Power Diagrams and Sparse Paged Grids for High Resolution Adaptive Liquids

MRIDUL AANJANEYA[†*], Rutgers University
MING GAO[†] and HAIXIANG LIU, University of Wisconsin - Madison
CHRISTOPHER BATTY, University of Waterloo
EFTYCHIOS SIFAKIS, University of Wisconsin - Madison

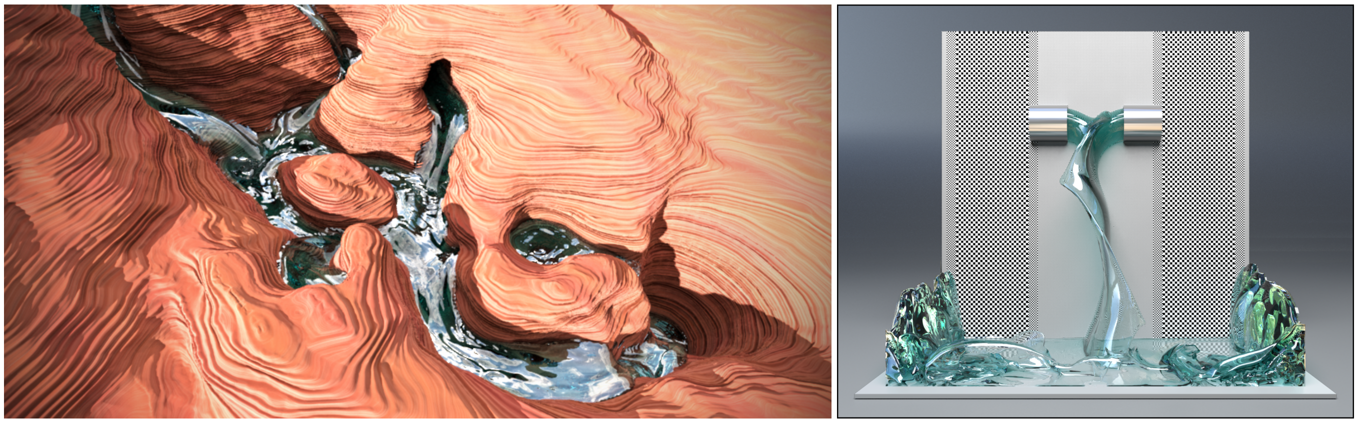


Fig. 1. (Left) Water filling a river bed surrounded by a canyon, with effective resolution $512^2 \times 1024$. Three refinement levels are used, based on proximity to the terrain. (Right) Sources inject water into a container and collide to form a thin sheet, with effective resolution 512^3. Adaptivity pattern shown on background.

We present an efficient and scalable octree-inspired fluid simulation framework with the flexibility to leverage adaptivity in any part of the computational domain, even when resolution transitions reach the free surface. Our methodology ensures symmetry, definiteness and second order accuracy of the discrete Poisson operator, and eliminates numerical and visual artifacts of prior octree schemes. This is achieved by adapting the operators acting on the octree's simulation variables to reflect the structure and connectivity of a *power diagram*, which recovers primal-dual mesh orthogonality and eliminates problematic T-junction configurations. We show how such operators can be efficiently implemented using a pyramid of sparsely populated uniform grids, enhancing the regularity of operations and facilitating parallelization. A novel scheme is proposed for encoding the topology of the power diagram in the neighborhood of each octree cell, allowing us to locally reconstruct it on the fly via a lookup table, rather than resorting to costly explicit meshing. The pressure Poisson equation is solved via a highly efficient, matrix-free multigrid preconditioner for Conjugate Gradient, adapted to the power diagram discretization. We use another sparsely populated uniform grid for high resolution interface tracking with a narrow band level set representation. Using the recently introduced SPGrid data structure, sparse uniform grids in both the power diagram discretization and our narrow band level set can be compactly stored and efficiently updated via streaming operations. Additionally, we present enhancements to adaptive level set advection, velocity extrapolation, and the fast marching method for redistancing. Our overall framework gracefully accommodates the task of dynamically adapting the octree topology during simulation. We demonstrate end-to-end simulations of complex adaptive flows in irregularly shaped domains, with tens of millions of degrees of freedom.

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

Additional Key Words and Phrases: Power diagrams, Octrees, Adaptivity

ACM Reference format:
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. Graph.* 36, 4, Article 140 (July 2017), 12 pages.
DOI: http://dx.doi.org/10.1145/3072959.3073625

1 INTRODUCTION

Liquids exhibit complex and detailed motion across a vast range of scales, from tiny ripples to huge waves; this fact motivates the desire for liquid simulation tools that can handle ever increasing levels of resolution. While a key avenue towards this goal is the development of more efficient numerical methods on regular uniform grids that conserve mass with large time steps [Chentanez and Müller 2012; Lentine et al. 2011, 2012] and allow for fast pressure projection [Ando et al. 2015; Dick et al. 2016; Lentine et al. 2010; Liu

[†] M. Aanjaneya and M. Gao are joint first authors.
[*] M. Aanjaneya was with the University of Wisconsin - Madison during this work.
This work was supported in part by National Science Foundation grants IIS-1253598, CCF-1423064, CCF-1533885 and by the Natural Sciences and Engineering Research Council of Canada under grant RGPIN-04360-2014. The authors are grateful to Nathan Mitchell for his indispensable help with modeling and rendering of examples. C. Batty would like to thank Ted Ying for carrying out preliminary explorations on quadtrees.

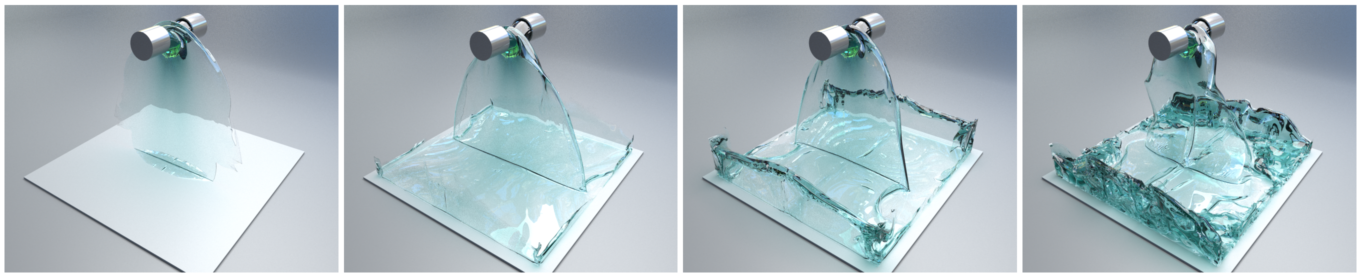


Fig. 2. Two sources inject water into a container forming a thin sheet, with effective resolution 512^3. The adaptivity pattern is shown in Figure 1; a narrow band level set at resolution 2048^3 is used for interface tracking. As instabilities develop in the flow field over time, the sheet starts to wobble, creating ripples.

et al. 2016; McAdams et al. 2010; Molemaker et al. 2008; Zhang and Bridson 2014], further computational gains can be obtained through *spatial adaptivity* — dedicating higher resolution to regions of high importance. Amongst the most natural approaches for augmenting standard staggered grid discretizations with spatial adaptivity is to replace the underlying grid structure with that of an octree. The critical challenge that arises for octrees is the treatment of T-junction (or hanging node) configurations where small cells are incident on larger cells. This is not solely a matter of bookkeeping in data structures: the discretization of the physics at these points must be done with care, or disturbing visual and numerical artifacts can emerge.

Losasso et al. [2004] proposed the first method to simulate liquids on octrees. They sacrificed accuracy in the discretization of pressure gradients near T-junctions in exchange for a simpler symmetric and positive definite Laplacian matrix that can be solved by Conjugate Gradient. The price for this choice was the loss of orthogonality between the stored velocity components at the T-junction cell faces and the corresponding discrete pressure gradients across those faces. This approximation reduces the accuracy of the pressure to first order, and gives rise to non-physical parasitic currents at T-junctions even for hydrostatic scenarios, as discussed by various authors [Ando et al. 2013; Batty et al. 2010; Chentanez et al. 2007; Ferstl et al. 2014; Losasso et al. 2006]. Losasso et al. [2006] subsequently proposed a modification that raises the accuracy of the pressure field to second order, but allows for adaptivity *only in regions completely interior to the fluid.* It is not clear how to incorporate solid or free surface boundaries in the presence of adaptivity, which necessitates uniformly refining all regions near surfaces and boundaries.

In this paper, we describe a new staggered octree discretization that retains the positive definiteness and advantageous scaling of Losasso's scheme, while eliminating inconsistent pressure gradients and enabling accurate boundary condition enforcement, even across regions of changing resolution. Inspired by fluid animation approaches relying on Voronoi and power diagrams [Brochu et al. 2010; de Goes et al. 2015; Sin et al. 2009], we modify the implied geometry of the octree near T-junctions to have the form of a power diagram; this effectively eliminates the problematic T-junctions and recovers orthogonality between pressure gradients and cell faces. Embedded free surface and solid boundary conditions can then be directly incorporated on this hybrid mesh, per Brochu et al. [2010]. The resulting octree Poisson solver is second order accurate in pressure and artifact-free. In spite of this different geometric perspective used in our discretization, we retain (with minor caveats) the ability to store our simulation variables on a traditional octree rather than an explicit unstructured mesh, and exploit the regularity of this representation in the interest of performance.

We complete our octree simulator to support end-to-end liquid simulation of complex large-scale scenarios via several complementary enhancements. We leverage the *SPGrid* data structure [Setaluri et al. 2014] to allow our solver to process octree data at performance levels competitive with regular uniform grids containing a similar number of variables. Notably, this standard of efficiency is notoriously difficult to match with pointer-based octree implementations or unstructured mesh discretizations, due to the impact that indirect and irregular memory access (or the overhead of explicit storage of topology) has on the optimal utilization of the available bandwidth. We accurately track the surface using a uniformly high resolution narrow band level set, exploiting the compact footprint of the SPGrid structure and an adaptively multi-stepped semi-Lagrangian advection scheme. To support accurate fast-marching-based level set redistancing on octrees, we perform a localized Delaunay triangulation of the cell centers incident on T-junctions, and adapt standard fast marching strategies to the resulting hybrid tetrahedral/hexahedral mesh. We solve the sparse, symmetric definite linear systems that arise from our Poisson discretization using a lightweight multigrid preconditioner for the Conjugate Gradient method. We efficiently store the multigrid hierarchy using SPGrid, while showing how a first order accurate multigrid V-cycle can be used as a building block of a very effective preconditioner for the second order problem.

Contributions. We developed an accurate and efficient staggered octree-based fluid simulation framework, featuring support for:

- second order accurate pressure projection allowing adaptivity even along solid and air boundaries;
- excellent affinity to parallelism-optimized data structures for octree storage (SPGrid), allowing us to easily scale to tens of millions of degrees of freedom on a single computer;
- a simple and effective narrow band level set interface tracking scheme that uses SPGrid to reduce memory footprint;
- an enhanced fast marching method for octrees;
- a tailored multigrid-preconditioner for Conjugate Gradient.

2 RELATED WORK

Our approach builds on standard grid-based fluid animation techniques; Bridson [2015] provides a useful overview of this family of methods. We adopt a staggered discretization, which naturally avoids instabilities from odd-even decoupling ("checkerboarding") in the pressure field. Our treatment of irregular free surface and solid boundaries draws on the ghost fluid method [Enright et al. 2003; Gibou et al. 2002] and the variational/cut-cell solid scheme [Batty et al. 2007; Ng et al. 2009], respectively. Incorporating these ideas into our solver ensures accurate pressure solutions and avoids voxelization artifacts. While we focus on adaptive approaches for Eulerian methods, Lagrangian SPH and vortex methods have also been applied in the context of adaptivity [Adams et al. 2007; Golas et al. 2012; Solenthaler and Gross 2011; Takahashi and Lin 2016].

Octrees. Popinet [2003] proposed the first octree-based fluid solver, extending earlier adaptive mesh refinement (AMR) schemes for the Poisson problem in the computational fluid dynamics community [Martin and Cartwright 1996; Minion 1996]. Losasso et al. [2004] extended this approach with support for free surface flow, and incorporated a simplification to yield symmetric positive definite systems for pressure projection. Applications of their method have included simulation of bubbles [Hong and Kim 2005], coupling with dynamic objects [Guendelman et al. 2005], and even lightning [Kim and Lin 2007]. Unfortunately, this approach reduces the pressure accuracy to first order and introduces visual artifacts in the velocity field. Losasso et al. [2006] later improved the discretization of the pressure projection to recover second order accuracy while preserving symmetry, although it is unclear how to incorporate irregular boundary conditions near T-junctions because the necessary modifications to the Poisson stencil are mutually incompatible; we overcome this limitation to allow refinement and boundaries to seamlessly co-exist without loss of accuracy. Ferstl et al. [2014] proposed a finite element method (FEM) that subdivides octree surface cells cut by surface geometry to yield a conforming mesh, applying Nitsche's method for the free surface and stabilization to minimize checkerboarding. Like Losasso's method, it requires boundaries to be uniformly resolved. Furthermore, solid boundaries were treated in a voxelized fashion. In contrast to the preceding methods, our approach also leverages the recent SPGrid data structure [Setaluri et al. 2014] to overcome the significant efficiency penalties inherent in typical pointer-based octree structures. Nielsen and Bridson [2016] recently discussed some details of the FEM-based adaptive tile-tree scheme used in Maya's Bifrost simulator, although full details are not available. Like our method, they aim to preserve a regular grid-like structure to maximize efficiency, but do so using a more rapidly branching octree involving $5 \times 5 \times 5$ blocks, rather than our pyramid of SPGrids. Their use of FEM was explicitly motivated by the fact that previous finite volume octree schemes have struggled to simultaneously provide matrix symmetry, second order accuracy, and support for non-axis-aligned boundaries; our method achieves all three.

Stretched Grid Cells. Another quite useful, albeit more restrictive, approach to spatial adaptivity is the use of stretched grid cells. For example, portions of a regularly sampled domain may be replaced with tall cells that have been stretched along a single axis [Chentanez and Müller 2011; Irving et al. 2006], or larger sections of a regular grid may be stretched along multiple axes by essentially translating entire grid lines [Zhu et al. 2013]. These approaches preserve most of the regular grid efficiency while offering some benefits of adaptivity.

Tetrahedral Meshes. An important family of alternatives to octrees are unstructured or partially structured adaptive tetrahedral meshes. An array of staggered Eulerian finite volume discretizations have been presented for such meshes [Batty et al. 2010; Chentanez et al. 2007, 2006; Elcott et al. 2007; Feldman et al. 2005; Klingner et al. 2006]. In particular, Batty et al. [2010] extended these methods with support for irregular free surfaces and solids to allow straightforward adaptivity even in the presence of boundaries, a feature we likewise pursue. While purely unstructured tetrahedral methods incur non-trivial computational costs due to frequent remeshing and data structure overhead, semi-structured lattice-based variants can somewhat reduce these costs by exploiting an underlying octree structure [Chentanez et al. 2007], although they still incur greater overhead than a pure octree. One clear advantage of tetrahedral meshes over basic octrees is that they support adaptivity without any T-junctions. A drawback is that pressures must be placed at tetrahedron circumcenters to avoid artifacts arising from the pressure solve, yet circumcenters are not guaranteed to be inside their corresponding tetrahedra [Batty et al. 2010]. Ando et al. [2013] partially circumvent this issue with a finite element variation that instead places velocity vectors at tetrahedron barycenters and pressures at vertices. However, the authors note that they must drop back to a first order discretization for poorly shaped tetrahedra which occur near resolution transitions in their lattice mesh; by contrast, we preserve second order accuracy across transitions and throughout the domain. Node-based Lagrangian schemes that use dynamically remeshed unstructured tetrahedral meshes can also support adaptivity [Clausen et al. 2013; Misztal and Bærentzen 2012; Misztal et al. 2010]. These methods rely on stabilization terms or extra remeshing procedures to avoid locking or odd-even pressure decoupling artifacts. Moreover, they are generally costlier than Eulerian approaches due to the overhead of continuous remeshing and unstructured mesh manipulation.

Voronoi Diagrams. Since the circumcentric dual of a Delaunay tetrahedralization is a Voronoi diagram, the Poisson equation can be readily discretized on this dual mesh too, using essentially the same staggered finite volume approaches as the tetrahedral schemes above. This has been exploited in several ways. Sin et al. [2009] constructed a particle-based Lagrangian scheme that performs pressure projection on the Voronoi diagram of the particles at each step. Brochu et al. [2010] combined a mesh-based surface tracking scheme with a carefully constructed Voronoi diagram to enable capturing of thin liquid features. English et al. [2013a; 2013b] constructed a nested (Chimera) grid scheme relying on the flexibility of the local Voronoi structure to stitch together the boundaries between regular grids of differing resolutions. Voronoi diagrams have similarly been exploited in computational physics: Guittet et al. improved the ghost-fluid method for two-phase Poisson problems by aligning Voronoi faces with the fluid interface [Guittet et al. 2015a], and separately used the Voronoi diagram of fluid face midpoints to treat viscosity on non-graded octrees [Guittet et al. 2015b].

Power Diagrams. Mullen et al. [2011] observed that regular triangulations and their associated dual power diagrams can offer

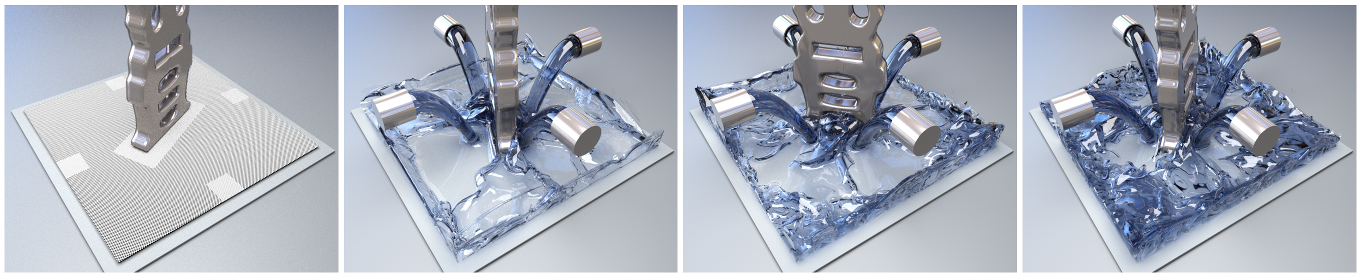


Fig. 3. Four sources pour water that collides with a kinematic rotating object in a container, with effective resolution 512^3. The adaptivity pattern, illustrated on the left, continuously rotates with the central spinning object. A narrow band level set with 2048^3 resolution is used to track the free surface.

increased flexibility in meshing, while preserving primal-dual orthogonality. This property is critical for constructing accurate discrete gradient and divergence operators [Batty et al. 2010] in the style of Discrete Exterior Calculus (e.g., [Elcott and Schröder 2006]). Building on this idea, de Goes et al. [2015] developed a Lagrangian particle method that constructs a power diagram at each time step for enforcing incompressibility. Krivá et al. [2016] used the power diagram of a quadtree in two dimensions to solve a Poisson problem in the context of image processing, although they did not identify their construction as a power diagram. Similarly, Sifounakis et al. [2016] proposed a so-called virtual slanting approach for a Navier-Stokes solver on quad/octrees; however, their strategy of simply slanting faces breaks down in three dimensions, where proper orthogonality cannot be maintained without (explicitly or implicitly) changing the local cell connectivity. By contrast, we observe that the desired construction is a power diagram, and use this fact to easily determine the correct grid geometry and topology.

Interface Tracking. We represent the liquid surface using a level set scheme [Sethian 1999], a common choice in fluid animation, and specifically a narrow band variant [Adalsteinsson and Sethian 1995] which we further customize to our application. The particle level set method of Enright et al. [2002] is a popular variant; it maintains Lagrangian particles around the Eulerian interface to accurately reconstruct the surface, building on ideas proposed by Foster and Fedkiw [2001]. Rather than exploit particles, however, we employ a purely Eulerian narrow band method at a resolution higher than the simulation mesh, similar to previous authors (e.g., [Bojsen-Hansen and Wojtan 2013; Goldade et al. 2016; Kim et al. 2009]). We differ in proposing a multi-stepping semi-Lagrangian strategy for reduced dissipation, and in leveraging the SPGrid data structure for a highly optimized implementation. Rather than increase resolution, Heo et al. [2010] obtained greater detail by proposing a pseudo-spectral level set method. The two most common alternatives to level sets are pure particle representations, as in PIC or SPH schemes [Desbrun and Gascuel 1996; Foster and Metaxas 1996; Jiang et al. 2015; Zhu and Bridson 2005], and Lagrangian triangle meshes with dynamic topology [Brochu and Bridson 2009; Müller 2009; Wojtan et al. 2009]. Various hybrids have also been proposed [Bargteil et al. 2006; Yu et al. 2012]. While each approach has strengths and weaknesses, level sets stand out for their simplicity, smoothness, and potential for efficiency. The affinity of narrow band level sets to the SPGrid data structure allows for a particularly compact surface representation.

2.1 SPGrid arrays and sparse uniform grid pyramids

Our algorithmic formulation draws upon the work of Setaluri et al. [2014] and exploits two of their key proposals. First, it was observed that in lieu of a traditional pointer-based representation, a Cartesian octree can be stored as a pyramid of *sparsely populated uniform grids*. Thus, any cell of the octree can be uniquely identified with a single cell of one of the uniform grids in the pyramid. It was shown that the global computation of the Laplace operator can be equivalently performed by alternating (a) perfectly uniform stencils on each individual grid, and (b) highly structured data transfers solely between adjacent grids in the pyramid. Section 4 describes how we exploit this capability for our power diagram discretization.

Second, a low-level data structure named *SPGrid* (or "Sparse Paged Grid") was proposed for compact storage and efficient stream processing of sparsely populated uniform grids. SPGrid leverages the Virtual Memory subsystem, in conjunction with a Morton ordering, to index into a vast address range, but only materialize into physical memory those parts of the array that are actually touched. Thus, an array that may occupy more than 1TB if densely allocated, can occupy as little as 1GB in physical memory if only a spatially coherent 0.1% of this array was actually populated with data. Recognizing that simulation applications commonly utilize a large number of scalar or vector fields with nearly (or exactly) identical index domains, SPGrid interleaves several of these distinct *"channels"* within a 4KB page. Thus, an SPGrid with 4 float-valued channels fits a geometric block of $8 \times 8 \times 4$ variables for each of those channels in a 4KB page, while 16 float-valued channels would yield a block size of $4 \times 4 \times 4$. The number of channels can easily vary to cater to different tasks; for example, we use only 4 channels for interface tracking, and 16 channels for solving the incompressible Euler equations.

3 METHOD OVERVIEW

Our simulation methodology necessitates certain interventions in various parts of a standard liquid simulator. A large fraction of our contributions are centered around **pressure projection**, which as many authors have asserted [Ando et al. 2015; Dick et al. 2016; Lentine et al. 2010; Liu et al. 2016; McAdams et al. 2010; Molemaker et al. 2008; Zhang and Bridson 2014] is typically the most expensive component of the simulation loop in high resolution domains. Section 4 discusses several aspects of our pressure solver, including the discretization of the Poisson equation using power diagrams, the efficient storage of the discrete equations and unknowns in an adaptive grid structure, and fast solution of the resulting system using a preconditioner for Conjugate Gradient adapted from a multigrid scheme. Section 5 presents our **interface tracking** approach that uses a highly resolved implicit geometry representation of the free surface, localized to a narrow band around the moving interface and stored in a sparse grid structure (SPGrid) [Liu et al. 2016; Setaluri et al. 2014]. The same section also discusses details on how this representation is advected forward in time, interpolation of scalar and vector quantities, and enhancements to **fast marching and velocity extrapolation** that are mandated by our power diagram discretization. Important implementation details and optimizations in our formulation are reviewed in Section 6.

4 PRESSURE PROJECTION ON MODIFIED OCTREES

To achieve an accurate staggered finite volume discretization of the Poisson equation, the faces of the primal mesh which store velocity components should be orthogonal to the edges of the dual mesh, which correspond to pressure gradients [Batty et al. 2010; Perot and Subramanian 2007]. For sufficiently regular meshes this gives second order accurate pressures. In this section, we describe our discretization for the Poisson equation that exploits both the regularity of octrees for aggressive parallelization and the inherent primal-dual orthogonality of power diagrams for accuracy. Our objective is to solve the incompressible Euler equations

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

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

with a splitting scheme following Stam [1999]. Here, $\mathbf{u} = (u, v, w)$ is the vector velocity field, $\mathbf{f}$ encapsulates external forces, p is the scalar pressure field, t is time, and ρ is the liquid density.

4.1 Power diagram discretization

Aurenhammer [1987] defines the *power* of a point x with respect to a sphere of radius r as $d^2 - r^2$, where d is the distance of the point from the center of the sphere. Given a set of spheres, their power diagram is then a partition of space where each point is associated to the sphere with minimum power. We construct an octree-based power diagram by conceptually placing a sphere at each octree cell center with a radius of $\Delta x/\sqrt{3}$ (or $\Delta x/\sqrt{2}$ in 2D), where Δx is the cell's side length; this yields the circumsphere for each cell. For uniform resolution regions of the tree, the resulting power diagram leaves the regular Cartesian grid pattern unchanged. More remarkably, for T-junctions in two dimensional graded quadtrees, this choice gives a power diagram with *exactly* the same cell connectivity as the original quadtree, but with the geometry adapted to recover primal-dual orthogonality (see Figure 4(a,b)). However, in three dimensions a minor wrinkle arises: a given cell remains face-connected to its original face neighbors, but may now also share new faces with its original edge neighbors (see Figure 4(d,e)). Nevertheless, as we describe below, this geometry is still amenable to efficient computation of the Poisson equation.

We store pressures p at octree cell centers and normal components of the velocity field $\mathbf{u}$ at power cell faces. For computing the pressure, we discretize the Poisson equation in finite volume fashion over the

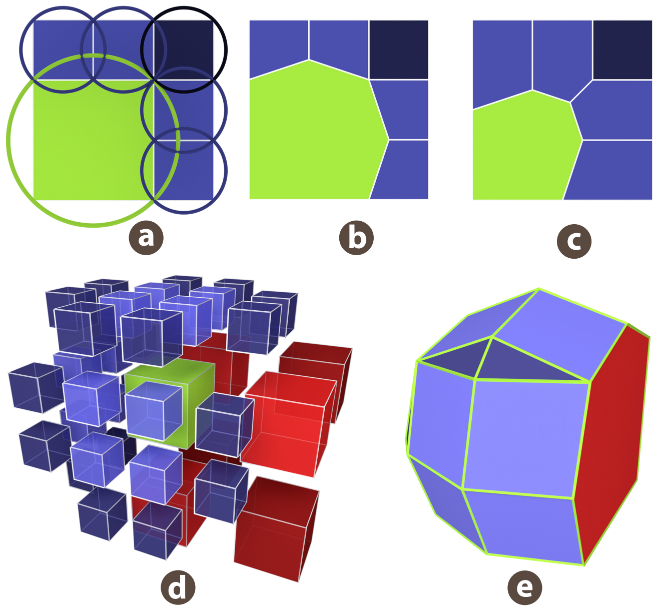


Fig. 4. (Top) A 2D cell bordering two T-junctions, along with its corresponding power diagram (middle), and the Voronoi diagram of the cell centers (right). The cells of the power diagram have the same neighbor relationships as the original quadtree, but the dual edges are orthogonal to cell faces. The Voronoi diagram of the same geometry creates new face connections between nearby cells at the same level. (Bottom) A 3D cell bordering three T-junctions, along with its corresponding power cell. Note the new faces that emerge between neighbors that previously only shared an edge. Colors imply resolution and neighbor relationship to the central coarse cell.

power cells as

$$\mathrm{V}_{\text{cell}}\nabla\cdot\frac{\nabla p}{\rho} = \sum_{\text{faces}} \mathrm{A}_{\text{face}}\left(\frac{\nabla p}{\rho}\cdot\mathbf{n}\right) \tag{3}$$

where $\mathbf{n}$ is the unit normal vector pointing out of a face, A_{face} is the face area, and V_{cell} is the volume. We similarly compute the discrete divergence as

$$\mathrm{V}_{\text{cell}}\nabla\cdot\mathbf{u} = \sum_{\text{faces}} \mathrm{A}_{\text{face}}(\mathbf{u}\cdot\mathbf{n}). \tag{4}$$

For treating boundary conditions, we adapt an earlier unstructured mesh embedded boundary approach [Batty et al. 2010; Brochu et al. 2010]. In our setting, this allows both the free surface and solid boundaries to arbitrarily cut through the domain, even near T-junctions. It requires only liquid signed distance values at cell centers and solid signed distance values at cell vertices. By contrast, previous methods [Losasso et al. 2006, 2004] required uniformly refined cells near boundaries. Our resulting linear system remains sparse, symmetric and positive definite, and ensures second order accuracy of the computed pressure field.

Voronoi diagram octrees. Another option to recover orthogonality would be the Voronoi diagram of the octree cell centers. For non-graded trees this can yield very general unstructured meshes that discard the regularity benefits of the tree structure (see e.g., [Guittet et al. 2015b]). While the variety of mesh configurations can naturally be reduced by grading, the Voronoi topology still differs more strongly from the original tree than for the power diagram, even in the simpler 2D setting (see Figure 4(c)). We therefore prefer the power diagram as it provides the necessary orthogonality while better preserving the original octree structure.

4.2 Pyramid of sparse uniform grids

By default, a power diagram would require an explicit mesh for storing the topological connectivity. However, such a representation would necessitate expensive lookups near T-junctions. Since power diagrams preserve the original octree structure, with the caveat of introducing some additional faces between edge neighbors, we adopt an approach similar to Setaluri et al. [2014] of organizing computation on a pyramid of sparsely populated uniform grids that make much better use of the available hardware memory bandwidth. We first briefly review this approach and subsequently highlight the necessary modifications required for power diagrams. To identify cells and faces that carry degrees of freedom, Setaluri et al. [2014] defined *active* cells and faces as follows:

- A cell at a given level of the pyramid is *active* if it is geometrically present and undivided in the octree.
- A face at a given level is *active* if that face is geometrically present and undivided in the octree, implying that every active face has at least one active cell neighbor at that level.

For efficiently emulating the building blocks of adaptive fluid simulation, they used two fundamental algorithmic kernels: stencil operations within a single level, and transfer operations between adjacent levels in the pyramid. This additionally requires the concept of *ghost* cells. Suppose we index each level with $1 \le l \le L$ where lower indices denote finer grids, and let C_I^l denote a cell that natively lives at level l with multi-dimensional index I. Then, a cell C_I^l is *ghost* if all three of the following conditions are *jointly* met:

(1) C_I^l is not active at level l,
(2) C_I^l neighbors a cell that is active at level $s \le l$, and
(3) An active, *coarser* parent of C_I^l exists at some level $l^\star > l$.

Figure 5 shows the pyramid of sparse grids, along with all ghost cells and active cell/face degrees of freedom. To transfer data between adjacent levels of the pyramid, two more operators are used:

(1) GhostValuePropagate(): an upsampling routine in which data from level l is *copied to* fine ghost children at level $l-1$,
(2) GhostValueAccumulate(): a downsampling routine in which data in level l *accumulates* contributions from any fine ghost children at level $l-1$.

Essentially, ghost cells provide a mechanism to combine information that is shared across different levels of resolution, while only focusing effort one level at a time, enabling optimizations native to uniform grids. A ghost cell can either be a source of a numerical value originating from a coarser level in the pyramid (by mirroring the coarse value), or a placeholder for the partial result of an operation that would have otherwise required access to a coarser variable. We refer the reader to the work of Setaluri et al. [2014] for details on how this representation can be used to efficiently compute the discrete gradient, divergence, and Laplace operators.

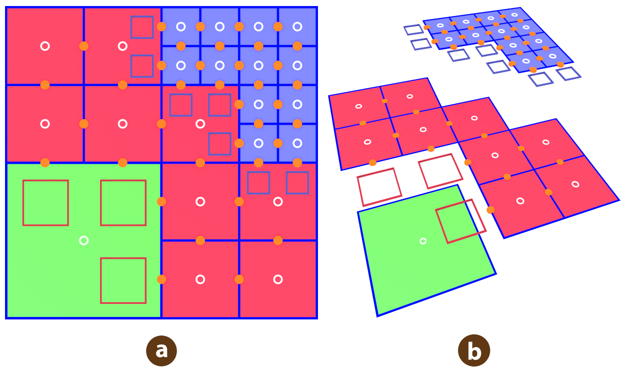


Fig. 5. Pyramid of sparse uniform grids. All active cell and face degrees of freedom are shown along with ghost cells (square outlines) at each level.

Enhancements for power diagrams. Since power diagrams may introduce new faces between cells that previously only shared an edge, the definition of ghost cells requires some modifications. In the second of the three criteria we previously listed for identifying a ghost cell, Setaluri et al. [2014] considered only face neighbors of C_I^l to check if any of them were active at level $s \le l$; we now examine all octree cells that share an edge with C_I^l (in 3D), and treat those that share a face of the power diagram with C_I^l as additional neighbors in deciding whether a ghost cell should be spawned. Moreover, their implementation of the discrete Poisson operator (which was based on the formulation of Losasso et al. [2004]) yielded stencils with closed-form coefficients even at level transitions. By contrast, our stencil coefficients are determined by the geometry of the power diagram. To efficiently retrieve these coefficients at run-time, we adopt certain restrictions on our grading scheme that make the power diagram geometry of a cell dependent *only* on its 1-ring neighborhood. This allows us to retrieve stencil coefficients via a look-up table, indexed by a compact descriptor of the local topology as we propose in Section 6.

As mentioned in Section 4.1, we store velocities at all power cell faces. For faces that are exactly homologous to faces on the original octree, similar to Setaluri et al. [2014], we dedicate 3 channels for storing the individual components of the velocity field (in the standard octree, those would have been the u, v, w axis-components of the velocity field; in our case they are normal velocity components relative to the respective power diagram faces). In our power diagram, however, faces of the power diagram (and correspondingly normal velocity variables that require storage) can exist in 3D between cells that were edge neighbors in the original octree. An edge of the octree can have up to four cells incident to it; however only two of those four cells can actually be connected in any given power diagram under our formulation. Specifically, it is possible for the pair of cells indexed $C_{i,j,k}$ and $C_{i,j+1,k+1}$ (sharing an edge aligned with the x-axis) to share a face of the power diagram. Likewise, cells $C_{i,j+1,k}$ and $C_{i,j,k+1}$ (incident on the same edge, as far as the octree is concerned) could instead share a face of the power diagram. However, *at most one* of the aforementioned pairs could have a power diagram face connecting them, in any given scenario. Thus, we can associate the normal velocity component associated with either of these two cases with the same octree edge, with the understanding that at most one of these two neighboring relations can be present at the same time; the determination of which of the two cases (if any) we have can be made by examining the local octreee topology. Hence, velocity components associated with such cases of edge neighbors can be conceptually stored on 3 offset grids, each of them centered on the midpoint of edges aligned with the x, y, and z axes, respectively. We simply dedicate three additional channels in SPGrid to store those special velocity components (we would have called them *edge-centered velocities* if such a concept existed in the original octree). We do not explicitly store a description for the direction of this edge-centered velocity component, as this will be automatically reflected in the stencils of those cells incident to it.

4.3 Multigrid preconditioner

Our power diagram discretization, paired with a topology encoding scheme (see section 6) will allow the Laplace operator to be applied implicitly without building its explicit matrix form. This gives the opportunity to use a matrix-free method such as Conjugate Gradient for solving the Poisson equation. Naturally, a good preconditioner is essential; Setaluri et al. [2014] used a very effective adaptive multigrid preconditioner. However, their approach was specifically tailored for the first order accurate discretization of Losasso et al. [2004]. Since both our power diagram discretization, as well as the first-order accurate approach [Losasso et al. 2004; Setaluri et al. 2014] have exactly the same set of pressure variables, it would be natural to start with the multigrid V-cycle preconditioner proposed by Setaluri et al. [2014] as a baseline, and attempt to adapt it to the power diagram discretization. The most straightforward way to make such an adaptation would be to replace the discretization at the finest level of the hierarchy with our newly proposed power diagram formulation, retaining the first-order accurate operators at all coarser levels. This adaptation is quite effective; in our tests it consistently achieves preconditioning efficiency comparable with what Setaluri et al. [2014] report for the first-order problem.

Although this approach is effective, there is a modest penalty to be paid in terms of implementation efficiency. The single most costly component of a multigrid V-cycle is the smoothing routine at the finest resolution of the discretization hierarchy. This step is now using our second order power diagram discretization which, although quite efficient, does not permit the full extent of aggressive optimizations of the first order scheme [Setaluri et al. 2014], which does not require local topology queries to define its stencils. Thus, we have implemented a hybrid alternative: We first invoke the smoothing routine, using the second order power diagram discretization, but iterate it only near boundaries and level transitions. We subsequently update the residual, and use a *first order accurate* multigrid V-cycle to solve the error equation and generate a correction across the entire domain. We follow up with a second sweep of smoothing the second order discretization (near boundaries and level transitions) to ensure symmetry of the preconditioner thus crafted. Let us denote by $\mathbf{L}_1$ the first order Laplace operator from Setaluri et al. [2014], and let $\mathbf{M}_1 \approx \mathbf{L}_1^{-1}$ be the multigrid preconditioner they defined in their work. Finally, let $\mathbf{L}_2$ be our second order Laplace discretization, using the power diagram. We define a new preconditioner $\mathbf{M}$, whose action $\mathbf{w} = \mathbf{M}\mathbf{q}$ is computed as follows:

(1) Starting with a zero initial guess $\mathbf{p}_0 = \mathbf{0}$, execute k iterations of the damped Jacobi method on the equation $\mathbf{L}_2\mathbf{p} = \mathbf{q}$ (input vector is used as right hand side). This is only applied to a band (about 3 voxels wide) around the boundaries and level transitions. Let $\mathbf{p}_1$ be the iterate that results from this operation, and $\mathbf{r}_1 = \mathbf{q} - \mathbf{L}_2\mathbf{p}_1$ the associated residual.
(2) Compute a correction $\delta\mathbf{p} = \mathbf{M}_1\mathbf{r}_1$ by applying the first order accurate preconditioner $\mathbf{M}_1$ from Setaluri et al. [2014] to the residual $\mathbf{r}_1$. Add the correction to $\mathbf{p}_1$, to obtain a new approximation $\mathbf{p}_2 = \mathbf{p}_1 + \delta\mathbf{p}$.
(3) Repeat k additional iterations of the same Jacobi method on the equation $\mathbf{L}_2\mathbf{p} = \mathbf{q}$, but this time use $\mathbf{p}_2$ as the initial guess. Let $\mathbf{w}$ denote the result at the end of this iteration.

We can easily verify that vectors $\mathbf{w}$ and $\mathbf{q}$ are related by a linear map $\mathbf{w} = \mathbf{M}\mathbf{q}$ (there is nothing in steps 1-3 above that introduces any nonlinearity, and it is easy to verify that when $\mathbf{q} = \mathbf{0}$ we would also have $\mathbf{w} = \mathbf{0}$). In fact, a careful look at the matrix formula for the Jacobi iteration reveals that the entire matrix $\mathbf{M}$ *is symmetric and positive definite*, assuming that $\mathbf{M}_1$ is SPD as well. We use the algorithm above to implicitly apply $\mathbf{M}$ as a preconditioner to Conjugate Gradient for the system $\mathbf{L}_2\mathbf{p} = \mathbf{f}$. In our examples, this preconditioner achieved very similar convergence performance to the straightforward alternative approach of using a second order operator at the finest level of a V-cycle preconditioner, achieving satisfactory convergence within 6-10 iterations across all resolutions. It was necessary to perform an adequate number ($k \approx 8$) of boundary smoothing iterations in steps (1) and (3) of the aforementioned preconditioner application to achieve this good convergence behavior, but the ability to restrict this effort to just the boundary region (while using an aggressively optimized first order V-cycle) made this a favorable trade-off at large resolutions.

5 INTERVENTIONS IN THE SIMULATION PIPELINE

To track the dynamically evolving liquid surface, we designed a level set scheme similar in spirit to prior work [Bargteil et al. 2006; Bojsen-Hansen and Wojtan 2013; Goldade et al. 2016; Kim et al. 2009] in the sense that we use the SPGrid data structure to store an Eulerian description of a narrow band around the free surface at a significantly higher resolution than the velocity data — typically by a factor of 4 or 8 (see Figure 6). Thus, we use two separate meshes, a background simulation mesh that is an octree, and an interface tracking grid (or fine level set grid) that is a single SPGrid (rather than a pyramid). Note that the velocities natively live *only* on the octree, and we also store and evolve a separate (coarse) level set on the octree because our fine representation does not carry inside/outside information beyond a narrow band of the free surface. The fine level set representation only requires an SPGrid with 4 channels (1 channel for Boolean flags associated with domain geometry, 1 for the level set, and 2 more channels for fast marching).

Level set advection. The time step Δt is determined by the finest effective resolution of the octree, since the velocities live natively on the octree. However, this time step will be overly dissipative for advecting the fine level set forward in time using a standard semi-Lagrangian update, and may require more expensive schemes for volume conservation [Lentine et al. 2012]. We follow a more

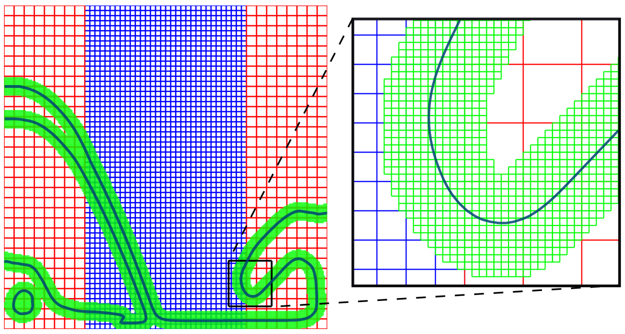
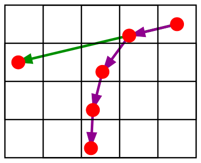


Fig. 6. A time-evolving Eulerian description of a narrow band around the interface is stored at a significantly higher resolution (shown in green) in our framework. This representation is used to correct the level set field on the octree after every advection step.

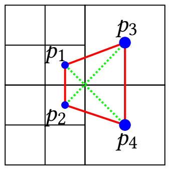
accurate approach where we divide this time step by m, where m is the factor by which the fine level set resolution is higher than the effective resolution of the octree. Next, we take m steps (of forward Euler) backwards along the velocity field, where each step is performed with a time step size of $\Delta t/m$.



At each step, we reach a new point p where we compute a new interpolated velocity before taking the next step, tracing a piecewise linear curve in the process (see the figure inset). After the m^{th} step we interpolate the level set, and assign its value back to the starting cell of the trajectory. Note that this approach still allows for efficient parallelization without decreasing the time step size according to the fine level set resolution. Of course, we maintain a wide enough band so that the free surface still falls inside this band after advection. Compared to simply upsampling the background octree, tracing back a piecewise linear curve for semi-Lagrangian advection foregoes the numerical dissipation that would have been introduced by m individual advection and reinitialization steps, producing more accurate results (see Figure 7). For advecting forward the coarse level set stored on the octree, we follow the approach of Setaluri et al. [2014]. Subsequently, we use the fine level set to correct the coarse level set wherever we have valid ϕ-values.

Velocity interpolation and advection. The discretization of velocity advection is more subtle as not all faces are axis-aligned, and so velocities are not neatly segregated into orthogonal components. Fortunately, this problem has been studied in prior work on unstructured mesh fluid simulation. To interpolate velocities at arbitrary points, we first compute full velocity vectors at all cell centers. For regular uniform cells, we average the axis-aligned face velocities; for all power cells, we perform a least squares fit based on all the face normal components [Feldman et al. 2005]. The dual mesh of our power diagram consists of cubes and tetrahedra, whose vertices are the octree cell centers. Therefore, having constructed velocity vectors at the vertices of this mesh, we apply trilinear interpolation inside cubic cells and barycentric interpolation on tetrahedra. For improved efficiency and accuracy in regular regions away from level transitions, we revert to standard per-axis face-based velocity interpolation. With this interpolation procedure at hand, we compute full velocity vectors at the centroid of each face and advect them using a standard semi-Lagrangian update, subsequently projecting the velocity onto the face normal direction. Likewise, we use this interpolant as needed for advecting the fine resolution level set.

Dynamic topology. After each advection step, the topology of the SPGrid storing the high resolution level set needs to be updated; new cells may emerge in the narrow band which require valid signed distance values, and old cells may fall outside the band and should be deleted. We do this by instantiating a new SPGrid and copying over level set values for all cells at the interface. Subsequently, we run fast marching on this SPGrid, activating new cells wherever necessary. For efficiency, we first allocate all pages corresponding to blocks that either contain the interface, or lie in the 1-ring of a block that contains the interface. After this step, cells that lie within the narrow band can be safely marked in parallel without any data hazards. We note that Setaluri et al. [2014] speculated on the utility of an enhancement to SPGrid, that could actually discard physical pages no longer in use (in Linux, this would be via a variant of the `madvise` system call), to allow dynamic additions and removals to the *same* SPGrid. We did not find such an optimization to be worthwhile here, as evaluating refinement criteria at the same time while discarding prior data is an unnecessary complexity.



Fast marching and velocity extrapolation. We store level set values ϕ on cell centers of the octree, and solve for fast marching in uniform regions similar to Foster and Fedkiw [2001]. Near T-junctions we adopt the approach of Sethian and Vladimirsky [2000] and run fast marching on a tetrahedral mesh. Note that this approach mandates that the solid angle of each tetrahedron incident at the center of the current cell should be less than $\pi/2$, and so we use the *Delaunay* tetrahedralization of the centers of the face/edge neighbors and the current cell. Note that ϕ-values were stored at grid nodes by Losasso et al. [2004], and they simply ignored missing neighbors near T-junctions as their grading scheme specifically coarsened *away* from the interface. However, our discretization for the pressure specifically requires ϕ-values at the cell centers to obtain second order accuracy near the free surface [Gibou et al. 2002], and we adapt the simulation grid topology even across the interface, thus our different approach. A slight subtlety is that these tetrahedral meshes should be computed *locally* per cell. To understand this consider the 2D illustration shown in the figure inset: $\Delta p_1p_2p_4$ and $\Delta p_1p_3p_4$ must be used at the point p_1 to ensure an acute angle, while $\Delta p_2p_1p_3$ and $\Delta p_2p_3p_4$ should be used at the point p_2. To extrapolate velocities outside the liquid region we first interpolate velocities from power faces to *regular* octree faces and use an approach similar to fast marching, this time operating on faces instead of cells and copying over the velocity value from the face closest to the free surface. For regular uniform cells, identifying face neighbors is straightforward, while near T-junctions we use the connectivity of the tetrahedral mesh to identify all faces incident to cells that correspond to nodes in the mesh. Subsequently, we interpolate back velocities from regular octree faces to all power faces. Figure 7 shows a comparison of the particle level set method [Enright et al. 2002] and our interface tracking scheme for a sphere that undergoes a deformation in a circular velocity field. The background grid resolution is 128^2 in 2D (or 128^3 in 3D); particle level set has 256 particles per cell, and our fine level set has 256 fine cells in 2D (or 4096 fine cells in 3D) per coarse cell.

6 TOPOLOGY ENCODING AND OPERATOR STORAGE

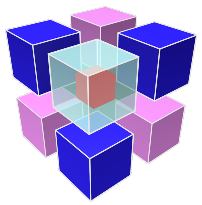
The affinity of power diagrams to octrees is a crucial precondition for leveraging aggressive optimizations. In Section 4, we discussed how ghost cells in a pyramid of sparse uniform grids can facilitate the transfer of information between variables at different levels. It should be evident from our earlier discussion that the performance potential of our proposed storage paradigm is strongly predicated on the ability to easily load and store neighboring variables, and efficiently perform stencil applications. We now describe our scheme for encoding the local mesh connectivity and the stencils of the discrete Laplace and divergence operators in a compact form.

6.1 Encoding of local power diagram topology

Our encoding scheme is based on the observation that the local structure of each power cell is completely determined by its *face and edge neighbors*. The octree neighborhood for each cell can be trivially inferred from the SPGrid pyramid; in fact, every relevant property of its neighborhood can be inferred by looking exclusively at a *single* level of the pyramid. If a face/edge neighbor exists at the same resolution, it is flagged as active. Otherwise, if that same neighbor is flagged as ghost, there is a *coarser* neighbor on the other side of that face/edge. If a neighbor is neither present as an active or ghost cell, cells of *finer* resolution must reside on the other side. To ensure that each neighborhood admits a well-defined encoding, we grade such that all neighbors of a cell are (a) at the same resolution as the current cell or one coarser, or (b) at the same resolution as the current cell or one finer. We discuss both these cases below:

Cells with no coarser neighbor. If a cell has no ghost neighbors, all its neighbors reside at the same resolution as itself or one level *finer*, giving rise to exactly 2^{18} distinct possibilities: each of the 6 face neighbors and 12 edge neighbors (18 neighbors total) is present at the same resolution, or absent (in which case four finer neighbors are present across each face, and two finer neighbors across each edge). We assemble this into an 18-bit number $N = b_0 b_1 b_2 \cdots b_{18}$, where every bit is defined as

$$b_j = \begin{cases} 1, & \text{if the } j\text{-th neighbor exists at same resolution} \\ 0, & \text{otherwise (neighbors are finer)} \end{cases}$$



Cells with no finer neighbor. This is the complement of the previous case; our grading restriction ensures that for such a cell, all its neighbors reside at the same resolution as itself or one level *coarser*. To elucidate our encoding scheme, consider a cell (orange) and its coarser parent (light blue); there are 8 possible arrangements of the child cell within its parent, and for each of these configurations, there are only 6 possible coarse neighbors allowable due to parity reasons (see figure inset). We use 3 bits to encode the position of the child cell within its parent, and 6 bits to indicate whether this cell has a coarse neighbor in the X, Y, and Z (face) directions, and XY, YZ, and XZ (edge) directions. For each configuration of the child cell, there can only be *one* coarse neighbor along each of these directions, making this encoding unique and well-defined. Thus, a total of 9 bits are required for this case.

Ultimately, we dedicate one more bit to encode which of the above two cases we have at hand, for a maximum of 19 bits that can capture all possible local topologies afforded by our grading scheme.

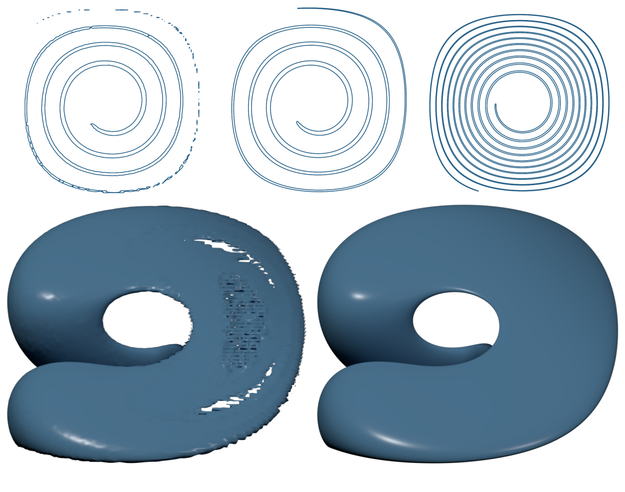


Fig. 7. (Top) A comparison of (left) particle level set on a 128^2 grid, and (middle) our interface tracking scheme on a 2048^2 grid for a deforming sphere at $t = 5$ seconds; (right) our scheme at $t = 15$ seconds. (Bottom) A comparison of (left) particle level set on a 128^3 grid and (right) our method on a 2048^3 grid. For particle level set, near-interface cells use 256 particles. The narrow band level set conserves volume well, over long time periods.

6.2 Retrieval of local Delaunay tetrahedralizations

Section 5 described our approach of using local Delaunay tetrahedralizations for fast marching and velocity interpolation; here we discuss their storage. Under our grading scheme, there are at most 114 geometrically distinct possibilities for the neighborhood of an octree cell; 18 of those are face and edge neighbors at the same resolution, 48 are cells of one finer resolution (4 cells across each of the 6 faces, and 2 across each of the 12 edges), and 48 are cells of one coarser resolution (8 configurations of a child in its parent cell, and 6 possible coarse neighbors in each case). Hence, all tetrahedra can only choose their vertices from 114 distinct cells, *relative* to the current cell. As a consequence, each vertex only requires a byte to encode (or 4 bytes per tetrahedron). We construct a lookup table with a (conservatively reserved) maximum of 128 tetrahedra incident to each node, requiring 512 bytes per topological case, or slightly more than 128MB to store the table for each of the $2^{18} + 2^9$ different local topologies. We populate this table on-demand, constructing the contents on-the-fly for any newly encountered case; our experiments have indicated that this table is very sparsely occupied, making it even easier for its contents to remain cached.

6.3 Hierarchical evaluation of differential operators

Efficient evaluation of the discrete Laplacian operator is critical to the performance of the Conjugate Gradient solver. Our treatment

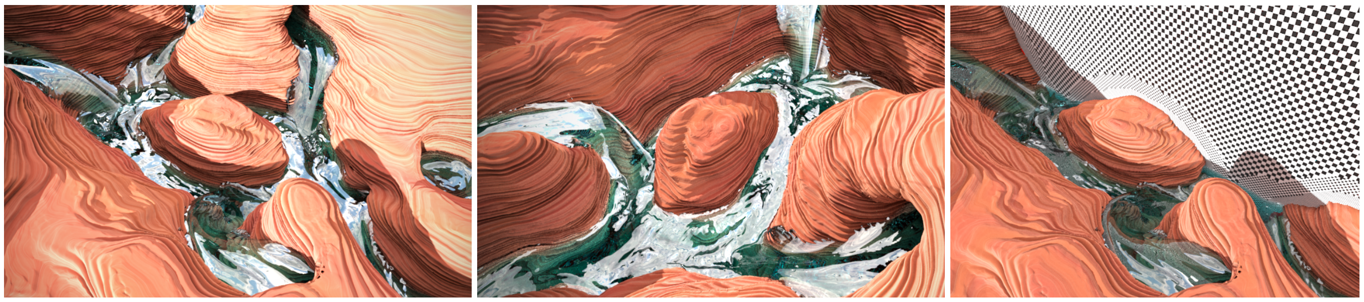


Fig. 8. Close-up view of the simulation of a river flowing through a canyon, previously shown in Figure 1. The adaptivity pattern, based on proximity to rigid boundaries, is shown along a vertical cross-section, on the right. A narrow band level set with resolution $2048^2 \times 4096$ is used for interface tracking.

leverages the symmetry of this operator, allowing us to only store the "convenient half" of the coefficient pairs. We first use the routine GhostValuePropagate() helper to populate all ghost cells with the value of their coarser parent. The following three cases arise:

Active cells with no finer neighbor The Laplacian for these cells can be readily computed using their 6 face neighbors and 12 edge neighbors (some of these neighbors may be ghost cells). We use a lookup table (identical in terms of indexing to the one storing the Delaunay tetrahedra) to retrieve the 19 stencil coefficients, corresponding to the central cell and its 18 neighbors. The storage cost is 19 floating point values or 76 bytes, for each of the 2^9 different topologies. To maximize alignment with cache lines, we pad this stencil up to 128 bytes, for a still nimble lookup table size of 64KB.

Ghost cells For each ghost cell, we use a 6 bit code to record which of its 6 topologically allowable neighbors (3 face and 3 edge neighbors, the direction being determined by parity) include this ghost cell in their stencil. Depending on the value of these bits, we look up the stencils of those neighboring cells and retrieve the coefficient for this ghost cell. All such contributions are added together. At the end of the Laplacian kernel, the routine GhostValueAccumulate() is invoked to accumulate all partial Laplace contributions temporarily deposited on ghost cells back to their coarse parents.

Active cells with no coarser neighbor In this case, we only consider the contribution of face/edge neighbors at the *same* resolution, and rely on the GhostValueAccumulate() routine to contribute the missing stencil spokes to all finer neighbors. We retrieve 19 coefficients from our lookup table of 2^{18} possible local topologies. As one would expect, these coefficients are zero for all fine neighbors.

The evaluation of the discrete divergence operator is similar, with the exception of accessing the velocity for the corresponding face from one of the 6 different velocity channels. This approach only requires the storage of 19 coefficients in all cases, and retains the performance potential offered by the SPGrid framework.

7 RESULTS

We simulated a number of examples to demonstrate the effectiveness of our framework. Each uses two or three levels of refinement, although this is not a strict limitation. For rendering the liquid surface, we used the fine level set wherever we had valid values, otherwise we used the coarse level set. Figure 2 shows two sources pouring water in a container, creating a thin sheet. The octree topology is fine near the sources and coarser away from them, with three levels of adaptivity. Figure 3 shows an example with dynamic adaptation, where four sources pour water in a pool with a rotating object. The octree topology is fine within a bounding box of the object and the sources, and coarser elsewhere, with two levels of adaptivity. The box enclosing the object is updated every frame, so that it can track complex solid-fluid interactions. During advection, whenever the mesh topology changes, we trace back from the new mesh face locations, but perform interpolation into the old mesh, thereby avoiding a separate mesh transfer step [Klingner et al. 2006]. Finally, Figure 8 shows river flow in a canyon, with three levels of adaptivity. The thin geometry of this domain mandates more smoothing effort near the boundary (10 Jacobi smoothing iterations, as opposed to 3 in the other examples), increasing the cost of the projection step.

Table 1 shows the memory bandwidth utilization for a streaming copy kernel (for reference), the first order Laplacian of Setaluri et al. [2014], our optimized second order Laplacian, and an unoptimized Laplacian that uses an explicit mesh near T-junctions to store the stencil coefficients. These computations were performed on the river data set from Figure 8. Our kernel achieves good performance by exploiting the ghost cell mechanism of the SPGrid pyramid. Table 2 shows the average timing breakdown of one time step for all our examples. The memory footprint of our fine level set representation was 8.09 GB (543M voxels) for the twin source example from Figure 2, 7.36 GB (493M voxels) for the rotating paddle example from Figure 3, and 3.84GB (253M voxels) for the river example from Figure 8. In spite of the exceptionally high resolution afforded in the narrow band, SPGrid yields a storage footprint which is quite acceptable for simulations of this scale. Finally, as Table 2 reveals, maintenance of the narrow band is efficient enough to not be the bottleneck, even though we rely on a serial Fast Marching method for reinitialization.

In the supplemental material, we use analytical test cases in two and three dimensions to provide numerical evidence that our projection technique achieves second order accuracy and our fast marching method achieves first order accuracy.

8 LIMITATIONS AND FUTURE WORK

Our proposed combination of a power diagram discretization, storage using an SPGrid pyramid, and the use of a highly refined narrow band level set for interface tracking exposes and exploits a number of opportunities for performance optimizations, but also incurs some conscious limitations. In this section, we highlight the most notable limitations, and discuss whether they are intrinsic to our approach, or a temporary exclusion from our scope that could be amended without fundamental changes to our framework.

Our interface tracking scheme is expressly capable of capturing geometric details at higher resolutions than its counterpart stored on the power diagram. Since the dynamics are driven by the level set as sampled onto the octree power diagram, sub-grid droplets or air pockets may at times be overlooked by the simulation, leading to non-physical motions such as the slender suspended splash in the rotating paddle scenario. Therefore, a sub-grid treatment similar to prior work [Bojsen-Hansen and Wojtan 2013; Goldade et al. 2016] is called for. In addition, our interface tracking scheme uses first order semi-Lagrangian advection and the fast marching method for reinitialization, both of which are known to be overly dissipative. In the future, we would like to explore improved advection schemes, such as FLIP [Zhu and Bridson 2005] or MacCormack [Selle et al. 2008], and higher order accurate level set reinitialization. Reconstructing full velocities at tetrahedra circumcenters rather than nodes and applying generalized barycentric interpolation over polyhedra would also reduce dissipation [Elcott et al. 2007; Klingner et al. 2006]. Moreover, in principle, it is possible to use adaptivity even for interface tracking while, for simplicity, we have restricted our implementation to a single uniform grid. We are presently reinitializing the signed distance with a serial Fast Marching method on the narrow band; alternative reinitialization schemes that admit parallelism certainly merit attention in future work. Finally, given that our method makes extensive use of the SPGrid data structure to solve the Navier-Stokes equations, tracking the surface with a high-resolution narrow band SPGrid level set was a natural and convenient choice. Nevertheless, our underlying octree fluid solver is not intrinsically tied to this choice, and we expect that it could alternatively be paired with a purely particle- or mesh-based surface tracking strategy if desired.

Thus far we have sought to simulate strictly large-scale single-phase inviscid free surface flows, which leaves ample room for future exploration of more specialized effects such as surface tension, contact-line dynamics, viscosity, multiple phases, solid-fluid interaction, and so on. Some of these extensions should be straightforward; for example, explicit surface tension should involve only a minor modification to the right-hand-side of the linear system based on surface curvature [Enright et al. 2003].

As discussed in Section 4, edge neighbors in the octree may share a face in the power diagram. We use three additional channels associated with edges for storing velocities on these faces. These channels only store meaningful values near T-junctions, resulting in a non-optimal storage density. It is also possible to use the non-graded approach of Losasso et al. [2006] in regions deep interior to the liquid, and our power diagram discretization only near the free surface to obtain a lower cost Laplace operator that still yields a second order accurate pressure field. We have tested this idea on a simple prototype and plan to include it in our simulation pipeline in future work. Finally, we have consciously restricted our scope to a grading scheme that mandates that a cell and all its neighbors span only two levels of resolution; however, there exist weaker grading rules that do not destroy the essential octree structure in the power diagram, and lifting this restriction would not prevent us from using the ghost cell mechanism within the SPGrid framework either. However, in the absence of this grading restriction, our encoding and lookup scheme for local tessellations and stencils would not apply. We look forward to investigating algorithmic paradigms for more intricate adaptive topology patterns in future work.

Streaming Copy	1st Order Laplacian	2nd order Laplacian	Unoptimized Laplacian
14.45	5.71	3.95	0.49

Table 1. Memory bandwidth utilization (in GB/sec) for streaming and stencil operations run on an Intel Xeon E3-1241 at 3.5GHz.

	Fig. 2	Fig. 3	Fig. 8
Time Step	80	58	81
Level Set Advection	26	14	14
Reinitialization	20	15	19
Velocity Advection	5	2	5
Projection	24	18	40
Velocity Extrapolation	4	3	2
Grid Adaptation	N/A	5	N/A
Number of PCG iterations	5	4	10

Table 2. Average timing breakdown (in seconds) for all examples.

REFERENCES

David Adalsteinsson and James A. Sethian. 1995. A fast level set method for propagating interfaces. *J. Comp. Phys.* 118 (1995), 269–277.

Bart Adams, Mark Pauly, Richard Keiser, and Leonidas J. Guibas. 2007. Adaptively sampled particle fluids. *ACM Trans. Graph. (SIGGRAPH)* 26, 3 (2007), 48.

Ryoichi Ando, Nils Thuerey, and Chris Wojtan. 2013. Highly adaptive liquid simulations on tetrahedral meshes. *ACM Trans. Graph. (SIGGRAPH)* 32, 4 (2013), 103.

Ryoichi Ando, Nils Thuerey, and Chris Wojtan. 2015. A dimension-reduced pressure solver for liquid simulations. *Computer Graphics Forum (Eurographics)* 34, 2 (2015), 473–480.

Franz Aurenhammer. 1987. Power diagrams: properties, algorithms, and applications. *SIAM J. Comput.* 16, 1 (1987), 78–96.

Adam W. Bargteil, James F. O'Brien, Tolga G. Goktekin, and John A. Strain. 2006. A semi-Lagrangian contouring method for fluid simulation. *ACM Trans. Graph.* 25, 1 (2006), 19–38.

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

Christopher Batty, Stefan Xenos, and Ben Houston. 2010. Tetrahedral embedded boundary methods for accurate and flexible adaptive fluids. *Computer Graphics Forum (Eurographics)* 29, 2 (2010), 695–704.

Morten Bojsen-Hansen and Chris Wojtan. 2013. Liquid surface tracking with error compensation. *ACM Trans. Graph. (SIGGRAPH)* 32, 4 (2013), 79:1–79:10.

Robert Bridson. 2015. *Fluid simulation for computer graphics, 2nd edition.* A. K. Peters, Ltd.

Tyson Brochu, Christopher Batty, and Robert Bridson. 2010. Matching fluid simulation elements to surface geometry and topology. *ACM Trans. Graph. (SIGGRAPH)* 29, 4 (2010), 47.

Tyson Brochu and Robert Bridson. 2009. Robust topological operations for dynamic explicit surfaces. *SIAM J. Sci. Comput.* 31, 4 (2009), 2472–2493.

Nuttapong Chentanez, Bryan E. Feldman, François Labelle, James F. O'Brien, and Jonathan Richard Shewchuk. 2007. Liquid simulation on lattice-based tetrahedral meshes. In *Symposium on Computer Animation.* ACM Press, 219–228.

Nuttapong Chentanez, Tolga G. Goktekin, Bryan E. Feldman, and James F. O'Brien. 2006. Simultaneous coupling of fluids and deformable bodies. In *Symposium on Computer Animation.* ACM Press, 83–89.

Nuttapong Chentanez and Matthias Müller. 2011. Real-time Eulerian water simulation using a restricted tall cell grid. *ACM Trans. Graph. (SIGGRAPH)* 30, 4 (2011), 82.

Nuttapong Chentanez and Matthias Müller. 2012. Mass-conserving Eulerian liquid simulation. In *Symposium on Computer Animation.* 245–254.

Pascal Clausen, Martin Wicke, Jonathan Richard Shewchuk, and James F. O'Brien. 2013. Simulating liquids and solid-liquid interactions with Lagrangian meshes. *ACM Trans. Graph.* 32, 2 (2013), 17.

Fernando de Goes, Corentin Wallez, Jin Huang, Dmitry Pavlov, and Mathieu Desbrun. 2015. Power particles: an incompressible fluid solver based on power diagrams. *ACM Trans. Graph. (SIGGRAPH)* 34, 4 (2015), 50.

Mathieu Desbrun and Marie-Paule Gascuel. 1996. Smoothed particles: a new paradigm for animating highly deformable bodies. In *Eurographics Workshop on Computer Animation and Simulation.* 61–76.

Christian Dick, Marcus Rogowsky, and Rüdiger Westermann. 2016. Solving the fluid pressure Poisson equation using multigrid - evaluation and improvements. *IEEE TVCG* 22, 11 (2016), 2480–2492.

Sharif Elcott and Peter Schröder. 2006. Building your own DEC at home. In *ACM SIGGRAPH Course Notes.* 55–59.

Sharif Elcott, Yiying Tong, Eva Kanso, Peter Schröder, and Mathieu Desbrun. 2007. Stable, circulation-preserving, simplicial fluids. *ACM Trans. Graph.* 26, 1 (2007), 4.

Robert Elliot English, Linhai Qiu, Yue Yu, and Ronald Fedkiw. 2013a. An adaptive discretization of incompressible flow using a multitude of moving Cartesian grids. *J. Comp. Phys.* 254 (2013), 107–154.

Robert Elliot English, Linhai Qiu, Yue Yu, and Ronald Fedkiw. 2013b. Chimera grids for water simulation. In *Symposium on Computer Animation.* ACM Press, 85–94.

Doug Enright, Stephen Marschner, and Ronald Fedkiw. 2002. Animation and rendering of complex water surfaces. *ACM Trans. Graph. (SIGGRAPH)* 21, 3 (2002), 736–744.

Doug Enright, Duc Nguyen, Frédéric Gibou, and Ron Fedkiw. 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.

Bryan E. Feldman, James F. O'Brien, and Bryan M. Klingner. 2005. Animating gases with hybrid meshes. *ACM Trans. Graph. (SIGGRAPH)* 24, 3 (2005), 904–909.

Florian Ferstl, Rüdiger Westermann, and Christian Dick. 2014. Large-scale liquid simulation on adaptive octree grids. *IEEE TVCG* 20, 10 (2014), 1405–1417.

Nick Foster and Ronald Fedkiw. 2001. Practical animation of liquids. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01).* 23–30.

Nick Foster and Dimitris Metaxas. 1996. Realistic animation of liquids. *Graphical Models and Image Processing* 58, 5 (1996), 471–483.

Frédéric Gibou, Ron Fedkiw, Li-Tien Cheng, and Myungjoo Kang. 2002. A second order accurate symmetric discretization of the Poisson equation on irregular domains. *J. Comp. Phys.* 176, 1 (2002), 205–227.

Abhinav Golas, Rahul Narain, Jason Sewall, Pavel Krajcevski, Pradeep Dubey, and Ming Lin. 2012. Large-scale fluid simulation using velocity-vorticity domain decomposition. *ACM Trans. Graph. (SIGGRAPH Asia)* 31, 6 (2012), 148.

Ryan Goldade, Christopher Batty, and Chris Wojtan. 2016. A practical method for high-resolution embedded liquid surfaces. *Computer Graphics Forum (Eurographics)* 35, 2 (2016), 233–242.

Eran Guendelman, Andrew Selle, Frank Losasso, and Ronald Fedkiw. 2005. Coupling water and smoke to thin deformable and rigid shells. *ACM Trans. Graph. (SIGGRAPH)* 24, 3 (2005), 973–981.

Arthur Guittet, Mathieu Lepilliez, Sebastian Tanguy, and Frédéric Gibou. 2015a. Solving elliptic problems with discontinuities on irregular domains: the Voronoi Interface Method. *J. Comp. Phys.* 298 (2015), 747–765.

Arthur Guittet, Maxime Theillard, and Frédéric Gibou. 2015b. A stable projection method for the incompressible Navier-Stokes equations on arbitrary geometries and adaptive Quad/Octrees. *J. Comp. Phys.* 292 (2015), 215–238.

Nambin Heo and Hyeong-Seok Ko. 2010. Detail-preserving fully-Eulerian interface tracking framework. *ACM Trans. Graph. (SIGGRAPH Asia)* 29, 6 (2010), 176:1—-176:8.

Jeong-Mo Hong and Chang-Hun Kim. 2005. Discontinuous fluids. *ACM Trans. Graph. (SIGGRAPH)* 24, 3 (2005), 915–920.

Geoffrey Irving, Eran Guendelman, Frank Losasso, and Ronald Fedkiw. 2006. Efficient simulation of large bodies of water by coupling two and three dimensional techniques. *ACM Trans. Graph. (SIGGRAPH)* 25, 3 (2006), 805–811.

Chenfanfu Jiang, Craig Schroeder, Andrew Selle, Joseph Teran, and Alexey Stomakhin. 2015. The affine particle-in-cell method. *ACM Trans. Graph. (SIGGRAPH)* 34, 4 (2015), 51.

Doyub Kim, Oh-young Song, and Hyeong-Seok Ko. 2009. Stretching and wiggling liquids. *ACM Trans. Graph. (SIGGRAPH Asia)* 28, 5 (2009), 120.

Theodore Kim and Ming C. Lin. 2007. Fast animation of lightning using an adaptive mesh. *IEEE TVCG* 12, 2 (2007), 390–402.

Bryan M. Klingner, Bryan E. Feldman, Nuttapong Chentanez, and James F. O'Brien. 2006. Fluid animation with dynamic meshes. *ACM Trans. Graph. (SIGGRAPH)* 25, 3 (2006), 820–825.

Zuzana Krivá, Angela Handlovičová, and Karol Mikula. 2016. Adaptive cell-centered finite volume method for diffusion equations on a consistent quadtree grid. *Advances in computational mathematics* 42, 2 (2016), 249–277.

Michael Lentine, Mridul Aanjaneya, and Ronald Fedkiw. 2011. Mass and momentum conservation for fluid simulation. In *Symposium on Computer Animation.* 91–100.

Michael Lentine, Matthew Cong, Saket Patkar, and Ronald Fedkiw. 2012. Simulating free surface flow with very large time steps. In *Symposium on Computer Animation.* 107–116.

Michael Lentine, Wen Zheng, and Ronald Fedkiw. 2010. A novel algorithm for incompressible flow using only a coarse grid projection. *ACM Trans. Graph. (SIGGRAPH)* 29, 4 (2010), 114.

Haixiang Liu, Nathan Mitchell, Mridul Aanjaneya, and Eftychios Sifakis. 2016. A scalable schur-complement fluids solver for heterogeneous compute platforms. *ACM Trans. Graph. (SIGGRAPH)* 35, 6 (2016), 201.

Frank Losasso, Ronald Fedkiw, and Stanley Osher. 2006. Spatially adaptive techniques for level set methods and incompressible flow. *Computers & Fluids* 35, 10 (2006), 995–1010.

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

David F. Martin and Kelley L. Cartwright. 1996. *Solving Poisson's equation using adaptive mesh refinement.* Technical Report. EECS Department, University of California, Berkeley. 19 pages.

Aleka McAdams, Eftychios Sifakis, and Joseph Teran. 2010. A parallel multigrid Poisson solver for fluids simulation on large grids. In *Symposium on Computer Animation.* 65–74.

Michael L. Minion. 1996. A projection method for locally refined grids. *J. Comp. Phys.* 127, 1 (1996), 158–178.

Marek Misztal and Andreas Bærentzen. 2012. Topology-adaptive interface tracking using the deformable simplicial complex. *ACM Trans. Graph.* 31, 3 (2012), 24.

Marek Misztal, Robert Bridson, Kenny Erleben, Andreas Bærentzen, and François Anton. 2010. Optimization-based fluid simulation on unstructured meshes. In *VRIPHYS.* 11–20.

Jeroen Molemaker, Jonathan M. Cohen, Sanjit Patel, and Jonyong Noh. 2008. Low viscosity flow simulations for animation. In *Symposium on Computer Animation.* 9–18.

Patrick Mullen, Pooran Memari, Fernando de Goes, and Mathieu Desbrun. 2011. HOT: Hodge-optimized triangulations. *ACM Trans. Graph. (SIGGRAPH)* 30, 4 (2011), 103.

Matthias Müller. 2009. Fast and robust tracking of fluid surfaces. In *Symposium on Computer Animation.* ACM, New York, NY, USA, 237–245.

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

Michael B. Nielsen and Robert Bridson. 2016. Spatially adaptive FLIP fluid simulations in Bifrost. In *ACM SIGGRAPH talks.* 41.

Blair Perot and V. Subramanian. 2007. Discrete calculus methods for diffusion. *J. Comp. Phys.* 224, 1 (2007), 59–81.

Stéphane Popinet. 2003. Gerris: a tree-based adaptive solver for the incompressible Euler equations in complex geometries. *J. Comp. Phys.* 190, 2 (2003), 572–600.

Andrew Selle, Ronald Fedkiw, Byungmoon Kim, Yingjie Liu, and Jarek Rossignac. 2008. An unconditionally stable MacCormack method. *SIAM J. Sci.Comput.* 35, 2-3 (2008), 350–371.

Rajsekhar Setaluri, Mridul Aanjaneya, Sean Bauer, and Eftychios Sifakis. 2014. SPGrid: A sparse paged grid structure applied to adaptive smoke simulation. *ACM Trans. Graph. (SIGGRAPH Asia)* 33, 6 (2014), 205.

James Sethian. 1999. *Level set methods and fast marching methods.* Cambridge University Press.

James A. Sethian and Alexander Vladimirsky. 2000. Fast methods for the Eikonal and related Hamilton-Jacobi equations on unstructured meshes. *Proceedings of the National Academy of Sciences* 97, 11 (2000), 5699–5703.

Adamandios Sifounakis, Sangseung Lee, and Donghyun You. 2016. A conservative finite volume method for incompressible Navier-Stokes equations on locally refined nested Cartesian grids. *J. Comp. Phys.* 326 (2016), 845–861.

Funshing Sin, Adam W. Bargteil, and Jessica K. Hodgins. 2009. A point-based method for animating incompressible flow. In *Symposium on Computer Animation.* ACM Press, 247–255.

Barbara Solenthaler and Markus Gross. 2011. Two-scale particle simulation. *ACM Trans. Graph. (SIGGRAPH)* 30, 4 (2011), 81.

Jos Stam. 1999. Stable fluids *(SIGGRAPH '99).* 121–128.

Tetsuya Takahashi and Ming C. Lin. 2016. A Multilevel SPH Solver with Unified Solid Boundary Handling. *Computer Graphics Forum* 35, 7 (2016), 517–526.

Chris Wojtan, Nils Thuerey, Markus Gross, and Greg Turk. 2009. Deforming meshes that split and merge. *ACM Trans. Graph. (SIGGRAPH)* 28, 3 (2009), 76.

Jihun Yu, Chris Wojtan, Greg Turk, and Chee Yap. 2012. Explicit mesh surfaces for particle based fluids. *Computer Graphics Forum (Eurographics)* 31, 2 (2012), 815–824.

Xinxin Zhang and Robert Bridson. 2014. A PPPM fast summation method for fluids and beyond. *ACM Trans. Graph. (SIGGRAPH Asia)* 33, 6 (2014), 206.

Bo Zhu, Wenlong Lu, Matthew Cong, Byungmoon Kim, and Ronald Fedkiw. 2013. A new grid structure for domain extension. *ACM Trans. on Graph.(SIGGRAPH)* 32, 4 (2013), 63.

Yongning Zhu and Robert Bridson. 2005. Animating sand as a fluid. *ACM Trans. Graph. (SIGGRAPH)* 24, 3 (2005), 965–972.