60–85.

JU, T., SCHAEFER, S., AND WARREN, J. 2005. Mean value coordinates for closed triangular meshes. *ACM Trans. Graph. 24*, 3, 561–566.

KIM, D., SONG, O.-Y., AND KO, H.-S. 2009. Stretching and wiggling liquids. *ACM Trans. Graph. 28*, 5, 120.

KLINGNER, B. M., FELDMAN, B. E., CHENTANEZ, N., AND O'BRIEN, J. F. 2006. Fluid animation with dynamic meshes. *ACM Trans. Graph. 25*, 3, 820–825.

LABELLE, F., AND SHEWCHUK, J. R. 2007. Isosurface stuffing: Fast tetrahedral meshes with good dihedral angles. *ACM Trans. Graph. 26*, 3, 57.

LANGER, T., BELYAEV, A., AND SEIDEL, H.-P. 2006. Spherical barycentric coordinates. In *Eurographics Symposium on Geometry Processing*, 81—-88.

LENAERTS, T., AND DUTRÉ, P. 2008. Unified SPH model for fluid-shell simulations. Tech. rep., Katholieke Universiteit Leuven.

LOSASSO, F., GIBOU, F., AND FEDKIW, R. 2004. Simulating water and smoke with an octree data structure. *ACM Trans. Graph. 23*, 3 (aug), 457–462.

LOSASSO, F., FEDKIW, R., AND OSHER, S. 2005. Spatially adaptive techniques for level set methods and incompressible flow. *Computers & Fluids 35*, 10, 995–1010.

MISZTAL, M., BRIDSON, R., ERLEBEN, K., BAERENTZEN, A., AND ANTON, F. 2010. Optimization-based fluid simulation on unstructured meshes. In *VRIPHYS*.

MITTAL, R., AND IACCARINO, G. 2005. Immersed boundary methods. *Annual review of fluid mechanics 37*, 239–261.

MOLINO, N., BAO, Z., AND FEDKIW, R. 2004. A virtual node algorithm for changing mesh topology during simulation. *ACM Trans. Graph. 23*, 3 (aug), 385–392.

NESME, M., KRY, P. G., JEVRÁBKOVÁ, L., AND FAURE, F. 2009. Preserving topology and elasticity for embedded deformable models. *ACM Trans. Graph. 28*, 3, 52.

NG, Y. T., MIN, C., AND GIBOU, F. 2009. An efficient fluid-solid coupling algorithm for single-phase flows. *J. Comp. Phys. 228*, 23, 8807–8829.

OZGEN, O., KALLMANN, M., RAMIREZ, L., AND COIMBRA, C. 2010. Underwater cloth simulation with fractional derivatives. *ACM Trans. Graph. 29*, 3, 23.

PURVIS, J. W., AND BURKHALTER, J. E. 1979. Prediction of critical Mach number for store configurations. *AIAA Journal 17*, 11, 1170–1177.

QIU, L., YU, Y., AND FEDKIW, R. 2015. On thin gaps between rigid bodies two-way coupled to incompressible flow.

RASMUSSEN, N., ENRIGHT, D., NGUYEN, D., MARINO, S., SUMNER, N., GEIGER, W., HOON, S., AND FEDKIW, R. 2004. Directable photorealistic liquids. In *Symposium on Computer Animation*, 193–202.

ROBINSON-MOSHER, A., SHINAR, T., GRETARSSON, J., SU, J., AND FEDKIW, R. 2008. Two-way coupling of fluids to rigid and deformable solids and shells. *ACM Trans. Graph. 27*, 3, 46.

ROBINSON-MOSHER, A., ENGLISH, R. E., AND FEDKIW, R. 2009. Accurate tangential velocities for solid fluid coupling. In *Symposium on Computer Animation*, 227–236.

ROBLE, D., BIN ZAFAR, N., AND FALT, H. 2005. Cartesian grid fluid simulation with irregular boundary voxels. In *SIGGRAPH Sketches*, 138.

ROSATTI, G., CESARI, D., AND BONAVENTURA, L. 2005. Semi-implicit, semi-Lagrangian modelling for environmental problems on staggered Cartesian grids with cut cells. *J. Comp. Phys. 204*, 1 (mar), 353–377.

SCHWARTZ, P., BARAD, M., COLELLA, P., AND LIGOCKI, T. 2006. A Cartesian grid embedded boundary method for the heat equation and Poisson's equation in three dimensions. *J. Comp. Phys. 211*, 2, 531–550.

SELLE, A., FEDKIW, R., KIM, B., LIU, Y., AND ROSSIGNAC, J. 2008. An unconditionally stable MacCormack method. *SIAM J. Sci.Comput. 35*, 2-3, 350–371.

SHEPARD, D. 1968. A two-dimensional interpolation function for irregularly-spaced data. In *ACM '68 Proceedings of the 1968 23rd ACM national conference*, 517–524.

SIFAKIS, E., DER, K., AND FEDKIW, R. 2007. Arbitrary cutting of deformable tetrahedralized objects. In *Symposium on Computer Animation*, 73–80.

SIN, F., BARGTEIL, A. W., AND HODGINS, J. K. 2009. A point-based method for animating incompressible flow. In *Symposium on Computer Animation*, ACM Press, 247–255.

STAM, J. 1999. Stable fluids. In *SIGGRAPH*, 121–128.

TERAN, J., SIFAKIS, E., BLEMKER, S. S., NG-THOW-HING, V., LAU, C., AND FEDKIW, R. 2005. Creating and simulating skeletal muscle from the visible human data set. *IEEE TVCG 11*, 3, 317–328.

WANG, K., GRETARSSON, J., MAIN, A., AND FARHAT, C. 2012. Computational algorithms for tracking dynamic fluidstructure interfaces in embedded boundary methods. *International Journal for Numerical Methods in Fluids 70*, 4, 515–535.

WANG, Y., JIANG, C., SCHROEDER, C., AND TERAN, J. 2014. An adaptive virtual node algorithm with robust mesh cutting. In *Symposium on Computer Animation*, 77–85.

WEBER, D., MUELLER-ROEMER, J., STORK, A., AND FELLNER, D. 2015. A cut-cell geometric multigrid Poisson solver for fluid simulation. *Computer Graphics Forum 34*, 2, 481–491.

WOJTAN, C., THUEREY, N., GROSS, M., AND TURK, G. 2010. Physically-inspired topology changes for thin fluid features. *ACM Trans. Graph. 29*, 3, 50.

ZHENG, W., ZHU, B., KIM, B., AND FEDKIW, R. 2015. A new incompressibility discretization for a hybrid particle MAC grid representation with surface tension. *J. Comp. Phys. 280*, 1, 96–142.

ZHU, Y., AND BRIDSON, R. 2005. Animating sand as a fluid. *ACM Trans. Graph. 24*, 3 (jul), 965–972.