A Practical Octree Liquid Simulator with Adaptive Surface Resolution

RYOICHI ANDO, National Institute of Informatics, Tokyo
CHRISTOPHER BATTY, University of Waterloo, Canada

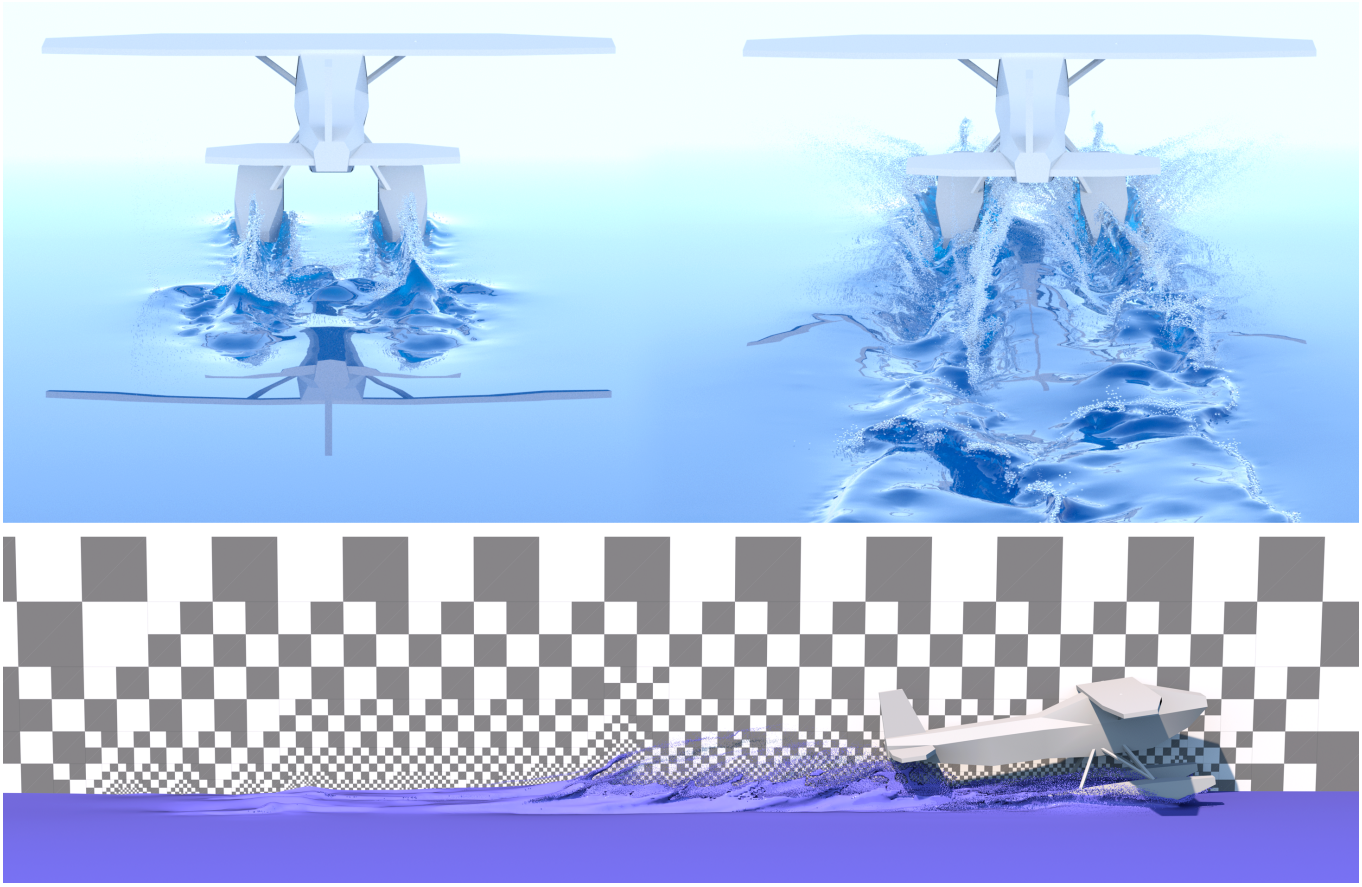

Fig. 1. Top: a seaplane (wingspan: 11m; height: 4.3m) takes off from a lake (30m depth) at high speed. Effective resolution: $1024 \times 1024 \times 512$. Compute time: 2.85 minutes per video frame. Bottom: the background octree cells, exhibiting a wide range of resolutions.

We propose a new adaptive liquid simulation framework that achieves highly detailed behavior with reduced implementation complexity. Prior work has shown that spatially adaptive grids are efficient for simulating large-scale liquid scenarios, but in order to enable adaptivity *along the liquid surface* these methods require either expensive boundary-conforming (re-)meshing or elaborate treatments for second order accurate interface conditions. This complexity greatly increases the difficulty of implementation and maintainability, potentially making it infeasible for practitioners. We therefore present new algorithms for adaptive simulation that are comparatively easy to implement yet efficiently yield high quality results. First, we develop a novel staggered octree Poisson discretization for free surfaces that is second order in pressure and gives smooth surface motions even across octree T-junctions, without a power/Voronoi diagram construction. We augment this discretization with an adaptivity-compatible surface tension force that likewise supports T-junctions. Second, we propose a moving least squares strategy for level set and velocity interpolation that requires minimal knowledge of the local tree structure while blending near-seamlessly with standard trilinear interpolation in uniform regions. Finally, to maximally exploit the flexibility of our new surface-adaptive solver, we propose several novel extensions to sizing function design that enhance its effectiveness and flexibility. We perform a range of rigorous numerical experiments to evaluate the reliability and limitations of our method, as well as demonstrating it on several complex high-resolution liquid animation scenarios.

Authors' addresses: Ryoichi Ando, National Institute of Informatics, Tokyo, rand@nii.ac.jp; Christopher Batty, University of Waterloo, Canada, christopher.batty@uwaterloo.ca.

0730-0301/2020/7-ART32 $15.00
https://doi.org/10.1145/3386569.3392460

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

Additional Key Words and Phrases: fluid simulation, liquid, octrees

ACM Reference Format:
Ryoichi Ando and Christopher Batty. 2020. A Practical Octree Liquid Simulator with Adaptive Surface Resolution. *ACM Trans. Graph.* 39, 4, Article 32 (July 2020), 17 pages. https://doi.org/10.1145/3386569.3392460

1 INTRODUCTION

Simulation practitioners deciding whether to adopt a given method must consider how best to realize their ultimate objectives with limited time and resources, both human and computational. The implementation complexity and expected benefits of a method are therefore critical factors. For example, staggered grids, FLuid-Implicit-Particle (FLIP) [Zhu and Bridson 2005], and Affine-Particle-in-Cell (APIC) [Jiang et al. 2015] have been widely incorporated into the film production toolkit due to their relative ease of implementation, proven reliability, and clearly demonstrated advantages. On the other hand, spatially adaptive fluid simulation has seen comparatively slow adoption in visual effects [Nielsen and Bridson 2016], despite a long history of work on this topic. We believe there are primarily two reasons for this: significant implementation costs, due to algorithmic complexity, and difficulties in achieving performance gains on actual simulations, as compared to highly optimized regular grid codes.

All adaptive schemes inevitably exhibit greater complex than regular grids, but this is especially the case for methods that allow adaptivity *along* the surface itself. For example, tetrahedral schemes [Ando et al. 2013; Clausen et al. 2013; Misztal et al. 2012] require comparatively elaborate mesh generation and traversal. The overset grids of English et al. [2013] generate unstructured Voronoi cells at grid-grid boundaries, similar to pure Voronoi and power diagram schemes [Brochu et al. 2010; de Goes et al. 2015; Zhai et al. 2018]. Aanjaneya et al. [2017] uses an *implicit* power diagram layered on top of an octree for efficiency, but the connectivity is nevertheless complex to precompute and manipulate (e.g., requiring extra diagonal neighbors and large precomputed lookup tables).

Therefore, the aim of this paper is to simplify the algorithms necessary to achieve *surface-adaptive octree liquid simulation* without compromising the quality of the resulting visual details. We provide an extensive variety of numerical tests and practical evaluations of our method, along with comparisons against the uniform staggered grid method, so that practitioners can evaluate the benefits and expected performance gains that our method makes possible.

At the heart of our method is a novel adaptive grid Laplace operator that supports free surface and solid boundary conditions in the presence of level transitions. For each cell, its stencil involves only the neighboring cells with which it shares a face in the octree. That is, compared to the prior hybrid octree/power diagram method of Aanjaneya et al. [2017], our method works directly on the basic octree and requires neither additional diagonal neighbor connectivity nor elaborate precomputation.

Our Laplace discretization comes in two flavors. If *strict* second order accuracy is desired at the free surface, our approach yields a non-symmetric Poisson system which can, for example, be efficiently solved with preconditioned BiCGStab. If a Symmetric Positive Definite (SPD) matrix is desired, in order to access the numerical benefits of Preconditioned Conjugate Gradients (PCG), we offer a slight modification in certain geometric configurations that nevertheless provides qualitatively indistinguishable results.

With this foundation for surface-adaptive liquid simulation in place, we present several additional enhancements to its expressiveness, quality, and convenience. To support surface-tension effects in the presence of adaptivity, we propose a new T-junction compatible approach based on a reinterpretation of the uniform grid method of Enright et al. [2003]. To simplify interpolation while minimizing additional dissipation, we adopt a Moving Least Squares (MLS) strategy that has much less dependence on the intricate local structure of the octree, while yielding the same result as trilinear interpolation in uniform regions. To take full advantage of the surface-adaptivity offered by our method, an effective dynamic octree sizing function is also critical; we therefore propose several extensions to prior sizing function methods to better track moving details, improve temporal coherence, offer user-controllability, and reduce octree initialization and adaptation costs.

In summary, the main novel contributions of this paper are:

- A surface-adaptive octree liquid simulation framework with reduced implementation complexity.
- A new pair of pressure discretizations for octree T-junctions near liquid surfaces that offer smooth surface motion and qualitatively indistinguishable results, and exhibit either symmetric positive definiteness or strict second order accuracy.
- A compatible octree surface tension force that supports varying surface resolution by generalizing the method of Enright et al. [2003].
- An approach to easily interpolate velocity and level set values near T-junctions based on Moving Least Squares, which transitions seamlessly to trilinear interpolation on regular cells.
- Several enhancements to sizing function design that together provide artist control and more effective grid adaptation according to key features of the motion and geometry.

Along the way, we suggest a set of complementary algorithms to round out our high-quality liquid simulation pipeline. This includes: Extended Narrow Band FLIP (EXNBFLIP) [Sato et al. 2018] to enrich the detail of splashes, an octree grid construction that does not require explicit balancing, and a cell-based adaptation of dual contouring [Ju et al. 2002] for reconstructing adaptive surfaces from cell-centered octree level set data.

2 PREVIOUS WORK

Spatial adaptivity for fluid simulation has a long history in computer graphics, as well as computational fluid dynamics. We primarily focus our review on the former for brevity.

Octree discretizations of fluid dynamics were first considered in computational physics by Popinet [2003] with a non-symmetric scheme. Subsequently, Shi and Yu [2004] developed the first octree-based adaptive method to animate smoke and Losasso et al. [2004] proposed the first octree-based liquid solver on non-graded trees.

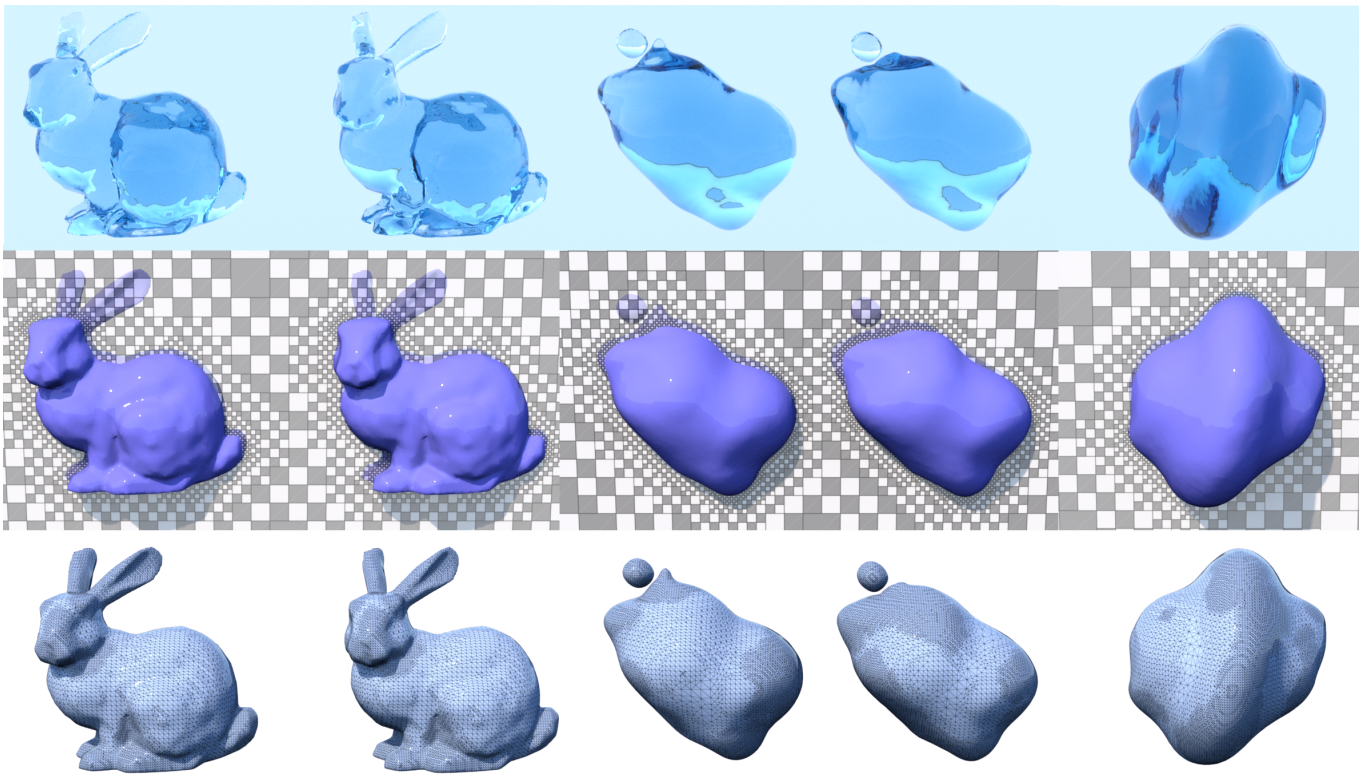

Fig. 2. A pair of liquid bunnies in zero gravity collapse due to surface tension and eventually merge (top), simulated with varying octree resolution (middle), including adaptive surface detail (bottom). Effective resolution: 256^3. Compute time: 1.6 minutes per video frame.

The latter method was improved to second order accuracy in pressure [Losasso et al. 2006] with a minor modification, thereby eliminating problematic motion artifacts. These methods can be used with standard cut-cell and/or ghost-fluid methods to support second order accuracy with irregular solid [Batty et al. 2007; Ng et al. 2009] and free surface [Enright et al. 2003] boundary conditions, respectively. However, doing so requires the entire boundary to be uniformly refined so that it does not interact with T-junctions (grid level transitions); this may be tremendously wasteful depending on the scenario under consideration. Ferstl et al. [2014] proposed a finite element-based cut-cell discretization for octree-based fluids with free surfaces, but again assumed a uniformly refined interface. Nielsen and Bridson [2016] similarly proposed a finite element-based approach on (generalized) octrees used within the Bifrost simulator, using mass-lumping to achieve a sparser Laplace stencil. While the details are not described in their paper, the framework currently supports adaptivity along solid boundaries, but not at free surfaces (R. Bridson, personal communication, April 27, 2020). Setaluri et al. [2014] presented a new data structure to efficiently support octrees using a pyramid of sparsely allocated grids.

Unstructured or semi-structured tetrahedral and/or polyhedral meshes are an attractive alternative for adaptivity, since they do not generally exhibit T-junction configurations. Klingner et al. [2006] was the first to exploit spatially adaptive tetrahedral meshes for smoke animation, with Chentanez et al. [2007] extending these ideas to liquid animation. Support for embedded/sub-grid boundaries in adaptive Delaunay tetrahedral meshes (with a Voronoi dual mesh) was proposed by Batty et al. [2010], which is made possible by the absence of T-junctions; Ando et al. [2013] similarly showed a finite element-like ghost fluid discretization on tetrahedra that offers varying resolution along the liquid surface, supported by a novel surface-adaptive sizing function. Voronoi meshes can likewise incorporate embedded boundaries and support adaptivity as illustrated by Brochu et al. [2010], including surface tension, building on the earlier point-based Voronoi liquid solver of Sin et al. [2009]. A closely related alternative is power diagrams [de Goes et al. 2015; Zhai et al. 2018] which possess a primal-dual mesh orthogonality that Voronoi diagrams also share. Aanjaneya et al. [2017] observed that a graded octree could be conceptually overlaid with a power diagram to effectively eliminate T-junctions from the pressure solver, and thereby recover accurate support for irregular embedded boundaries that cross grid levels. However, compared to our work, this entails either explicit power diagram meshing or complex precomputation of geometric cases, and gives a somewhat denser structure in the Poisson matrix. Many of the preceding schemes adopt specialized mesh-dependent interpolation schemes for velocity or level set values; an exception is the (conceptually meshless) Moving Least Squares [Nealen 2004] approach used for tetrahedral meshes by

Feldman et al. [2005]. We propose a related scheme that is tailored to the octree setting to avoid dissipation and ensure continuity with uniform (trilinear) regions. MLS has also been used by Sousa et al. [2019] inside the Poisson solve to construct ghost data for non-graded octree discretizations that yield second order gradients (in the absence of complex boundaries), and by Guittet et al. [2015] for more costly *quadratic* interpolation, using inverse distance weights rather than our trilinear kernel. MLS has seen many more applications in animation, including recent improvements to the material point method [Hu et al. 2018].

A large variety of additional adaptivity strategies have been presented in the literature, which we survey briefly below. The use of overset uniform grids in axis-aligned (Adaptive Mesh Refinement or AMR) or more general configurations is traditional in computational fluid dynamics [Berger and Oliger 1984]. The latter case is sometimes referred to as a Chimera grid as considered, for example, by English et al. [2013] who used a Voronoi diagram to stitch different regular grids together. Localized regions of tall cells were considered by Irving et al. [2006] and Chentanez and Müller [2011] to avoid the cost of simulating deep liquid regions that have little impact on the surface. Zhu et al. [2013] suggested using more general axially stretched regular grids for adaptivity in multiple directions. Others have considered warped [Ibayashi et al. 2018] and curvilinear [Azevedo and Oliveira 2013] grids, which maintain the connectivity of a uniform grid but significantly deform to locally increase resolution or match object boundaries. Lastly, adaptive Lagrangian schemes for conforming tetrahedral meshes have also been explored (e.g., [Clausen et al. 2013; Misztal et al. 2012]), and while these can support surface-adaptivity including surface tension, they suffer from significant remeshing costs.

3 METHOD OVERVIEW

3.1 Notation

For clarity, hereafter $[\cdot]$ denotes a matrix (or discrete linear operator) and $\{\cdot\}$ denotes a discretized variable expressed in vector form. Symbols without [] or {} express continuous variables/operators or a single-valued scalar/vector field. For a vector variable q_i the subscript denotes the ith scalar element of the vector. A single subscript i on a matrix variable denotes the ith row of the matrix. We use q^T to express the transpose of a matrix/vector. A list of symbols is available in Table 1. While our derivation is non-trivial, the resultings equations are relatively straightforward; we indicate equations that are important for the final implementation using a gray background. Throughout, we use the notation <vids/xxx.mp4> to indicate specific supplemental video materials.

3.2 Overview

We simulate liquid by solving the incompressible Euler equations,

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\frac{1}{\rho}\nabla p + \boldsymbol{g}, \qquad \nabla \cdot \boldsymbol{u} = 0, \tag{1}$$

where $\boldsymbol{u}, t, \rho, p$ and $\boldsymbol{g}$ denote velocity, time, density, pressure and gravity, respectively. For convenience, we set $\rho = 1$ for liquid, and treat air as a vacuum with $\rho = 0$. For regular cells, we use a standard staggered grid layout and operator-splitting [Bridson 2015] to integrate time. For interface tracking we use the level set method [Osher et al. 2004] and enforce second order accurate boundary conditions both on free surfaces [Enright et al. 2003] and solids [Batty et al. 2007; Ng et al. 2009]. We later describe an (optional) extension to incorporate extended narrow band FLIP [Sato et al. 2018] for enriched splash details.

We use basic semi-Lagrangian advection for both the velocity and the level set due to its ease of use. MacCormack advection is also possible, but as Selle et al. [2008] discuss, first order semi-Lagrangian advection is recommended for velocity near liquid surfaces anyway. We did not observe significant improvement in animation quality using MacCormack, and therefore we did not prefer it, given its added storage and computational cost. This is illustrated in our supplemental material in <vids/maccormack.mp4>. The added cost of MacCormack may be worth paying in smoke simulation contexts.

Similar to previous methods, we start with construction of the octree grid. Next, we advect the level set and velocity values from the previous step's octree grid and simultaneously assign the resulting

Table 1. List of symbols used in the paper.

Sections from 4.1 to 4.5			
Sym	Type	Location/Dim	Description
$\boldsymbol{u}$	Vector	Continuous	Incompressible velocity
$\{\boldsymbol{u}\}$	Vector	Face centers	Discretized velocity above
$\boldsymbol{u}^*$	Vector	Continuous	Advected velocity
$\{\boldsymbol{u}^*\}$	Vector	Face centers	Discretized advected velocity
p	Scalar	Continuous	Pressure
$\{p\}$	Vector	Cell centers	Discretized pressure on cells
k	Integer	Per face	Reference number of a face
∇	Operator	$\mathbb{R} \rightarrow \mathbb{R}^3$	Gradient operator
$\nabla\cdot$	Operator	$\mathbb{R}^3 \rightarrow \mathbb{R}$	Divergence operator
$[\nabla]$	Matrix	Faces × Cells	Discretized gradient operator
$[\nabla]_k$	Matrix	1 × Cells	kth row of the above matrix
$[\nabla]^T$	Matrix	Cells × Faces	Discretized $-\nabla\cdot$ operator
$\boldsymbol{e}_k$	Vector	Per face	Face normal (unit vector)
$[A]$	Matrix	Faces × Faces	Area fraction (diagonal)
$[F]$	Matrix	Faces × Faces	Inverse of fluid fraction (diag)
$[V]$	Matrix	Faces × Faces	Face volume (diag)
ϕ	Scalar	Continuous	Level set of liquid
$\{\phi\}$	Vector	Cell centers	Discretized level set
$\boldsymbol{n}$	Vector	Continuous	Liquid surface normal
$\boldsymbol{Q}_k^*$	Set	Per face	All the cells referenced in $[\nabla]_k$
$\boldsymbol{Q}_k$	Set	Per face	Cells inside liquid in $\boldsymbol{Q}_k^*$
W_k	Scalar	Per face	Scalar coefficient per face
Sections from 4.5 to 7			
$\{\boldsymbol{u}^\star\}$	Vector	Continuous	Pre-modified $\boldsymbol{u}^*$
ϕ_{solid}	Scalar	Continuous	Level set of solid
$N(\cdot)$	Function	$\mathbb{R}^3 \rightarrow \mathbb{R}$	Trilinear shape function
$S(\cdot)$	Function	$\mathbb{R}^3 \rightarrow \mathbb{R}$	Sizing function
$\{S\}$	Scalar	Cell centers	Sizing value
κ	Constant	N/A	Surface tension parameter
$\{\kappa H\}$	Scalar	Cell centers	Scaled mean curvature
$\boldsymbol{x}$	Vector	Per cell/face	Discretized sample position
$\boldsymbol{p}$	Vector	Arbitrary	Query position

values to the new octree grid (as first demonstrated for tetrahedra by Klingner et al. [2006]). Optionally, we seed extended FLIP particles and advect/delete existing ones. We then enforce incompressibility through Chorin's projection [Chorin 1968]. Finally, we redistance the level set and extrapolate velocity outwards.

4 OCTREE PRESSURE DISCRETIZATION

4.1 Pressure projection overview

Since the central component of our method is the pressure projection, we begin with a brief review of this step. Let $\{\boldsymbol{u}^*\}$ be the divergent velocity field after advection has been performed. The projection starts with solving a linear system,

$$-[\nabla]^T[V][A][F][\nabla]\{p\} = -[\nabla]^T[V][A]\{\boldsymbol{u}^*\}, \tag{2}$$

or, more simply,

$$[\nabla]^T[VA][F\nabla]\{p\} = [\nabla]^T[VA]\{\boldsymbol{u}^*\}, \tag{3}$$

where $[V]$, $-[\nabla]^T$, $[\nabla]$, $[A]$, and $[F]$ denote, respectively, the volume of a face (not the volume of a cell), a discrete divergence operator, a discrete gradient operator, a diagonal matrix of fluid area fractions (second order accurate Neumann boundary conditions for solids [Ng et al. 2009]) and the inverse of a diagonal liquid fraction matrix (second order accurate Dirichlet boundary conditions for liquid [Enright et al. 2003]). If we assume that grid cells are all uniform, $[V]$ can simply be an identity matrix. Next, the solution for the pressure $\{p\}$ is used to update the divergent velocity $\{\boldsymbol{u}^*\}$ to complete the projection:

$$\{\boldsymbol{u}\} = \{\boldsymbol{u}^*\} - [F][\nabla]\{p\}. \tag{4}$$

To form (3) the key question is how to discretize $[\nabla]$, $[F\nabla]$ and $[VA]$, which we detail in the next subsections.

4.2 Discretizing spatial variables

We discretize spatial variables according to the method of [Losasso et al. 2006]. Consistent with staggered regular grids, we sample pressure and level set values at the center of cells and velocity components on faces. At T-junction faces, we sample velocity only at the center of the large (parent) face; the incident small (child) faces effectively inherit this velocity. We always assume a 2:1 graded tree for simplicity, as did Aanjaneya et al. [2017]). This is illustrated in Figure 3.

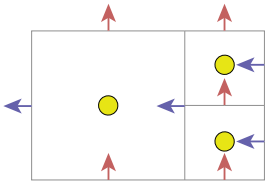

Fig. 3. Locations of pressure (yellow circle) and velocity (blue and red arrows).

4.3 Discretizing the gradient operator

This subsection describes the computation of $[\nabla]$ in (3). Once $[\nabla]$ is available, we can compute $-[\nabla]^T$ (in practice, this is done by a local matrix transpose), noting that $[\nabla]^T = -[\nabla\cdot]$. Figure 4 left illustrates a 2D example of the discrete gradient along the horizontal direction on a T-junction. We first compute the average pressure value and average position between the two smaller cells' centers. Next, we measure the distance L between this average position and

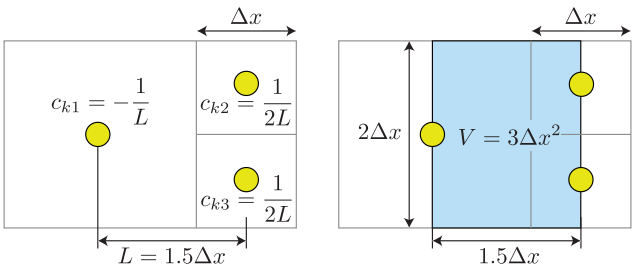

Fig. 4. 2D discrete gradient (left) and the volume (right) on a T-junction face. c_{k1},c_{k2},c_{k3} denote coefficients of $[\nabla]_k$: the gradient of the kth face in the face normal direction.

the center of the large cell. Finally, a finite difference estimate of the gradient at the face is constructed using the difference of the average pressure value and that of large cell divided by L. The analogous procedure is employed for the 3D case (that is, the coefficients for smaller cells will be $1/(4L)$ in 3D while L remains the same). Note that in this way the differentiation direction is aligned with the face normal, and as such overall second order accuracy in pressure is preserved (in the absence of boundary conditions for now) [Batty et al. 2010; Losasso et al. 2006]. Note that this is not the case for some adaptive methods [Chentanez et al. 2007; Feldman et al. 2005; Losasso et al. 2004]. Gradients on faces of regular cells are computed with standard centered finite differences. To summarize, let $[\nabla]_k$ be the kth row of $[\nabla]$ where k denotes an (integer) reference to a face. Then $[\nabla]_k$ is given in 2D by

$$[\nabla]_k^{2\mathrm{D}} = \mathrm{sgn(face)}\begin{bmatrix}-1 & 1/2 & 1/2\end{bmatrix}/(1.5\Delta x), \tag{5}$$

where the leftmost entry corresponds to a large cell, and the remaining entries to small cells. The expression $\mathrm{sgn(face)} \in \{-1, 1\}$ indicates the sign of the face direction relative to the large cell, which is 1 for Figure 4, for example. The symbol Δx is the grid cell size of the smaller side. Similarly, $[\nabla]_k$ in 3D is given by

$$[\nabla]_k^{3\mathrm{D}} = \mathrm{sgn(face)}\begin{bmatrix}-1 & 1/4 & 1/4 & 1/4 & 1/4\end{bmatrix}/(1.5\Delta x). \tag{6}$$

For faces incident on two regular cells,

$$[\nabla]_k = \begin{bmatrix}-1 & 1\end{bmatrix}/\Delta x, \tag{7}$$

for both 2D and 3D (a sign flip is not needed). Therefore, for faces whose incident cell centers are all inside the liquid, either (5), (6) or (7) is plugged into (3) as $[F\nabla]_k$.

4.4 Assembling the diagonal matrix

This subsection describes the computation of $[VA]$ in (3). For convenience, we use the same terminology "volume" when referring to face (edge) area in 2D and face volume in 3D. Figure 4 right illustrates a 2D example of the face volume on a T-junction. As in our gradient expressions, the same length L is used to measure the distance between incident cell centers. The remaining dimensions are straightforward to compute, leading to the expressions $V_{2\mathrm{D}} = 3\Delta x^2$ for

volume in 2D, and $V_{3D} = 6\Delta x^3$ for volume in 3D. When a face crosses a solid boundary, we compute the area fraction $0 < A < 1$ which encodes the non-solid volume fraction of the face, i.e., one minus the face's solid fraction [Ng et al. 2009]. The product VA is inserted as the diagonal entry of $[VA]$ in (3) for that face.

4.5 Second order accuracy on T-junctions near surfaces

The preceding sections are sufficient to achieve a second order accurate pressure projection on simple domains and the liquid interior, and is consistent with the method of Losasso et al. [2006]. However, careful treatment of non-axis-aligned surface configurations is known to be essential for ensuring smooth motion [Enright et al. 2003]. This subsection therefore describes the computation of $[F\nabla]$ in (3) on T-junction faces crossing an irregular liquid surface. Once $[F\nabla]$ is available, we can form and solve (3). Consider three cells incident on a T-junction that crosses the interface (Figure 5). Cell centers inside the liquid are colored yellow and the outside center is colored red. In the discussion below, we assume that all the spatial variables are linear within the T-junction cells, and any quadratic or higher order changes are ignored (which was also implicitly assumed in earlier derivations). Under these conditions, pressure p and level set value ϕ along the line p_1 and p_2 are given by

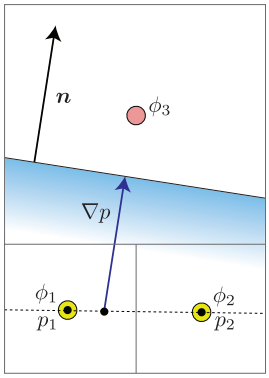

Fig. 5. Second order accurate free surface boundary conditions on a T-junction.

$$p = \theta_1 p_1 + \theta_2 p_2, \quad \phi = \theta_1\phi_1 + \theta_2\phi_2, \tag{8}$$

$$\theta_1 = w_1/(w_1 + w_2), \quad \theta_2 = w_2/(w_1 + w_2), \tag{9}$$

where w_1 and w_2 are arbitrary numbers that can be freely chosen for convenience. Notice that the constraint $\theta_1 + \theta_2 = 1$ is imposed by construction. Hence, the equations above describe either an interpolation or an extrapolation based on the two data points. More generally, a linear combination of an arbitrary number of data points is given by

$$p = \sum_{i\in Q_k} \theta_i\{p\}_i, \quad \phi = \sum_{i\in Q_k} \theta_i\{\phi\}_i, \quad \theta_i = w_i \Big/ \sum_{j\in Q_k} w_j, \tag{10}$$

where Q_k denotes a set of indices in $[\nabla]_k$ for which $\{\phi\}_i < 0$, indicating that cell center i resides in the liquid. The θ values again form a partition of unity. Since the gradient of a function is perpendicular to its isosurfaces, and our boundary conditions assume the pressure on the surface (and outside) is $p = 0$, the interior pressure gradient near the surface can be approximated as

$$\nabla p = \left(\frac{p-0}{\phi}\right)\boldsymbol{n} = \frac{p\boldsymbol{n}}{\phi}, \tag{11}$$

where $\boldsymbol{n}$ is the liquid surface normal in the vicinity of the T-junction cells. (A similar approximation was used by Bojsen-Hansen and Wojtan [2013].) Plugging the expressions (10) in for p and ϕ in (11) and simplifying gives:

$$\nabla p = \left(\sum_{i\in Q_k} \theta_i\{p\}_i\right)\boldsymbol{n} \Big/ \left(\sum_{i\in Q_k} \theta_i\{\phi\}_i\right) \tag{12}$$

$$= \left(\sum_{i\in Q_k} w_i\{p\}_i\right)\boldsymbol{n} \Big/ \left(\sum_{i\in Q_k} w_i\{\phi\}_i\right). \tag{13}$$

Notice that in (13) the θ parameters completely vanish and we are only left with the choice of w without constraints (except when the w_i are all zeros). Hence, the w_i need not necessarily sum to 1.

Now, let $[\nabla]$ be a discrete gradient operator (matrix) on a set of values $\{q\}_i$, where $\{q\}_i$ denotes arbitrary quantities sampled on the cell centers, including cells outside the liquid. Up to this point $[\nabla]$ has no dependence whatsoever on the free surface boundary; it is simply a discrete operator used to approximate ∇q. Recall that, given $[\nabla]$, a discrete divergence operator is conveniently given as $-[\nabla]^T$ and the Laplace operator $\nabla\cdot\nabla$ in the absence of free surfaces is approximated as $-[\nabla]^T[\nabla]$. Since this is the product of a matrix and its transpose, the resulting linear system is positive semidefinite by construction, and with at least one free surface (Dirichlet boundary value) it becomes SPD. Naturally we wish to impose that our resulting linear system near surfaces is also SPD.

Our next task is to determine the component of the pressure gradient in the face-normal direction, $\boldsymbol{e}_k\cdot\nabla p$, where $\boldsymbol{e}_k$ denotes the normal of the kth face, i.e., an axis-aligned unit vector. Let c_{kj} be the jth element of a single-row matrix $[\nabla]_k$ (that is, $c_{kj} = [\nabla]_{kj}$). In this setting, we have

$$\boldsymbol{e}_k\cdot\boldsymbol{n} = [\nabla]_k\{\phi\} = \sum_{j\in Q_k^*} c_{kj}\{\phi\}_j, \tag{14}$$

In the above we have used the relation $\boldsymbol{n} = \nabla\phi$, which holds since ϕ is a signed distance field. The symbol Q_k^* denotes the set of all cells referenced in $[\nabla]_k$, *including* cells outside the liquid. Using (14) with our pressure gradient approximations in (11) and (13), the pressure gradient component in the face normal direction is

$$\boldsymbol{e}_k\cdot\nabla p = \boldsymbol{e}_k\cdot\frac{p\boldsymbol{n}}{\phi} = (\boldsymbol{e}_k\cdot\boldsymbol{n})\frac{p}{\phi} \tag{15}$$

$$= \left(\sum_{j\in Q_k^*} c_{kj}\{\phi\}_j\right)\left(\sum_{i\in Q_k} w_i\{p\}_i\right) \Big/ \left(\sum_{i\in Q_k} w_i\{\phi\}_i\right). \tag{16}$$

Since this statement holds independent of the choice of w, we can specifically choose $w_j = c_{kj}$. By doing so, we get the following equations:

$$\boldsymbol{e}_k\cdot\nabla p = \left(\sum_{j\in Q_k^*} c_{kj}\{\phi\}_j\right)\left(\sum_{i\in Q_k} c_{ki}\{p\}_i\right) \Big/ \left(\sum_{i\in Q_k} c_{ki}\{\phi\}_i\right). \tag{17}$$

Recalling that we impose $p_i = 0$ outside the liquid, we have

$$\sum_{i\in Q_k} c_{ki}\{p\}_i = \sum_{j\in Q_k^*} c_{kj}\{p\}_j = [\nabla]_k\{p\}. \tag{18}$$

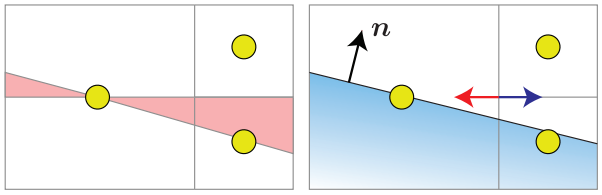

Fig. 6. Left: example of a degenerate zone in 2D with the liquid surface normal pointing upward. When the liquid surface lies within the red range, $W_k < 0$ and special care is needed. Right: the correct (face) normal direction (blue arrow) vs. the flipped normal direction (red arrow). Leaving $W_k < 0$ corresponds to non-physically flipping the normal direction of the pressure gradient, leading to "kink" artifacts.

Substituting (18) into (17) yields

$$\boldsymbol{e}_k \cdot \nabla p = \left(\sum_{j \in Q_k^*} c_{kj}\{\phi\}_j\right)\left(\sum_{j \in Q_k^*} c_{kj}\{p\}_j\right) \Big/ \left(\sum_{i \in Q_k} c_{ki}\{\phi\}_i\right) \quad (19)$$

$$= W_k[\nabla]_k\{p\}, \quad (20)$$

where

$$W_k = \left(\sum_{j \in Q_k^*} c_{kj}\{\phi\}_j\right) \Big/ \left(\sum_{i \in Q_k} c_{ki}\{\phi\}_i\right). \quad (21)$$

Therefore, this choice for w_j gives

$$\nabla p = \mathrm{diag}(W)[\nabla]\{p\}. \quad (22)$$

Applying the discrete divergence operator $-[\nabla]^T$ to (22) yields our discrete approximation of the Laplacian accounting for free surfaces:

$$\nabla \cdot \nabla p \approx -[\nabla]^T \mathrm{diag}(W)[\nabla]\{p\}. \quad (23)$$

The resulting linear system is only assuredly SPD if $W_k > 0$ (e.g., by Theorem 4.2.1 of Golub and Van Loan [2012]), but it is not uncommon to encounter geometric configurations for which $W_k < 0$. For example, such a case occurs when the interface lies entirely within the red region in Figure 6, left. In the worst case this results in negative diagonal terms in the resulting linear system.

When we encounter a case where $W_k < 0$, we can circumvent it either by setting $w_i = 1$ or clamping $W_k = 0$. Note that it is possible to leave W without any modification at all; however, we very occasionally observed small kinks on the surface when liquids nearly settle using this indefinite form. Physically, this is because $W_k < 0$ corresponds to a flip in the normal direction of the pressure gradient with respect to the liquid surface, as illustrated in Figure 6, right, which should not be allowed in practice.

The choice of setting $w_i = 1$ retains full second order accuracy, but the resulting linear system is asymmetric and therefore PCG cannot be used to solve it. We find that preconditioned BiCGStab can efficiently solve the system, although it lacks some of the favorable numerical and convergence properties of PCG.

The choice of clamping $W_k = 0$ instead sacrifices strict second order accuracy in exchange for recovering symmetric positive definiteness. Interestingly, this is also equivalent to assuming a solid face in that position, which is of course somewhat nonphysical. However, the SPD property ensures that PCG can be used and, moreover, we have in practice observed no resulting visual artifacts across a wide variety of simulation scenarios. A set of comparisons is provided in Figure 13. Note that clamping to zero does not necessarily imply zero diagonal terms in the resulting linear system: the contributions from other non-degenerate faces will still make the diagonal terms positive, except for the extreme case in which all the adjacent faces are degenerate (or solid). In our examples, we handle this by using a very small positive number ($W_k = 10^{-2}$) rather than 0 for faces adjacent to a cell for which at least one adjacent cell's face contains a solid fraction. Finally, the above expressions give $[F\nabla]$ as either

$$[F\nabla]_k = \frac{[\nabla]_k\{\phi\}}{\sum_{i \in Q_k}\{\phi\}_i}\begin{bmatrix}1 & \cdots & 1\end{bmatrix}, \quad (24)$$

for the fully second order case and

$$[F\nabla]_k = \max(W_k, 0)\,[\nabla]_k, \quad (25)$$

for the SPD case (in practice, albeit not unconditionally). See (21) for the computation of W_k. As described above, one may (infrequently) need to swap zero for a very small positive number if positive definiteness should be unconditionally guaranteed. When forming the above equations only the actual degrees of freedom of pressure must be considered. Hence, entries not in the actual degrees of freedom will be dropped. Our extension to support the presence of surface tension forces is described in Section 4.6. Interestingly, for regular faces our second order accurate approach produces exactly the same formula as the familiar ghost fluid method [Enright et al. 2003]. This can be seen by observing

$$p_G = \frac{\phi_G}{\phi_L}p_L, \quad \frac{p_G - p_L}{\Delta x} = \frac{\phi_G - \phi_L}{\phi_L \Delta x}p_L = \frac{[\nabla]_k[\phi_G\ \phi_L]^T}{\phi_L}p_L, \quad (26)$$

where p_G, p_L, ϕ_G, ϕ_L denote ghost pressure, pressure in the liquid cell, level set on the ghost cell, level set on the liquid cell. Notice that (26) is exactly the same as both (24) and (25). Our technique can alternatively be seen as an adaptation of the warped grid method of Ibayashi et al. [2018] to a variant of finite volume methods (their method is based on a finite element method); however, we also provide a more rigorous evaluation and a fully second order accurate non-symmetric alternative in the case of degenerate situations.

4.6 Surface tension forces

We develop a new surface tension formulation to complement our surface-adaptive pressure projection. We do so by generalizing the approach of Enright et al. [2003] to an arbitrary Chorin-style projection method [Chorin 1968]. Using H and κ to denote the mean curvature (of the form avoiding factors of two) and surface tension coefficient, respectively, we revisit (11) and replace the zero surface pressure with κH motivated by the Young-Laplace law:

$$\nabla p = \left(\frac{p - \kappa H}{\phi}\right)\boldsymbol{n}. \quad (27)$$

We compute H via level set differentiation [Bridson 2015], though a mesh-based evaluation could be used instead [Brochu et al. 2010;

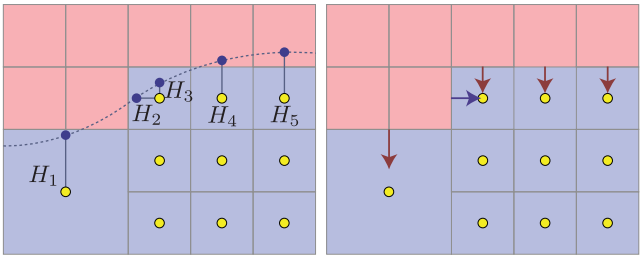

Fig. 7. Curvature/surface tension force computation on an octree grid. Red cells are air and blue cells are liquid. The dashed line indicates the liquid interface. Yellow circles represent active (i.e., interior) pressure samples and blue circles represent mean curvature evaluation points. Red and blue arrows represent the surface tension forces computed by applying $[F\nabla]\{\kappa H\}$ on the incident faces.

Thürey et al. 2010]. Equation 27 is equivalent to

$$\nabla p = \left(\frac{p-0}{\phi}\right)\boldsymbol{n} - \left(\frac{\kappa H - 0}{\phi}\right)\boldsymbol{n}, \tag{28}$$

which can in turn be conveniently rewritten using our discrete gradient operator as

$$\nabla p = [F\nabla]\left\{p - \kappa H\right\}. \tag{29}$$

The gradient operator $[F\nabla]$ is independent of the discretization choice, and henceforth $[F\nabla]$ can be either (24) or (25). This is made possible by our generalization from (27) to (29). (Our same final discrete equations could also be developed in the manner of Losasso et al. [2006], but the derivations would depend explicitly on which of our two gradient operators was adopted.)

Notice that in (29) we *store* mean curvature at the same position as the pressure samples, as illustrated in Figure 7, but the mean curvature itself is nevertheless still computed on the exact liquid surface position. In this way, a cell may have multiple mean curvature values; the appropriate value is selected depending on the relevant face in the gradient computation. On T-junction cells, we compute mean curvature *per T-junction*, hence the same mean curvature is assigned to multiple liquid cells. Next, we substitute (29) into (3) to get a new linear system,

$$[\nabla]^T[VA][F\nabla]\{p\} = [\nabla]^T[VA]\left(\{\boldsymbol{u}^*\} + [F\nabla]\{\kappa H\}\right). \tag{30}$$

The new velocity is then given as

$$\{\boldsymbol{u}\} = \{\boldsymbol{u}^*\} + [F\nabla]\{\kappa H\} - [F\nabla]\{p\}. \tag{31}$$

In practice, it is convenient to instead simply pre-modify $\{\boldsymbol{u}^*\}$ as

$$\{\boldsymbol{u}^\star\} = \{\boldsymbol{u}^*\} + [F\nabla]\{\kappa H\}, \tag{32}$$

and perform the standard projection on $\{\boldsymbol{u}^\star\}$ *without* considering surface tension. This pre-modification of $\{\boldsymbol{u}^*\}$ interpretation was also employed by Ando et al. [2015] in a different context. Our approach is therefore a decoupled, explicit, and sharp interface application of the surface tension force which can be used with any gradient discretization, but remains mathematically equivalent to the use of a boundary condition on the usual Poisson system [Enright et al. 2003].

5 OCTREE INTERPOLATION

5.1 Moving Least Squares Approach

We adapt a Moving Least Squares (MLS) approach for velocity and level set interpolations near T-junctions (for a concise summary of MLS, we recommend the review by Nealen [2004]). Regions where MLS is applied are shown in light blue, dark blue and green in the first row of Figure 9. However, our MLS interpolation scheme implicitly categorizes the colored regions shown in this figure without explicit knowledge of the grid structure. This conveniently allows us to interpolate arbitrarily positioned variables using the same routine. For example, when we interpolate an arbitrary discrete variable $\{q\}$ (e.g., a level set or velocity value) at a query point $\boldsymbol{p} = [p_x, p_y, p_z]$, we first collect a set of nearby samples. We discuss our sample collection strategy in Section 5.2. Let $\Delta\boldsymbol{x}$ be a vector of which the ith element represents the size of the ith cell/face, and let $\boldsymbol{x}_i = [x_i, y_i, z_i]$ be the position of the ith sample. The interpolated value $q(\boldsymbol{p})$ using MLS is given by

$$q(\boldsymbol{p}) = \begin{bmatrix} p_x & p_y & p_z & 1 \end{bmatrix}(Z^T D Z)^{-1} Z^T D \begin{bmatrix} \{q\}_1 \\ \vdots \\ \{q\}_n \end{bmatrix}, \tag{33}$$

$$Z = \begin{bmatrix} x_1 & y_1 & z_1 & 1 \\ \vdots & \vdots & \vdots & \\ x_n & y_n & z_n & 1 \end{bmatrix},\ D = \mathrm{diag}\left(\begin{bmatrix} N(\boldsymbol{p}-\boldsymbol{x}_1, \Delta x_1) \\ \vdots \\ N(\boldsymbol{p}-\boldsymbol{x}_n, \Delta x_n) \end{bmatrix}\right), \tag{34}$$

$$N(\boldsymbol{r}, h) = \prod_{j=1}^{3} \max(1 - \frac{||r_j||}{h}, \varepsilon), \tag{35}$$

where n denotes the number of points. The length h is the cell size of the grid level corresponding to a given sample point. The safety constant ε is used to avoid zero weights; this can occur if sample points are outside the trilinear kernel distance, as illustrated in dark blue in Figure 9. We set $\varepsilon = 10^{-2}$ for all our tests, and solved the normal equations linear system directly with a Cholesky solver.

We use trilinear interpolation for regular cells (red regions in Figure 9), except where the shape function of a larger grid resolution overlaps (green regions in Figure 9). Because our MLS gives the *exact same formula* as trilinear interpolation when applied on regular cells, our interpolation scheme is C^0 continuous, with the exception of the boundary of the dark blue region where ε has an effect. This equivalence is only true because we chose (35) for computing D, and will not be the case if we choose different shape functions (e.g., constant). We provide a sketch of a proof aided by a computer algebra package in the supplemental material (interpolation.maxima). This property also holds for 2D (bilinear interpolation).

If desired, a fully C^0 continuous interpolant can be constructed by inserting a set of ghost samples with anisotropic (axially scaled) trilinear kernels, as shown in the second row of Figure 9. However, this requires more explicit knowledge of the local grid structure, making the interpolation problem-specific (face centers vs. cell centers); it therefore comes with additional implementation complexity and effort. In practice, we find that the above (mildly discontinuous) interpolation scheme does not differ noticeably from the truly continuous interpolation scheme on actual simulations. A comparison

(both face and cell interpolations) is provided in the supplemental material (mildly discontinuous <vids/noncont_MLS.mp4>, C^0 continuous <vids/c0_MLS.mp4>). We also provide a number of numerical tests to clarify the discontinuity error and the quality of the above modified C^0 continuous interpolation scheme in Figure 17.

Special care is needed near boundaries of the simulation domain because there may be insufficiently many nearby points. In such cases, we mirror points across the boundary, as illustrated in Figure 8, and count them as additional ghost values.

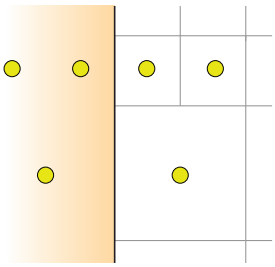

Fig. 8. Mirrored values. Samples adjacent to domain boundaries are mirrored across the boundary to prevent degenerate cases when performing MLS interpolation.

Note that mirrored ghost particles are generated only at the exterior (axis-aligned) domain boundary — we do not generate ghost particles within solids. As in our uniform MAC solver, we extrapolate level set and velocity values into solids, and use them during interpolation. Very thin solids may cause "cross-talk" artifacts, which could potentially be addressed by adapting a one-sided interpolation strategy (e.g., [Azevedo et al. 2016; Guendelman et al. 2005]).

Notably, we do not perform any (temporary) nodal/element-center velocity conversions [Aanjaneya et al. 2017; Batty et al. 2010; Brochu et al. 2010; Klingner et al. 2006; Losasso et al. 2004], which may require special care to prevent additional numerical diffusion. In our method, velocity is always separated into u,v,w components on faces, and our MLS interpolation scheme is applied independently to each. Hence, this potential source of numerical diffusion is avoided.

5.2 Sample point collection

An initial set of nearby samples from an arbitrary position $\boldsymbol{p}$ is determined by the condition $N(\boldsymbol{p}-\boldsymbol{x},\Delta x) > \varepsilon$ where $\boldsymbol{x}$ denotes a sample position and Δx its cell (or face) size. A brute-force approach *could* loop over all the samples and check if $N(\boldsymbol{p}-\boldsymbol{x},\Delta x) > \varepsilon$. This is a standard neighbor searching problem, for which a variety of methods exists (e.g., [Arya et al. 1998; Li et al. 2012]). However, for efficiency we exploit the local tree structure.

We first gather the trilinear interpolation weights corresponding to each active sample on every layer of the octree at $\boldsymbol{p}$. Since a graded octree guarantees that all the nearby sample points lie in levels L and $L+1$ (supposing that the level transition is smooth enough; if a sharp transition is present such that a cell contains both larger and smaller cell neighbors, one may also include the level $L-1$), where L denotes a layer level, we can avoid looping over all the layers in practice.

In most regions, the above sample collection scheme finds enough points, except for the dark blue region in the Figure. When $\boldsymbol{p}$ falls within such a region, we first find the nearest active sample from $\boldsymbol{p}$, and additionally incorporate all the "one-ring" neighbors of that sample; these are (implicitly) assigned a weight of ε by 35.

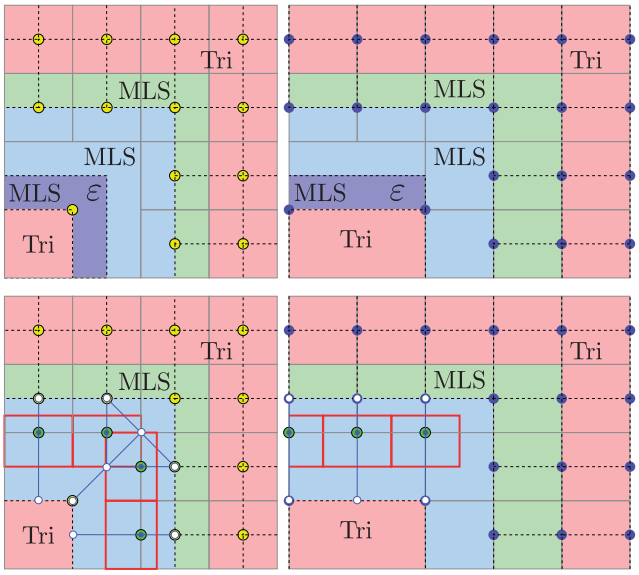

Fig. 9. Top row: Our partially discontinuous MLS interpolation scheme, applied to cell-centered data (left) and face-centered data (right). Trilinear interpolation is performed in the red region, while MLS interpolation is performed in the green, blue, and dark blue regions. Dark blue indicates the region where ε has an effect; the value across its boundary is not necessarily continuous. Bottom row: A fully continuous MLS interpolation. The discontinuity above is resolved by inserting ghost samples, shown as green circles. The red rectangular boxes illustrate the boundaries of the anisotropic trilinear shape functions, shown with *half* of their true widths and heights for the sake of visual clarity. The blue lines and white circles represent the interpolation points for constructing those ghost samples. In actual visual simulations, the former discontinuous interpolation suffices.

6 OCTREE SIZING FUNCTION AND CONSTRUCTION

A vital component of a spatially adaptive simulator is the design of a scalar sizing function $S:\mathbb{R}^3\rightarrow\mathbb{R}$ to determine the required level of local adaptivity. Much of the previous work primarily employed the distance to some geometry of interest (e.g., liquid surfaces or solid obstacles) [Aanjaneya et al. 2017; Chentanez et al. 2007; Ferstl et al. 2014; Losasso et al. 2004]. Such distance-based criteria may suffice when surfaces are uniformly resolved with the highest resolution cells; however, surface adaptivity requires us to distinguish the relative importance of different regions of the surface itself.

To the best of our knowledge, in the graphics literature, the only sizing function that adapts to surface detail is that of Ando et al. [2013], which also considers velocity derivatives and curvature metrics. Our contribution is to improve these metrics even further with the following four novel extensions:

- A *temporal* adaptivity mechanism for the sizing function (§6.1).
- A surface propagation method to precisely capture sharp surface details during refinement (§6.2).

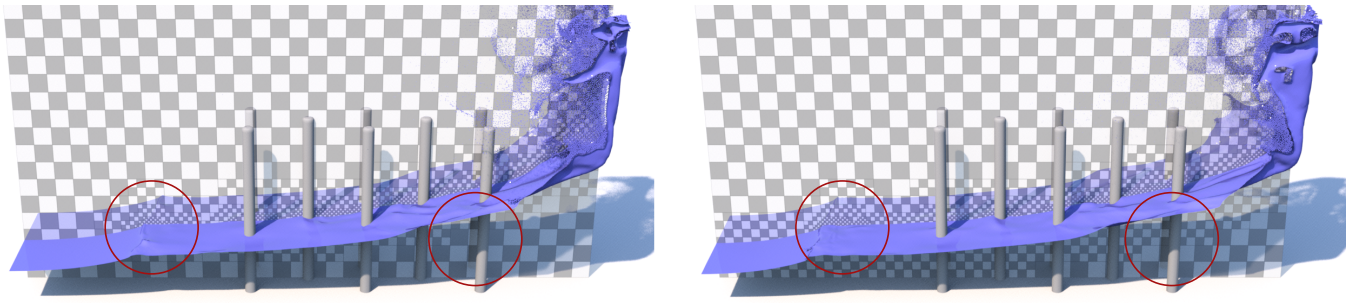

Fig. 10. Comparison of variable sizing strength. $\alpha = 1.0$ (left) and $\alpha = 0.4$ (right). Notice that cells inside the red circles have smaller sizes on the right. More simulations with alpha ranging from 0.0 to 1.0 with 0.2 intervals are provided in <vids/waterdrop_sizing/s00-100.mp4> (e.g., $\alpha = 0.4$ corresponds to s40.mp4).

- A progressive refinement scheme to avoid accessing all the high-resolution cells at the initial time step (§6.4).
- An additional user parameter to control the degree of adaptivity and conveniently blend the uniform and adaptive grid structures (§6.5).

6.1 Temporal adaptivity

Sato et al. [2018] assigned a (metaphorical) *heat* variable to FLIP particles in the context of their extended narrow-band FLIP method. The heat dissipates as time progresses and its value is used in blending between the level set and the particle-based surfaces. The motivating principle is that any interesting detail or motion that emerges should not be immediately deleted in the next time step, but is instead likely to be carried with the flow for a period of time. We can apply an analogous idea to the design of our sizing function. Unlike Sato et al. [2018] though, we will not depend on FLIP particles to carry heat, because we do not seed particles on coarse cells (Section 7.2).

First, assume that we already have sizing values $\{S_{t-\Delta t}\}$ stored on the octree from the previous time step. We advect the sizing values in the same manner as the level set, denoting the result by $\{S_t^*\}$. (Because the octree structure at the previous and new steps will typically differ, we perform the semi-Lagrangian trace-back using the old octree's velocity, starting from the desired sizing function sample position on the new octree [Klingner et al. 2006]). Next, we evaluate the proposed sizing values for the next time step as $\{S_t\}$. Finally, our new sizing value is computed for each point as $\max\left(R(\Delta t)\{S_t^*\}, \{S_t\}\right)$, where

$$R(\Delta t) = T_1^{(\Delta t / T_2)}, \tag{36}$$

and T_1 and T_2 are constants that control the rate at which the effect of the old sizing value diminishes. This indicates that a variable q would be reduced to $T_1 q$ after a specified period of time of T_2. We use $T_1 = 0.9$ and $T_2 = 0.01$ seconds in our examples. An example of Figure 14 without this extension is shown in <vids/notemporal_cylinder.mp4>. In addition to better tracking the motion of salient details, our temporal modification also improves temporal coherence in the refinement pattern (i.e., avoids flicker).

6.2 Propagating values outwards

For simplicity, when performing an octree grid remeshing, we adopt a coarse-to-fine strategy. In doing so, we would hope that all the fine details are eventually captured by the refinement process. Unfortunately, if we naively evaluate a sizing function only at the center of each coarse cell to decide if it should be subdivided, we would often fail to refine where needed and thereby overlook fine details. To ensure that all the interesting details are captured in the remeshing, we propose what we call a *propagation method*.

First, we assume that all relevant interesting details are concentrated near liquid surfaces. At this point, we have two octrees: one from the current time step and one for the next time step. Next, we compute sizing values only on the cells adjacent to liquid interfaces on the current step's octree. These sizing values are then propagated inwards/outwards into the liquid/air, similar to velocity extrapolation. One iteration of our propagation rule is given by

$$\{S^{\text{new}}\}_{ijk} = \frac{\sum_m v_m \max(\{S\}_{ijk}, \{S\}_{N_m})}{\sum_m v_m}, \tag{37}$$

where $v_m = \Delta x_m^3$ denotes cell volume and $\{S\}_{N_m}$ denotes the sizing values on cell neighbors which share the same face, similar to a PDE-based level set redistancing algorithm. As such, a cell is updated multiple times during the propagation, which may introduce excessive diffusion. To prevent such diffusion, we perform this update rule up to five times per cell.

6.3 Octree subdivision and smoothing adaptivity

With the above propagated sizing values, we perform subdivision on the next time step's octree by the following procedures:

(1) Locate the center of an octree cell $\boldsymbol{p}$, and denote its cell size as Δx.
(2) Evaluate the distance from the liquid surface, $|\phi|$, using the level set values at that position.
(3) Interpolate the (propagated) sizing value at $\boldsymbol{p}$ as $S(\boldsymbol{p})$.
(4) Subdivide if $|\phi| < \Delta x$ and $S(\boldsymbol{p}) > 1/\Delta x$.
(5) Move on to the subdivided cells and recursively apply the above procedures.

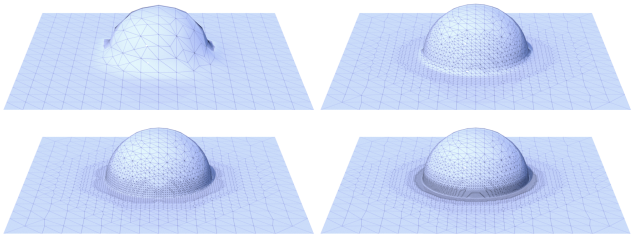

Fig. 11. Our progressive remeshing, applied at the initial time step, correctly captures the high curvature at the edge of the hemisphere. Iterations proceed from left to right and top to bottom. Our top-down iterative remeshing avoids looping over all the finest level cells only to find where the octree should be subdivided.

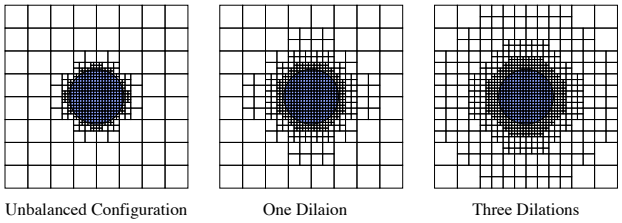

Fig. 12. Adaptivity smoothing. Left: An unbalanced octree configuration. Middle: Our "adaptivity smoothing" applied with one pass of dilation. Right: Two dilation passes. Our adaptivity smoothing operation not only eliminates sharp transitions but also smooths out the transitions, in a user-controllable fashion.

The above algorithm can be naively parallelized for each cell. It should be noted that this subdivision strategy does not yet guarantee that the tree is graded; i.e., that each cell differs from its neighbor by at most one level. Allowing more than one level of change would increase the algorithmic difficulty and potentially introduce unexpected visual artifacts (e.g., grid-aligned reflections due to steep resolution changes [Söderström et al. 2010]). Explicit octree balancing [Isaac et al. 2012] could be used to eliminate such sharp level transitions, but this entails additional implementation effort and computational cost. Other tree-balancing methods have been based specifically on the distance to the surface, rather than the tree structure alone (e.g., [Cecil et al. 2006; Strain 1999]). However, such approaches are incompatible with our aim of surface-adaptivity.

Instead, we propose "adaptivity smoothing", as follows:

(1) Starting from the highest resolution (finest) level L, dilate active cells 3 times.
(2) Upsample the locations of cells on the coarse level $L+1$, and mark (coarse) cells as active at these locations.
(3) If a coarse cell contains at least one fine cell but not all eight cells, mark the eight child cells as active.
(4) Move to one level coarser level. Recursively apply the same procedures for the remaining levels.

The above simple refinement method not only removes very sharp transitions, but also naturally smooths out the spatial rate of refinement as illustrated in Figure 12. The dilation count may be increased to control the transition smoothness.

6.4 Progressive refinement

In the above algorithms, we assumed that we have an auxiliary octree from the previous step, which we use to perform value propagation. Unfortunately, neither an octree nor prior values are available at the initial time step, so the aforementioned algorithm is not applicable. Accessing every cell at the highest resolution is often not a feasible solution for very large-scale scenarios.

We solve this problem by an iterative remeshing approach. We start from a comparatively low resolution octree, and assign sizing values on it. We then use the above algorithms to propagate values and subdivide the octree. This iteration is repeated until sufficient subdivision is achieved. In our examples, three iterations was sufficient. An example of our initial progressive remeshing is shown in Figure 11.

We note that our progressive refinement may still overlook sharp sizing values since the coarse grid may not be resolved enough to capture the value (this will not be the case after the first step except for analytically represented geometries, such as sharp solids or ballistic FLIP particles) . When locations of such sharp sizing values are known explicitly, we directly process the sizing field in a local way to capture it (e.g., this is similar to how Boyd and Bridson [2012] (Figure 7) post-process their level set field to capture sub-grid particles).

6.5 Controlling adaptivity strength

Oftentimes, we wish to control the strength of adaptivity, such as blending uniform grids and a sharp adaptive pattern, because uniform grids provide superior quality in animation (albeit at significantly greater computational cost). Including a new blending parameter α into a sizing function would fail since our subdivision algorithm also depends on the distance to the liquid surface. Instead of modifying the sizing function, we achieve this by introducing a pseudo cell size Δx^* as

$$\Delta x^* = 2^{\ln(\Delta x/\Delta x_0)/\ln(1+\alpha)}\Delta x_0, \tag{38}$$

where Δx_0 denotes a cell size at the highest resolution layer, and $0 < \alpha < 1$ denotes a blending coefficient. Setting $\alpha = 0$ indicates a uniform grid and $\alpha = 1$ dictates strong adaptivity. Note that substituting $\alpha = 0$ results in division by zero, and fully uniform grids should directly be used in this case. The pseudo cell size Δx^* is then used instead of Δx to check if a subdivision criterion is met. A comparison with different α is shown in Figure 10 and in the supplemental videos ($\alpha = 1.0$: <vids/s100.mp4>, $\alpha = 0.4$:<vids/s40.mp4>). We leave the choice of α up to a practitioner's discretion since it provides a balance between the desired visual quality and the availability of computational time/resources.

6.6 Sizing function

Up to this point we have not stated a specific sizing function, as we believe any reasonable generic sizing function would be compatible with the extensions described above. However, the specific sizing

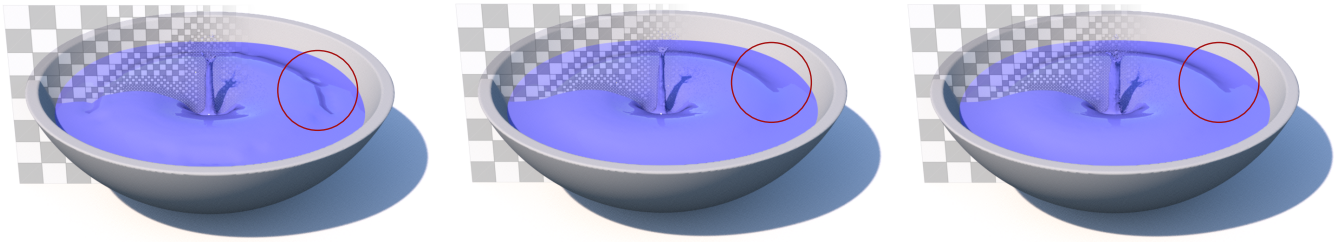

Fig. 13. Comparison of different boundary conditions on free surfaces. Left: A naive first order accurate method. Center: Our SPD method of (25) . Right: Our truly second-order accurate method of (24). Notice that the naive first order accurate method results in a clear artifact in the region marked with a red circle while the SPD method in the middle produces indistinguishable results from the second order solution on the right. Note that we used a *fixed* sizing function field to keep the horizontal position of T-junctions in place on the free surfaces, to eliminate the chance that dynamically changing surface adaptivity could hide such artifacts.

function we use is

$$S(\boldsymbol{p}) = \gamma_\phi \left|\nabla \cdot \nabla \left(\phi + \phi_{\text{solid}}\right)\right| + \gamma_u \sqrt{\sum_i \left(\frac{\partial u_i}{\partial x_i}\right)^2}, \tag{39}$$

where ϕ_{solid} denotes a solid level set. The coefficients γ_ϕ and γ_u denote weights, which we set to $\gamma_\phi = 4$ and $\gamma_u = 3$ except in Figure 1 for which we use $\gamma_u = 0.032$ because the Laplacian term of (39) is independent of spatial scale, but the velocity is dependent on the expected maximal velocity. Note that $S(\boldsymbol{p})$ has units of $1/m$ (as implied by Section 6.3, item (4)) so γ_ϕ is unitless, and γ_u has units of s/m.

We chose the velocity term in (39) because it conveniently responds to any non-translation motion and is easy to evaluate on staggered grids (by contrast, the Jacobian of velocity is somewhat complicated to evaluate on staggered grids). Although (39) technically also responds to rigid rotations, this can be regarded as a source of vorticity, which is also an important feature. Note that we only evaluate the sizing function near liquid surfaces, and propagate it inwards/outwards as in Section 6.2.

7 IMPLEMENTATION

7.1 Surface mesh extraction and extrapolation

We apply (a cell-centered variant of) Dual Contouring [Ju et al. 2002] on the cell-centered signed distance (level set) field to extract liquid surfaces. For velocity extrapolation, we use a simple iterated averaging technique. Since we coarsen grid resolution rapidly away from the surface on the air side, extrapolation can be quickly performed over the entire domain. Similarly, we redistance the level set throughout the entire domain (i.e., not just near the surfaces) using a variant of serial Fast Marching [Sethian 1996] on the set of neighboring cells.

7.2 Adding extended narrow band FLIP

We incorporated extended narrow-band FLIP (EXNBFLIP) [Sato et al. 2018] into our pipeline because it enriches our results both in advection (lessened dissipation near liquid surfaces) and visual expression (particulate splashes) without any special modification to the original EXNBFLIP. EXNBFLIP is used only on the highest resolution layer of the octree, both for simplicity and because our sizing function already suggests that these are the regions with the greatest need for fine details (although EXNBFLIP still uses its own internal seeding strategy). This extension to EXNBFLIP is optional, and an example without it is shown in <vids/levelset.mp4>.

8 RESULTS

Table 2. Timing breakdowns of our examples.

Timings (Seconds)				
Description	Fig. 1	Fig. 2	Fig. 14	Fig. 15
Time per step	21.96	9.27	20.16	16.12
Time per video frame	171.01	96.26	140.42	98.01
Level set advection	2.85	0.90	2.17	1.58
Velocity advection	5.20	1.97	3.12	2.30
Extrapolation	1.06	0.51	1.51	1.24
Projection	1.21	0.35	1.63	0.95
Octree construction	1.14	0.62	1.29	0.96
Sizing value evaluation	4.22	1.90	2.46	1.94
Mesh generation	0.81	0.45	1.27	1.21
EXNBFLIP operations	4.74	N/A	6.38	5.95
Numbers / Ratio / Measured Video Frames				
FLIP particles	632K	N/A	1288K	1167K
Total cells	716K	201K	1319K	1042K
Active fluid cells	332K	90K	647K	460K
Degenerate T-junctions	33.43%	11.13%	19.59%	28.43%
Corresponding video frames	216	0	194	353

We ran our simulations on an Intel(R) Core(TM) i9-9900K (Linux) except for Figure 2 which was run on an Intel(R) Core(TM) i7-6900K CPU 3.20GHz (Linux). Timing breakdowns are summarized in Table 2. Since the timings significantly fluctuate depending on the complexity of the scene, we chose representative challenging moments and averaged 100 time steps (10 frames for video) around those moments. The mesh generation is not included in "Time per step" since it is performed only when exporting a video frame. The corresponding video frame number is shown in the last row of the table. Unless noted, all of our simulations were run with our (mildly

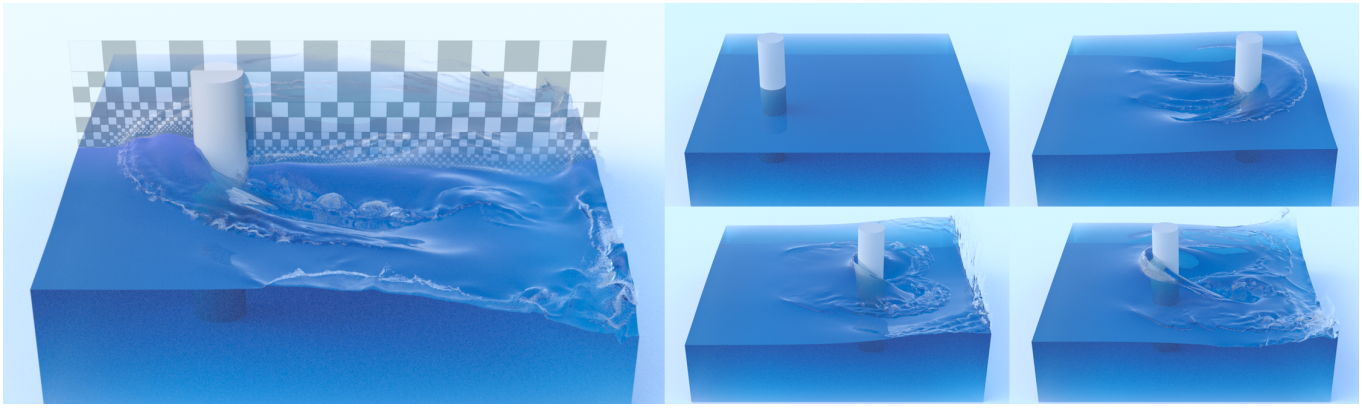

Fig. 14. A moving cylinder stirs a tank. Effective resolution: $512 \times 256 \times 512$. Compute time: 2.34 minutes per video frame.

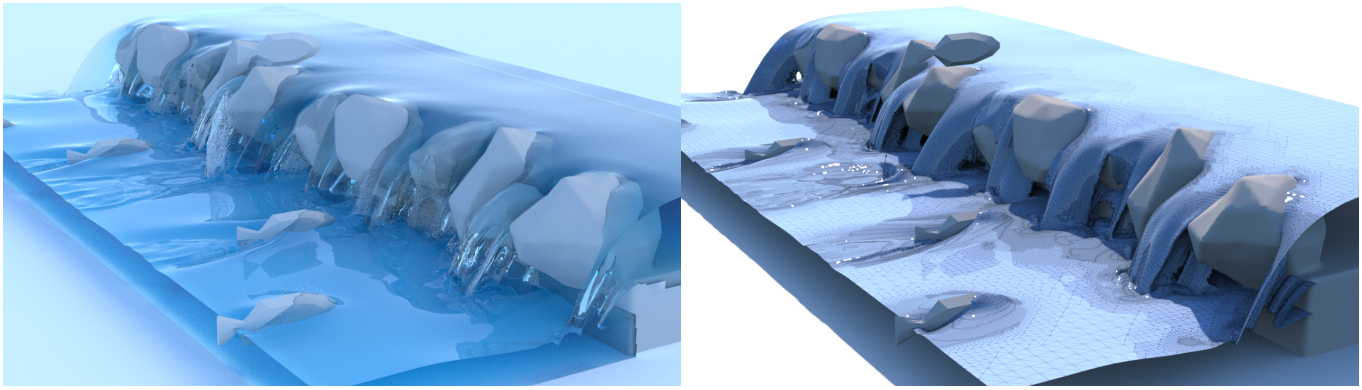

Fig. 15. Swimming fish jumping over a shallow rocky waterfall. Effective resolution: $320 \times 160 \times 640$. Compute time: 1.64 minutes per video frame.

discontinuous) MLS interpolation and the conditionally second-order accurate (i.e., SPD) boundary conditions on pressure at free surfaces using (25). A range of experiments demonstrating the numerical convergence orders of our Laplace discretizations, including second order for the non-symmetric scheme, are provided in the supplemental material.

Before reviewing our main results, we highlight the fact that our (technically first order accurate) pressure boundary conditions of (25) do not introduce *any* noticeable visual differences from the truly second-order accurate (non-symmetric) solutions on any of our tests. We found this a surprising result, given that for many of examples a significant fraction of the T-junctions exhibited the "degenerate" configurations that require clamping, as shown in the second row from the bottom in Table 2.

We emphasize that a naive first order pressure treatment (i.e., setting $p = 0$ without a proper gradient scaling) near T-junctions, even while using proper ghost fluid on regular cells, clearly manifests objectionable artifacts as shown in Figure 13. As such, our novel free surface treatment does effectively eliminate these artifacts despite technically being only first order accurate in grid refinement tests. Figure 13 also used a *fixed* adaptivity pattern to keep T-junctions in the same (horizontal) locations, eliminating the possibility that dynamic octree adaptivity might act to hide such artifacts. We do, however, observe error on an extreme numerical experiment (Figure 16). A more rigorous analysis of this finding is left for future work.

For all simulations including uniform grids, we used the Algebraic Multigrid code of Demidov [2019] to precondition the BiCGSTAB and CG solvers, with a relative residual tolerance of 10^{-4}. We also used a variant of global/regional volume correction schemes (e.g.,[Kim et al. 2007; Thürey et al. 2010]) to compensate long-term accumulated volume loss/gain of liquid.

8.1 Moving cylinder

Figure 14 demonstrates the ability of our method to effectively handle general liquid simulation scenarios. Notice that our automatic sizing function and octree construction capture the detailed motion without creating artifacts or seams near level transitions. The simulation had a resolution of $512 \times 256 \times 512$, and took 2.34 minutes per video frame. To give a rough sense of the expected storage and time savings on this scene, we have conducted two simulations: one with our method and another with the uniform grid method <vids/cylinder_mac.mp4> [Bridson 2015]. Both were run on an Intel(R) Core(TM) i9-9980XE CPU 3.00GHz (Linux). The

average liquid cell count for the uniform method was 33.3 million, while our method was 0.29 million, i.e., 114.8× fewer cells, thus requiring significantly less memory. Regarding the simulation time, the uniform method took 26 minutes per video frame while ours took 63.77 seconds. This is suggestive of a 24.4× speed-up.

We also ran the same setup using the truly second-order accurate method (24) with a BiCGSTAB solver, and found that the average time for the projection was 1.33× slower, indicating that this is also a practical choice. A comparison of the same scene using (24) with a BiCGSTAB solver is shown in <vids/2nd_cylinder.mp4>.

8.2 Fish jumping over a rocky waterfall

Figure 15 demonstrates that our method can provide highly detailed interactions of streaming water through rocks, while the calm bulk regions of the liquid are cheaply simulated with coarse cells. This simulation had a resolution of $320 \times 160 \times 640$ resolutions, and a cost of 1.64 minutes per video frame. Later in the simulation, a small school of fish enters the domain and jumps over the rocks. In those moments, our dynamic sizing function subdivides coarse cells and allows the simulator to capture the ensuing fine, detailed splashes.

8.3 Merging bunnies

Figure 2 shows two liquid bunny shapes deforming due to surface tension under zero gravity and eventually merging. The simulation resolution was $256 \times 256 \times 256$, with a compute time of 1.27 minutes per video frame. Notice that our dynamic sizing function indicates the relative importance of different surface regions, producing varying grid resolution across the bunny. Nonetheless, we did not observe any visual artifacts near grid level transition thanks to our novel treatment of surface tension for T-junctions. Unfortunately, for this single example our method did not outperform the uniform grid method (for example, at the first time step the MAC method took 5 seconds while our method took 9.27 seconds), because the overhead added to our algorithms (e.g., sizing function evaluation, MLS interpolation, octree construction) exceeds the performance benefit of using our adaptive strategy.

One may be tempted to increase the grid resolution; however, explicit surface tension forces come with a severe stability restriction (e.g., Δt must be $O(\Delta x^{3/2})$ [Sussman and Ohta 2009]), and naively increasing grid resolution drastically slows down the simulation. One possible direction to circumvent this issue would be to develop an implicit approximation for surface tension [Zheng et al. 2006] or an approach based on volume-preserving mean curvature [Sussman and Ohta 2009], which we leave for future work.

8.4 Seaplane taking off from a lake

Finally, Figure 1 highlights an ideal case for our method with a very large-scale scenario where the standard uniform grid method would fail. In this example, a seaplane (wingspan: 11m; height: 4.3m) takes off from a lake (30m depth) at high speed, leaving complicated splashes and waves in its wake. Such visually impactful features are also impossible to fully express with a pure level set method at this resolution, thereby also illustrating the benefit of coupling our method with EXNBFLIP. This scene has an effective resolution of $1024 \times 1024 \times 512$, simulated at 2.85 minutes per video frame.

A few apparently axis-aligned splash structures may be noted in Figure 1 (bottom). We attribute this to the horizontally fast-moving liquid, in combination with the usual (grid-dependent) manner in which level sets unintentionally lose thinning sheets. Nevertheless, we did not perceive the underlying motion to be objectionable in the corresponding video sequence, nor did we observe grid-dependency in our other simulations.

To approximately evaluate the speed-up achieved, we simulated the same scenario using the uniform method but with the water depth reduced to 2.3m <vids/seaplane_mac.mp4>. The average time per video frame for the first 3 simulation seconds was 38.46 minutes (suggestive of a 13.49× speed-up).

9 DISCUSSION AND LIMITATIONS

9.1 Discussion

9.1.1 Sparse grid and linear system solve. For of ease of implementation and maintainability, our octree used a naive tile-based sparse grid with each tile having a resolution of 16^3; however, our method could potentially be sped up even further by switching our sparse grid engine to the heavily optimized OpenVDB [Museth 2013] or SPGrid [Setaluri et al. 2014]. We used an Algebraic Multigrid method [Demidov 2019], but we expect that a dedicated geometric multigrid solver [Aanjaneya et al. 2017; Setaluri et al. 2014] and fast Laplacian streaming without explicit topology encoding [Setaluri et al. 2014] could provide further speed-ups.

9.1.2 MLS interpolation. We designed a new MLS-based interpolation which connects in a nearly continuous way with trilinear interpolation on regular cells. While in general MLS interpolants can suffer from instability when too few samples are used, since the set of possible (local) octree configurations is highly restricted, we did not encounter such situations. Our use of MLS interpolation could be naturally extended to other simulation contexts (e.g., overset grids [English et al. 2013]) without loss of generality because it handles interpolation using points and trilinear kernels alone, in grid-oblivious fashion. However, specific to octree grids, the improved node-based interpolation proposed by Setaluri et al. [2014] may be a practical alternative to our method, although it involves more explicit dependence on the local octree structure.

Some examples in Figure 17 exhibit kinks because our interpolation is C^0, not C^1; this is consistent with standard (multi-)linear interpolation. Increasing the MLS polynomial degree could increase accuracy and smoothness (e.g., Guittet et al. [2015] considered a quadratic basis on octrees), although this comes with potential for instabilities (overshooting) and significantly increased costs for neighbor finding and matrix inversion.

9.1.3 Choice of pressure discretization at free surfaces. In the above examples, we mainly chose the SPD form (25) because of CG's efficiency; however, as also mentioned in Section 8.1, the non-symmetric system of (24) also works without numerical difficulties. Hence, the choice of (24) or (25) is left up to the practitioner's preference.

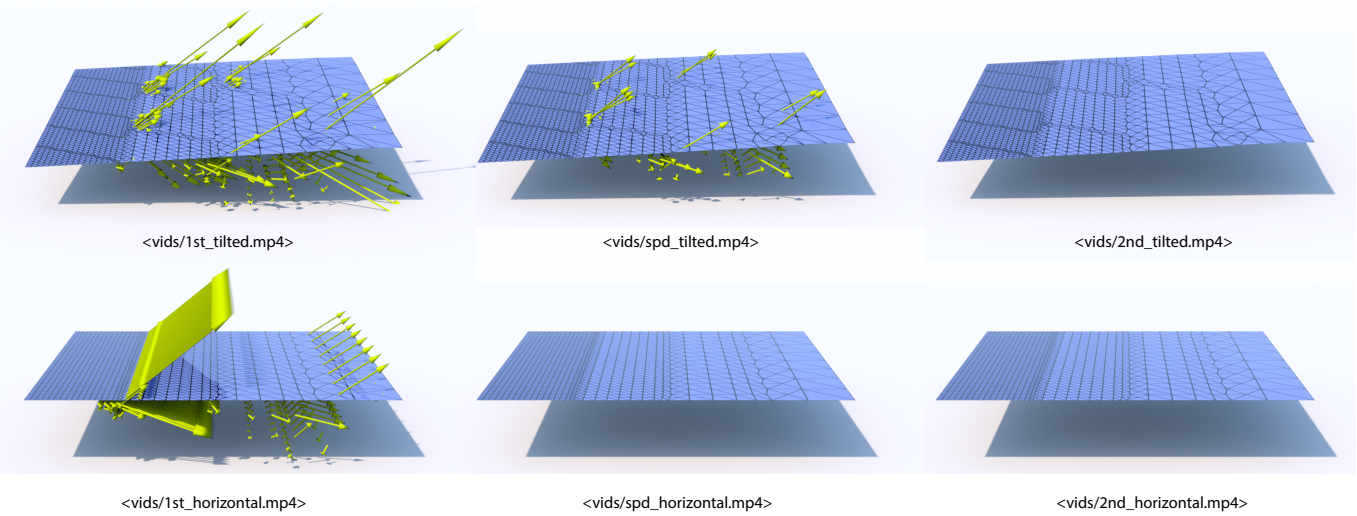

Fig. 16. Error visualization of free surface boundary conditions (as in Figure 11 in the work of Batty et al. [2010]). A strong uniform gravity is added in surface's normal direction. Yellow arrows visualize the error of the pressure gradient. Left: A naive first-order accurate method. Middle: Our method with an SPD system using (25). Right: Our truly second order accurate method (24). The first order accurate method produces error both on tilted (top) and horizontal (bottom) configurations. Our method with (25) also produces (smaller) error on the top but no error on the bottom. Our method with (24) is error-free in both cases.

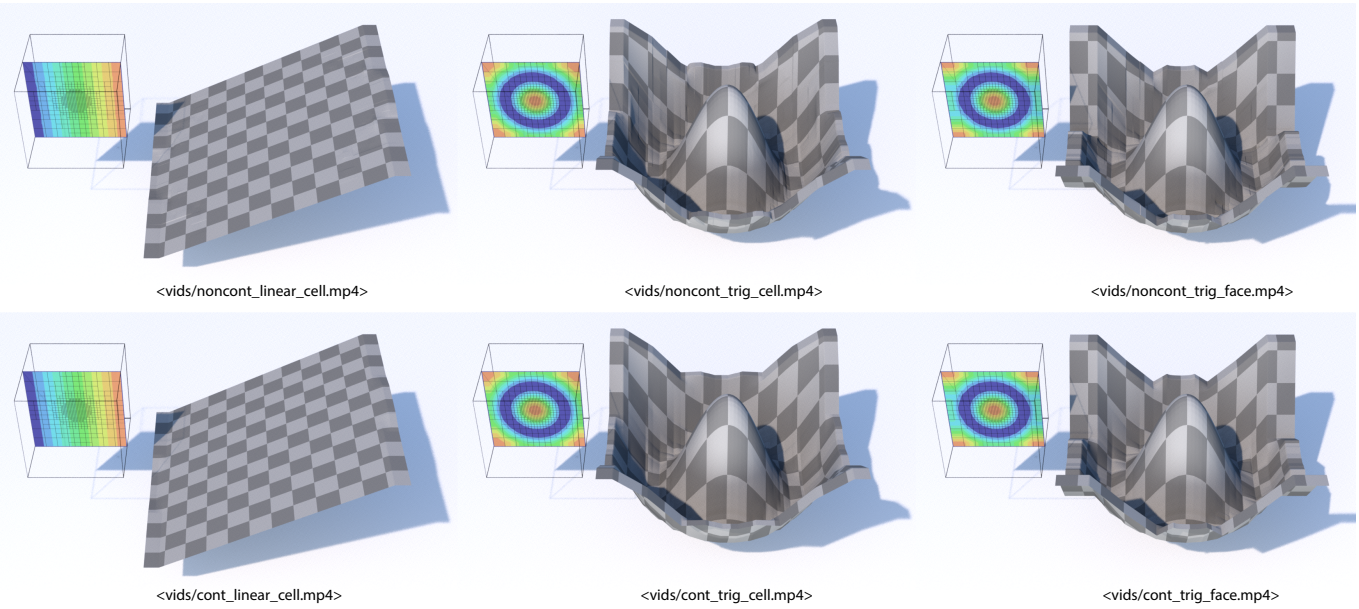

Fig. 17. Error visualization of our MLS interpolation schemes on different test cases. Top: our mildly discontinuous interpolation. Bottom: Fully C^0 continuous interpolation. Left pair: Cell-centered interpolation of a linear mode. Middle pair: Cell-centered interpolation of a trigonometric function. Right pair: Face-centered interpolation of a trigonometric function. Notice that both interpolation exactly schemes recover the linear modes (except for boundaries) while exhibiting some slight discontinuity for trigonometric functions in the top row.

9.2 Limitations

9.2.1 Temporal adaptivity. We were able to significantly speed up the runtime over uniform grids using spatial, but not temporal, adaptivity. Therefore, doubling the effective resolution implies that the time step size must be reduced by half to satisfy a reasonable CFL number. Although semi-Lagrangian advection and FLIP are unconditionally stable, using large CFL numbers introduces excessive diffusion [Lentine et al. 2012]. While the method of Lentine et al. enables simulation with large steps, it may still overlook some visually interesting motions that would otherwise develop during

large time steps. Incorporating temporal adaptivity (e.g., [Fang et al. 2018; Reinhardt et al. 2017]) is a promising research direction.

9.2.2 Compromised accuracy. For many practical scenes (but not all) our method outperforms its uniform grid counterpart, though this comes at the cost of somewhat damped visual details. As has previously been observed, we find that a uniform grid tends to better retain kinetic energy during simulations (e.g., motion remains lively for longer durations) while our method damps out motion noticeably faster (e.g., see an example using the uniform MAC method <vids/mac.mp4>). There are primarily two sources of this damping: a lack of resolution and excessive diffusion on coarse cells. A lack of resolution means that the pressure solver cannot capture high-frequency details, such as small localized regions of vorticity. Excessive diffusion means that the local CFL number may be too small for the coarse cells because the time step is determined by the highest resolution cells. Therefore the cell's velocity diffuses rapidly over a short time period due to semi-Lagrangian advection. The latter issue could be alleviated by the use of a bi-directional advection scheme [Qu et al. 2019] or high-order schemes [Heo and Ko 2010; Narain et al. 2019].

We observed second order accuracy on free surfaces, but we could not obtain empirical second order accuracy on Neumann boundaries (solids) crossing T-junctions, as detailed in the supplemental material. Although the accuracy of pressure on solid faces does not play a major role in our examples, this inaccuracy might cause artifacts in some scenarios (e.g., a liquid bead sliding down a slope).

9.2.3 Extended narrow band FLIP. We believe that EXNBFLIP is an effective extension to our pipeline, but we do observe some surface noise arising from the particle-based surface construction on calm surfaces. This noise eventually dissipates as the heat variable assigned on FLIP particles cools down; this noticeable artifact could be improved by designing another sizing function to prevent seeding FLIP particles on calm regions or performing surface smoothing as a post-process (which we did not).

9.2.4 Conservation. Since semi-Lagrangian advection is not conservative [Lentine et al. 2011], our method does not exactly preserve momentum/mass during advection. This may result in noticeable artifacts such as drifting center of mass, e.g., Figure 2. A momentum/mass conserving advection scheme [Chentanez and Müller 2012; Lentine et al. 2011] might help with this.

9.2.5 Surface adaptivity. Lastly, our method works particularly well for scenes where both calm and vibrant motions are present. When the entire liquid surface is chaotic (e.g., stormy ocean waves) surface adaptivity may not be beneficial because the sizing function would simply set all cells to the highest resolution. For such scenarios, adding subgrid models such as FFT-based waves as a post-process [English et al. 2013] might be effective.

10 CONCLUSIONS

We have proposed a set of new methods for octree-based liquid simulation with a strong emphasis on practical cost-effectiveness. Each of our core algorithms (accurate free surface octree Laplacian, MLS interpolation strategy, and sizing function computation) are designed with ease of implementation in mind without sacrificing computational efficiencies or visual quality. The effectiveness of these components is numerically verified through a range of tests and practical scenes. As a result, our method offers remarkably detailed liquid simulations with surface adaptivity, at modest computational cost, while being more accessible to many practitioners.

ACKNOWLEDGMENTS

This research was supported by the JSPS Grant-in-Aid for Young Scientists (18K18060) and the Natural Sciences and Engineering Research Council of Canada (Grant RGPIN-04360-2014).

REFERENCES

Mridul Aanjaneya, Ming Gao, Haixiang Liu, Christopher Batty, and Eftychios Sifakis. 2017. Power Diagrams and Sparse Paged Grids for High Resolution Adaptive Liquids. *ACM Trans. Graph.* 36, 4, Article 140 (2017), 12 pages. https://doi.org/10.1145/3072959.3073625

Ryoichi Ando, Nils Thürey, and Chris Wojtan. 2013. Highly Adaptive Liquid Simulations on Tetrahedral Meshes. *ACM Trans. Graph.* 32, 4, Article 103 (2013), 10 pages. https://doi.org/10.1145/2461912.2461982

Ryoichi Ando, Nils Thürey, and Chris Wojtan. 2015. A Dimension-Reduced Pressure Solver for Liquid Simulations. *Comput. Graph. Forum* 34, 2 (2015), 473–480. https://doi.org/10.1111/cgf.12576

Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. 1998. An Optimal Algorithm for Approximate Nearest Neighbor Searching Fixed Dimensions. *J. ACM* 45, 6 (1998), 891–923. https://doi.org/10.1145/293347.293348

Vinicius C Azevedo, Christopher Batty, and Manuel M Oliveira. 2016. Preserving geometry and topology for fluid flows with thin obstacles and narrow gaps. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–12.

Vinicius C Azevedo and Manuel M Oliveira. 2013. Efficient smoke simulation on curvilinear grids. In *Computer Graphics Forum*, Vol. 32. 235–244.

Christopher Batty, Florence Bertails, and Robert Bridson. 2007. A fast variational framework for accurate solid-fluid coupling. In *ACM Transactions on Graphics (TOG)*, Vol. 26. ACM, 100.

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

Marsha J Berger and Joseph Oliger. 1984. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of computational Physics* 53, 3 (1984), 484–512.

Morten Bojsen-Hansen and Chris Wojtan. 2013. Liquid surface tracking with error compensation. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 68.

Landon Boyd and Robert Bridson. 2012. MultiFLIP for Energetic Two-Phase Fluid Simulation. *ACM Trans. Graph.* 31, 2, Article 16 (2012), 12 pages. https://doi.org/10.1145/2159516.2159522

Robert Bridson. 2015. *Fluid simulation for computer graphics.* AK Peters/CRC Press.

Tyson Brochu, Christopher Batty, and Robert Bridson. 2010. Matching fluid simulation elements to surface geometry and topology. *ACM Trans. Graph.* 29, 4 (2010), 1–9. https://doi.org/10.1145/1778765.1778784

Thomas C. Cecil, Stanley J. Osher, and Jianliang Qian. 2006. Simplex free adaptive tree fast sweeping and evolution methods for solving level set equations in arbitrary dimension. *J. Comput. Phys.* 213, 2 (2006), 458 – 473. https://doi.org/10.1016/j.jcp.2005.08.020

Nuttapong Chentanez, Bryan E. Feldman, François Labelle, James F. O'Brien, James F. O'Brien, and Jonathan R. Shewchuk. 2007. Liquid Simulation on Lattice-based Tetrahedral Meshes. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '07)*. 219–228. http://dl.acm.org/citation.cfm?id=1272690.1272720

Nuttapong Chentanez and Matthias Müller. 2011. Real-time Eulerian water simulation using a restricted tall cell grid. In *ACM Transactions on Graphics (TOG)*, Vol. 30. ACM, 82.

Nuttapong Chentanez and Matthias Müller. 2012. Mass-Conserving Eulerian Liquid Simulation. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '12)*. 245–254.

Alexandre Joel Chorin. 1968. Numerical Solution of the Navier-Stokes Equations. *Math. Comp.* 22, 104 (1968), 745–762. http://www.jstor.org/stable/2004575

Pascal Clausen, Martin Wicke, Jonathan R. Shewchuk, and James F. O'Brien. 2013. Simulating Liquids and Solid-Liquid Interactions with Lagrangian Meshes. *ACM Transactions on Graphics* 32, 2 (2013), 17:1–15. https://doi.org/0.1145/2451236.2451243

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. 34, 4, Article 50 (2015), 11 pages. https://doi.org/10.1145/2766901

D. Demidov. 2019. AMGCL: An Efficient, Flexible, and Extensible Algebraic Multigrid Implementation. *Lobachevskii Journal of Mathematics* 40, 5 (2019), 535–546. https://doi.org/10.1134/S1995080219050056

R. Elliot English, Linhai Qiu, Yue Yu, and Ronald Fedkiw. 2013. Chimera Grids for Water Simulation. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '13)*. ACM, 85–94. https://doi.org/10.1145/2485895.2485897

Doug Enright, Duc Nguyen, Frederic Gibou, and Ron Fedkiw. 2003. Using the particle level set method and a second order accurate pressure boundary condition for free surface flows. In *ASME/JSME 2003 4th Joint Fluids Summer Engineering Conference*. American Society of Mechanical Engineers Digital Collection, 337–342.

Yu Fang, Yuanming Hu, Shi-Min Hu, and Chenfanfu Jiang. 2018. A Temporally Adaptive Material Point Method with Regional Time Stepping. *Computer Graphics Forum* 37, 8 (2018), 195–204. https://doi.org/10.1111/cgf.13524 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13524

Bryan E Feldman, James F O'brien, and Bryan M Klingner. 2005. Animating gases with hybrid meshes. In *ACM Transactions on Graphics (TOG)*, Vol. 24. Citeseer, 904–909.

F. Ferstl, R. Westermann, and C. Dick. 2014. Large-Scale Liquid Simulation on Adaptive Hexahedral Grids. *IEEE Transactions on Visualization and Computer Graphics* 20, 10 (2014), 1405–1417. https://doi.org/10.1109/TVCG.2014.2307873

Gene H Golub and Charles F Van Loan. 2012. *Matrix computations*. Vol. 3. JHU press.

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

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

Nambin Heo and Hyeong-Seok Ko. 2010. Detail-Preserving Fully-Eulerian Interface Tracking Framework. In *ACM SIGGRAPH Asia 2010 Papers*. ACM, Article 176, 8 pages. https://doi.org/10.1145/1866158.1866198

Yuanming Hu, Yu Fang, Ziheng Ge, Ziyin Qu, Yixin Zhu, Andre Pradhana, and Chenfanfu Jiang. 2018. A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–14.

Hikaru Ibayashi, Chris Wojtan, Nils Thuerey, Takeo Igarashi, and Ryoichi Ando. 2018. Simulating Liquids on Dynamically Warping Grids. *IEEE transactions on visualization and computer graphics* (2018).

Geoffrey Irving, Eran Guendelman, Frank Losasso, and Ronald Fedkiw. 2006. Efficient simulation of large bodies of water by coupling two and three dimensional techniques. In *ACM Transactions on Graphics (TOG)*, Vol. 25. ACM, 805–811.

T. Isaac, C. Burstedde, and O. Ghattas. 2012. Low-Cost Parallel Algorithms for 2:1 Octree Balance. In *2012 IEEE 26th International Parallel and Distributed Processing Symposium*. 426–437. https://doi.org/10.1109/IPDPS.2012.47

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

Tao Ju, Frank Losasso, Scott Schaefer, and Joe Warren. 2002. Dual Contouring of Hermite Data. *ACM Trans. Graph.* 21, 3 (2002), 339–346. https://doi.org/10.1145/566654.566586

Byungmoon Kim, Yingjie Liu, Ignacio Llamas, Xiangmin Jiao, and Jarek Rossignac. 2007. Simulation of Bubbles in Foam with the Volume Control Method. *ACM Trans. Graph.* 26, 3 (2007), 98–es. https://doi.org/10.1145/1276377.1276500

Bryan M. Klingner, Bryan E. Feldman, Nuttapong Chentanez, and James F. O'Brien. 2006. Fluid Animation with Dynamic Meshes. In *Proceedings of ACM SIGGRAPH 2006*. 820–825. http://graphics.cs.berkeley.edu/papers/Klingner-FAD-2006-08/

Michael Lentine, Mridul Aanjaneya, and Ronald Fedkiw. 2011. Mass and Momentum Conservation for Fluid Simulation. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '11)*. ACM, 91–100. https://doi.org/10.1145/2019406.2019419

Michael Lentine, Matthew Cong, Saket Patkar, and Ronald Fedkiw. 2012. Simulating Free Surface Flow with Very Large Time Steps. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '12)*. 107–116.

Shengren Li, Lance Simons, Jagadeesh Bhaskar Pakaravoor, Fatemeh Abbasinejad, John D. Owens, and Nina Amenta. 2012. KANN on the GPU with Shifted Sorting. In *Proceedings of the Fourth ACM SIGGRAPH / Eurographics Conference on High-Performance Graphics (EGGH-HPG'12)*. 39–47.

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. https://doi.org/10.1016/j.compfluid.2005.01.006

Frank Losasso, Frédéric Gibou, and Ron Fedkiw. 2004. Simulating Water and Smoke with an Octree Data Structure. *ACM Trans. Graph.* 23, 3 (2004), 457–462. https://doi.org/10.1145/1015706.1015745

M. K. Misztal, K. Erleben, A. Bargteil, J. Fursund, B. Bunch Christensen, J. A. Bundefinedrentzen, and R. Bridson. 2012. Multiphase Flow of Immiscible Fluids on Unstructured Moving Meshes. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '12)*. 97–106.

Ken Museth. 2013. VDB: High-Resolution Sparse Volumes with Dynamic Topology. *ACM Trans. Graph.* 32, 3, Article 27 (2013), 22 pages. https://doi.org/10.1145/2487228.2487235

Rahul Narain, Jonas Zehnder, and Bernhard Thomaszewski. 2019. A Second-Order Advection-Reflection Solver. *Proc. ACM Comput. Graph. Interact. Tech.* 2, 2, Article 16 (2019), 14 pages. https://doi.org/10.1145/3340257

Andrew Nealen. 2004. *An as-short-as-possible introduction to the least squares, weighted least squares and moving least squares methods for scattered data approximation and interpolation.* Technical Report. URL: http://www.nealen.com/projects/mls/asapmls.pdf.

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

Michael B. Nielsen and Robert Bridson. 2016. Spatially Adaptive FLIP Fluid Simulations in Bifrost. In *ACM SIGGRAPH 2016 Talks (SIGGRAPH '16)*. ACM, Article 41, 2 pages. https://doi.org/10.1145/2897839.2927399

Stanley Osher, Ronald Fedkiw, and K Piechor. 2004. Level set methods and dynamic implicit surfaces. *Appl. Mech. Rev.* 57, 3 (2004), B15–B15.

Stéphane Popinet. 2003. Gerris: A Tree-Based Adaptive Solver for the Incompressible Euler Equations in Complex Geometries. *J. Comput. Phys.* 190, 2 (2003), 572–600. https://doi.org/10.1016/S0021-9991(03)00298-5

Ziyin Qu, Xinxin Zhang, Ming Gao, Chenfanfu Jiang, and Baoquan Chen. 2019. Efficient and Conservative Fluids Using Bidirectional Mapping. *ACM Trans. Graph.* 38, 4, Article 128 (2019), 12 pages. https://doi.org/10.1145/3306346.3322945

Stefan Reinhardt, Markus Huber, Bernhard Eberhardt, and Daniel Weiskopf. 2017. Fully Asynchronous SPH Simulation. In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation (SCA '17)*. ACM, Article 2, 10 pages. https://doi.org/10.1145/3099564.3099571

Takahiro Sato, Christopher Wojtan, Nils Thuerey, Takeo Igarashi, and Ryoichi Ando. 2018. Extended Narrow Band FLIP for Liquid Simulations. In *Computer Graphics Forum*, Vol. 37. 169–177.

Andrew Selle, Ronald Fedkiw, Byungmoon Kim, Yingjie Liu, and Jarek Rossignac. 2008. An unconditionally stable MacCormack method. *Journal of Scientific Computing* 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.* 33, 6, Article 205 (2014), 12 pages. https://doi.org/10.1145/2661229.2661269

J A Sethian. 1996. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences* 93, 4 (1996), 1591–1595. https://doi.org/10.1073/pnas.93.4.1591 arXiv:https://www.pnas.org/content/93/4/1591.full.pdf

Lin Shi and Yizhou Yu. 2004. Visual smoke simulation with adaptive octree refinement. In *Computer Graphics and Imaging*. 13–19.

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

Andreas Söderström, Matts Karlsson, and Ken Museth. 2010. A PML-Based Nonreflective Boundary for Free Surface Fluid Animation. *ACM Trans. Graph.* 29, 5, Article 136 (2010), 17 pages. https://doi.org/10.1145/1857907.1857912

Fabricio Simeoni de Sousa, CF Lages, Jonas Laerte Ansoni, A Castelo, and A Simao. 2019. A finite difference method with meshless interpolation for incompressible flows in non-graded tree-based grids. *J. Comput. Phys.* 396 (2019), 848–866.

John Strain. 1999. Tree methods for moving interfaces. *J. Comput. Phys.* 151, 2 (1999), 616–648.

Mark Sussman and Mitsuhiro Ohta. 2009. A Stable and Efficient Method for Treating Surface Tension in Incompressible Two-Phase Flow. *SIAM J. Sci. Comput.* 31, 4 (2009), 2447–2471. https://doi.org/10.1137/080732122

Nils Thürey, Chris Wojtan, Markus Gross, and Greg Turk. 2010. A Multiscale Approach to Mesh-Based Surface Tension Flows. *ACM Trans. Graph.* 29, 4, Article 48 (2010), 10 pages. https://doi.org/10.1145/1778765.1778785

X. Zhai, F. Hou, H. Qin, and A. Hao. 2018. Fluid Simulation with Adaptive Staggered Power Particles on GPUs. *IEEE Transactions on Visualization and Computer Graphics* (2018), 1–1. https://doi.org/10.1109/TVCG.2018.2886322

Wen Zheng, Jun-Hai Yong, and Jean-Claude Paul. 2006. Simulation of Bubbles. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '06)*. 325–333.

Bo Zhu, Wenlong Lu, Matthew Cong, Byungmoon Kim, and Ronald Fedkiw. 2013. A New Grid Structure for Domain Extension. *ACM Trans. Graph.* 32, 4, Article 63 (2013), 12 pages. https://doi.org/10.1145/2461912.2461999

Yongning Zhu and Robert Bridson. 2005. Animating sand as a fluid. *ACM Transactions on Graphics (TOG)* 24, 3 (2005), 965–972.