Quadtree Tall Cells for Eulerian Liquid Simulation

FUMIYA NARITA, The University of Tokyo and GAME FREAK Inc., Japan
NIMIKO OCHIAI, GAME FREAK Inc., Japan
TAKASHI KANAI, The University of Tokyo, Japan
RYOICHI ANDO, Unaffiliated, Japan

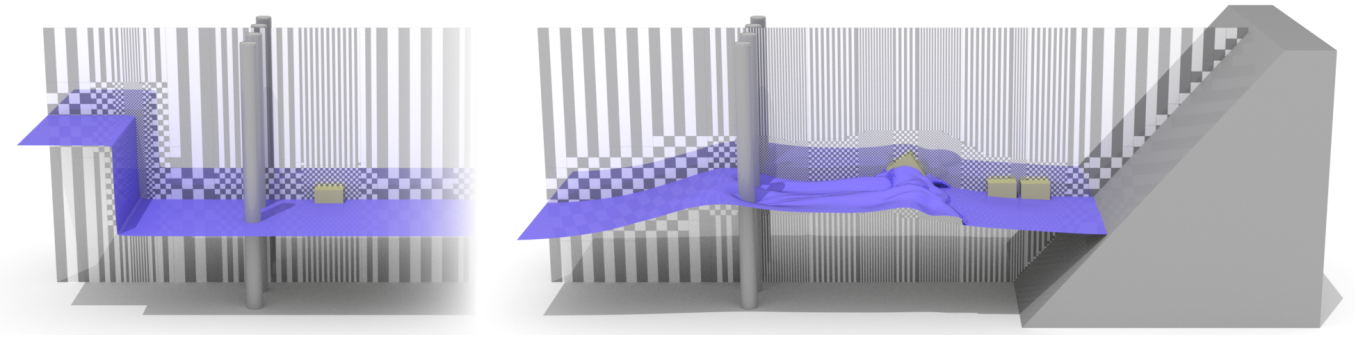


Fig. 1. Breaking wave reaches three cubical solids floating in its path, where the motion is strongly two-way coupled. Grid texture at the center visualize our tall grid cells with horizontal adaptivity applied. Notice that cells are dynamically subdivided to adapt to scene complexity, such as at the tips of breaking waves and in the vicinity of rigid bodies. Also rigid bodies in part intersect with tall cells of different sizes, necessitating our newly devised variational two-way coupling formulation of rigid bodies. The effective resolution is $512 \times 192 \times 192$.

This paper introduces a novel grid structure that extends tall cell methods for efficient deep water simulation. Unlike previous tall cell methods, which are designed to capture all the fine details around liquid surfaces, our approach subdivides tall cells horizontally, allowing for more aggressive adaptivity and a significant reduction in the number of cells. The foundation of our method lies in a new variational formulation of Poisson's equations for pressure solve tailored for tall-cell grids, which naturally handles the transition of variable-sized cells. This variational view not only permits the use of the efficacy-proven conjugate gradient method but also facilitates monolithic two-way coupled rigid bodies. The key distinction between our method and previous general adaptive approaches, such as tetrahedral or octree grids, is the simplification of adaptive grid construction. Our method performs grid subdivision in a quadtree fashion, rather than an octree. These 2D cells are then simply extended vertically to complete the tall cell population. We demonstrate that this novel form of adaptivity, which we refer to as quadtree tall cells, delivers superior performance compared to traditional uniform tall cells.

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

Additional Key Words and Phrases: Fluid, Liquid, Tall Grids

Authors' addresses: Fumiya Narita, The University of Tokyo and GAME FREAK Inc., Tokyo, Japan, fumiya.narita162@gmail.com; Nimiko Ochiai, GAME FREAK Inc., Tokyo, Japan, n.ochaduke@gmail.com; Takashi Kanai, The University of Tokyo, Tokyo, Japan, kanait@acm.org; Ryoichi Ando, Unaffiliated, Tokyo, Japan, ryich.ando@gmail.com.

SIGGRAPH Conference Papers '25, August 10–14, 2025, Vancouver, BC, Canada

ACM ISBN 979-8-4007-1540-2/2025/08
https://doi.org/10.1145/3721238.3730652

ACM Reference Format:
Fumiya Narita, Nimiko Ochiai, Takashi Kanai, and Ryoichi Ando. 2025. Quadtree Tall Cells for Eulerian Liquid Simulation. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers (SIGGRAPH Conference Papers '25), August 10–14, 2025, Vancouver, BC, Canada.* ACM, New York, NY, USA, Article 111, 11 pages. https://doi.org/10.1145/3721238.3730652

1 INTRODUCTION

Large-scale liquid simulation is an important research area in graphics due to its high industry demand and expressive capabilities [Lesser et al. 2022; Nielsen and Bridson 2016]. Historically, the mainstream effort has been directed towards tetrahedral grids [Ando et al. 2013; Batty and Houston 2011; Chentanez et al. 2007; Clausen et al. 2013] and octree grids [Aanjaneya et al. 2017; Ando and Batty 2020; Ferstl et al. 2014; Goldade et al. 2019; Losasso et al. 2004], which can encompass a wide range of fluid phenomena.

Aside from above, tall-cell grids [Chentanez and Müller 2011; Chentanez et al. 2014; Irving et al. 2006] are also introduced as an alternative. Tall-cell grids are more limited compared to above methods, as they require the presence of gravity to accurately simulate the motion of liquid. The advantages of tall-cell grids lie in their simplicity in the structure of grids. Unlike tetrahedral grids or octrees, cells can be fast populated using the height field. This significantly reduces grid generation costs while also providing adaptivity benefits in terms of runtime and memory consumption.

One existing issue with the tall-cell grids is their limited adaptivity in the horizontal directions, which we refer to as the xz-plane. In practice, liquid surfaces can exhibit both localized splashes and calm regions [Ando and Batty 2020], but one is forced to resolve both dynamics using the same grid resolution; resulting in inefficiency.

Resorting to tetrahedral or octree grids remains a promising option; however, it can become overwhelming in strongly horizontally-extended liquid scenes.

This paper aims to address this limitation by leveraging the tall-cell grids. Instead of uniformly distributing the cells on the xz-plane, we choose to subdivide it in a *quadtree* fashion while best inheriting all the merits of tall-cell approaches, such as simplicity and fast grid population.

In doing so, we also reformulate the pressure solver in a variational manner. Previous approaches discretize pressure using finite differences, resulting in either symmetric [Irving et al. 2006] or asymmetric [Chentanez and Müller 2011] matrix views. While such approaches may suffice for their needs, they can exhibit significant complexity when designing adaptive grids like ours, if chosen to do so. Moreover, they can hinder the seamless integration of two-way coupled solid dynamics, as interface treatments between fluids and solids require special care [Batty et al. 2007].

For the remaining components, we adopt the method introduced by Ando and Batty [2020] to ensure second-order accurate boundary conditions for free surfaces crossing T-junctions. We also employ dual contouring [Ju et al. 2002] for surface meshing, both to eliminate potential artifacts at the cell-size transition. In summary, our contributions are as follows

- Horizontally adaptive tall cells for liquid simulation.
- Variational pressure solver for our quadtree tall cells.
- Two-way coupled rigid bodies within our framework.

We demonstrate that our method achieves a significant acceleration in runtime compared to an existing tall-cell approach [Irving et al. 2006] and yet remains simple, offering a more accessible alternative to practitioners than general adaptive grids.

2 PREVIOUS WORK

Tall Cells. In graphics, tall cells are pioneered by Irving et al. [2006] and further extended for real-time applications by Chentanez and Müller [2011]. The former approach allocates a thick layer underneath the water surfaces consisting of uniform grids, which they refer to as the optical layer, to resolve dynamics not only for the surfaces but also for certain depths. Accordingly, the accuracy of the dynamics can be controlled by specifying the thickness of the optical layer, and they suggest setting the thickness to 1/4 of the depth [Irving et al. 2006].

The latter approach extends this concept by developing a fast novel multigrid solver designed for their slightly modified tall-cell grid structures, enabling the entire pipeline to efficiently run on the GPU. To the best of our knowledge, no significant advancements in tall-cell grids have been reported subsequently. Our method makes advancement in this field, as we will elaborate in the following sections. The key difference is that our method employs a variational framework, while both methods use finite differences.

Surface-Only Water Dynamics. Our method is also pertinent to the extensive literature on surface-only fluid dynamics. An example is the shallow water equations, where the motion of fluid is simplified under the assumption that pressure varies linearly with the depth of water [Bridson 2008]. An early work on this subject in graphics is by Layton and Michiel van de Panne [2002], where water waves are semi-implicitly solved to maintain stability. This work has been extended to triangular grids with the explicit symmetrization of matrices [Wang et al. 2007]. Beyond the shallow water equations, researchers have explored methods to reproduce a wider range of fluid motions using the Boundary Element Method (BEM). For instance, Keeler and Bridson [2014] employed the BEM to simulate the motion of vast oceans, assuming that fluid velocity is approximated by a curl-free potential flow. Huang et al. [2021] integrated Fluid-Implicit-Particle (FLIP) and BEM to simulate large-scale ocean scenes. The BEM is also used to simulate the dynamics of soap [Da et al. 2016], ferrofluids [Huang and Michels 2020] and fundamental solutions of waves [Schreck et al. 2019]. Some works adopt similar concepts [Canabal et al. 2016; Loviscach 2002], which involves distributing dispersion kernels at source points to simulate propagating waves.

Enriched Surface Dynamics. Rather than approximating the bulk motions of the Navier-Stokes equations using surface-only variables, many researchers have concentrated on wave dynamics that are visually important but too weak to affect the body of water. Yuksel et al. [2007] introduced the concept of wave particles to simulate propagating waves on liquid surfaces, with Largangian particles carrying a single wavelength. This work has been extended to carry multi-wavelength crests [Jeschke and Wojtan 2017]. Spline elements are also used to track the moving waves [Jeschke and Wojtan 2015; Skrivan et al. 2020; Thürey et al. 2007]. Surface dynamics are also solved using Eulerian approaches rather than Lagrangian elements [Kass and Miller 1990]. Kim et al. [2013] introduced a closest-point method that solves the wave equations without explicit surface parameterization by volumetrically approximating the solution within narrow-banded 3D cells.

Coupled Bulk-Surface Dynamics. Beyond a certain level of vibrancy, dynamics on the surfaces begin to exert a noticeable influence on the motion in the bulk. To achieve this, some works propose coupling both dynamics, with surfaces being solved using a wave solver while the bulk employs the Navier-Stokes equations to leverage the advantages of both approaches [Schreck and Wojtan 2022; Thürey et al. 2006]. These coupled approaches are further extended to surface-dominant cells and bulk particles [Golas et al. 2012], as well as height fields and secondary particles [Chentanez et al. 2014].

General Spatially Adaptive Methods. Numerous adaptive methods for fluid simulation exist in graphics. Losasso et al. [2004] propose the use of an octree data structure to simulate smoke and water. Tetrahedral meshes have been subsequently devised for simulating smoke [Feldman et al. 2005; Klingner et al. 2006] and water [Ando et al. 2013; Batty et al. 2010; Chentanez et al. 2007; Clausen et al. 2013]. In addition to octree and tetrahedral grids, several unique grid structures have been proposed, including staggered grids [Xiao et al. 2020], chimera grids [English et al. 2013], power particles [de Goes et al. 2015; Qu et al. 2022; Zhai et al. 2020], simplicial complexes [Misztal et al. 2010], coarse grid projection [Ando et al. 2015; Lentine et al. 2010], axis-aligned streched cells [Zhu et al. 2013], overlapping grids [Dobashi et al. 2008], curvilinear grids [Azevedo and Oliveira 2013; Ibayashi et al. 2020], and a variable-sized SPH [Adams et al. 2007; Solenthaler and Gross 2011; Winchenbach and Kolb 2021].

In-depth discussions on these methods are beyond the scope of this paper, and we refer readers to the survey by Manteaux et al. [2017].

3 CHALLENGES IN GRID CONSTRUCTION

Before delving into the details of our method, we would like to emphasize the challenges associated with constructing spatially-varying grid cells in prior literature. This also helps to clarify the motivation behind our work.

To partition the domain of interest into computational elements, the dihedral angles of the elements, running time, and the number of elements are some important measurements to evaluate the quality of the resultant grid construction algorithm [Sorgente et al. 2023]. Failure to adhere to these criteria leads to ill-conditioned systems or excessive overhead [Gao et al. 2017]. Various approaches have been proposed to meet these criteria, including variational meshing [Alliez et al. 2005], body-centered cubic (BCC) meshes [Ando et al. 2013; Labelle and Shewchuk 2007; Molino et al. 2003; Wang and Yu 2011], and others [Hu et al. 2020; Si 2015]. Unfortunately, all of the above methods require dedicated engineering for high-quality meshing, which can pose a hefty bottleneck in terms of both runtime and practicality.

Tall cells, on the one hand, reduce the aforementioned challenges to nearly negligible levels since all we need is the height field of water surfaces. Our method employs quadtree subdivision, which, of course, introduces more complexity than uniform tall cells, but it remains by far more affordable than the above volumetric adaptive grid construction methods.

4 METHOD OVERVIEW

4.1 Evaluating Sizing Function

Our method starts with constructing a 2D quadtree grid. This is accomplished by recursively evaluating a sizing function $f(\boldsymbol{x})$ at the center of cells in a coarse-to-fine order and deciding whether the cell must be subdivided into four. In doing this, special care is needed to avoid rapid coarsening. If the ratio of diameters of two neighboring cells exceeds two, our variational discretization, to be presented, may not work as expected due to the numerical reflection arising from the rapid change in the resolvable accuracy [Söderström et al. 2010a]. One promising solution is to adopt tree balancing [Isaac et al. 2012], but it can be overwhelming for our method. In our method, we employ an adaptive smoothing [Ando and Batty 2020].

To efficiently evaluate $f(\boldsymbol{x})$, we employ the following strategy. First, assuming we already have quadtree tall-cells constructed from the previous step or they are explicitly provided, we evaluate the sizing function on these cells. Second, we perform a vertical scan for each column to identify the maximal value. Finally, these maximal values are flattened onto the xz-plane.

Once again, we would like to emphasize the benefits of quadtree subdivision. In general, evaluating $f(\boldsymbol{x})$ can be expensive in 3D coordinates. For example, consider a function $f(\boldsymbol{x})$ defined as the minimum of distance functions from numbers of source points of interest (e.g., $f(\boldsymbol{x}) = \min_i |\boldsymbol{x} - \boldsymbol{y}_i|$ where $\boldsymbol{y}_i$ is a source point) distributed at random positions. If the source points are scattered around the cells with high turbulence, we end up evaluating all the points every time when deciding if a cell should be subdivided.

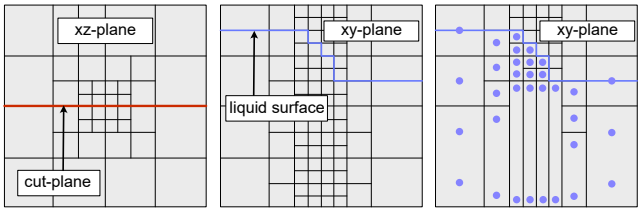


Fig. 2. Quadtree Tall Cell Building Overview: A quadtree subdivided on the xz-plane (left). Vertical cell extension (middle). Tall cell conversion (right) and pressure samples (blue dots).

In 3D, this number can be quite high, making brute-force visiting an impractical solution. Surely, we can make use of approximated nearest neighbor search (a.k.a ANN) [Mount and Arya 1998], but it would require rebuilding the k-d tree at every step and tree traversal for lookup; therefore, it still imposes some notable cost.

As quadtree cells are more manageable than unstructured 3D grids, we can perform a parallel vertical scan to gather values into a 2D image, allowing for quick lookup for subsequent subdivision.

As for the design of the sizing function, we use the variation in velocity and surface curvature as suggested by Ando et al. [2013]. However, this can be customized and is subject to the artist's needs.

4.2 Quadtree Tall-Cell Population

Once a quadtree grid is available, as illustrated in Figure 2 left, we vertically extend it all the way both to the bottom and to the top, as shown in the middle. Next, we concatenate vertical cells that are a few cells away from the surface to form tall cells, both in the downward and upward directions. Finally, we assign pressure variables at the centers for cubical cells and at the top and bottom for tall cells, as illustrated on the right.

For relatively flat water surfaces, the above strategy suffices. However, in cases where water surfaces exhibit complex topological changes such as splashes and bubbles, multiple tall cells may be needed for a single column. In that case, we vertically scan the fluid cells and transform the connected sequence of cells into tall cells based on the steps above.

Note that we are aware of the redundant cell allocation and removals in this process. We choose to employ this strategy both to simplify our grid generation algorithm and for clarification purposes. With some small additional engineering efforts, one should be able to optimize it and eliminate explicit allocations and scans by e.g., utilizing virtual cell allocations.

4.3 Variational Pressure Solver

Our method shares many similarities with the method of Ando and Batty [2020], and as such for brevity we use the same notations for gradient and divergence operators, among others. Our linear system for the pressure variables is given by

$$-[\nabla]^T[V][A][F][\nabla]\{p\} = -[\nabla]^T[V][A]\{\boldsymbol{u}^*\}, \tag{1}$$

where $[\nabla]$ and $[\nabla]^T$ denote discretized gradient and divergence operators, respectively. $[F]$ and $[A]$ are diagonal matrices representing fluid area-volume fraction near the free surfaces and the

flux-area fraction near the solids. Finally, $\{p\}$, $\{\boldsymbol{u}^*\}$ and $[V]$ are vectorized pressure, velocity after the advection and a diagonal matrix encoding face-cell volumes, respectively. Readers are referred to the work of Ando and Batty [2020] for a more detailed definition.

The difference from the method of Ando and Batty [2020] lies in the computation of $[\nabla]$, $[\nabla]^T$, $[F]$, $[A]$, and $[V]$. Thus, in the following, we will detail how to compute these operators and diagonal matrices.

4.3.1 Gradient and Area Volume. For clarity, we will illustrate our algorithm in 2D, but it should be analogously extended to 3D as we will explain. Let p_1 and p_2 be a pair of pressure samples on a tall cell, and p_3 and p_4 be a pair of pressure samples on an adjacent tall cell as illustrated in Figure 3. Let the red rectangle between the tall cells be a volume encapsulating the touching area of the two cells, which corresponds to area volume $[V]$. Similarly, let two yellow circles be a pair of vertically interpolated pressure samples such that

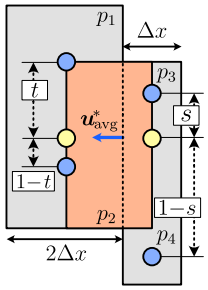


Fig. 3. Pressure Gradient Operator

$$p_{\text{left}} = (1-t)p_1 + tp_2, \tag{2}$$

$$p_{\text{right}} = (1-s)p_3 + sp_4, \tag{3}$$

where t and s denote interpolation coefficients, respectively. Finally, the gradient operator for the horizontal component in this example is given by

$$\frac{\partial p}{\partial x} = (p_{\text{right}} - p_{\text{left}})/\Delta x, \tag{4}$$

where Δx is the distance between p_{right} and p_{left}. For every adjacent pair of tall cells, we embed (4) into $[\nabla]$ to complete the construction of the $[\nabla]$ operator. (Please note that the denominator of (4) become $1.5\Delta x$ in Figure 3.)

The intermediate velocity located at the middle of the boundary of the two adjacent tall cells are formally defined such that

$$\boldsymbol{u}^*_{\text{avg}} = \frac{1}{A_{\text{shared}}} \iint_{S_{\text{shared}}} \boldsymbol{u}^* dS, \tag{5}$$

where S_{shared} and A_{shared} are the area of the boundary shared by two neighboring tall cells as illustrated by the red line in Figure 4. Direct evaluation of (5) would be cumbersome; instead, we use a simple averaging technique on the temporarily collapsed cells, as shown in Figure 4.

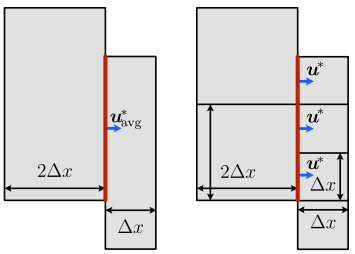


Fig. 4. Tall Cell Collapsing and Being Averaged

When choosing a middle location on the tall-cell face, it should be minded that the gradient direction should be aligned with the normal of the face, as failing to do so can result in first-order accurate spatial derivatives [Losasso et al. 2006], leading to spurious artifacts [Batty et al. 2010]. The divergence operator can be simply constructed by transposing $[\nabla]$. This contrasts with the method of Irving et al. [2006], where the divergence operator needs to be explicitly defined with care such that the resultant linear system becomes symmetric positive definite. Also notice that (4) accommodates not only uniform tall cells but also variable-sized tall cells while retaining this simple format, which is challenging with finite differences.

4.3.2 Boundary Conditions and Fractions. Fluid and area fractions corresponding to $[F]$ and $[A]$ are computed exactly as mentioned in Ando and Batty [2020]. For fluid fractions, one can focus solely on the cells that intersect interfaces, which can be analogously viewed as octree cells. Using this view, one can compute both the fluid fraction and enforce the second-order boundary conditions for free surfaces without making any changes to the original algorithm. Surface extraction is also straightforward with dual contouring [Ju et al. 2002], as no special care associated with the tall cells is needed.

4.4 Two-Way Coupled Rigid Bodies

In order to fully harness the benefits of our variational view, we would like to extend our method to handle two-way coupled rigid bodies. To this end, we aim to define two linear operators, denoted as $[J_{\text{trans}}]$ and $[J_{\text{rot}}]$, that satisfy the following relations

$$-[J_{\text{trans}}]\{p\} = \boldsymbol{f}_{\text{trans}} = \iint_s p\boldsymbol{n} dS, \tag{6}$$

$$-[J_{\text{rot}}]\{p\} = \boldsymbol{\tau}_{\text{rot}} = \iint_s (\boldsymbol{x} - \boldsymbol{X}_{\text{com}}) \times p\boldsymbol{n} dS, \tag{7}$$

where $\boldsymbol{f}_{\text{trans}}$ and $\boldsymbol{\tau}_{\text{rot}}$ are the net forces and torque, respectively, acting on a rigid body [Batty et al. 2007]. $\boldsymbol{X}_{\text{com}}$ and $\boldsymbol{n}$ denote the center of mass of the rigid body and the surface normal, respectively. We follow a similar strategy as outlined in Batty et al. [2007] to transform (6) and (7) into volume integrals and obtain

$$[J_{\text{trans}}]_1^{i,j,k} = \Delta x^2 (A'_{i+\frac{1}{2},j,k} - A'_{i-\frac{1}{2},j,k}), \tag{8}$$

$$[J_{\text{trans}}]_2^{i,j,k} = \Delta x^2 (A'_{i,j+\frac{1}{2},k} - A'_{i,j-\frac{1}{2},k}), \tag{9}$$

$$[J_{\text{trans}}]_3^{i,j,k} = \Delta x^2 (A'_{i,j,k+\frac{1}{2}} - A'_{i,j,k-\frac{1}{2}}), \tag{10}$$

$$\begin{aligned}[J_{\text{rot}}]_1^{i,j,k} &= -\Delta x^2 (z-Z)(A'_{i,j+\frac{1}{2},k} - A'_{i,j-\frac{1}{2},k}) \\ &\quad + \Delta x^2 (y-Y)(A'_{i,j,k+\frac{1}{2}} - A'_{i,j,k-\frac{1}{2}}),\end{aligned} \tag{11}$$

$$\begin{aligned}[J_{\text{rot}}]_2^{i,j,k} &= -\Delta x^2 (x-X)(A'_{i,j,k+\frac{1}{2}} - A'_{i,j,k-\frac{1}{2}}) \\ &\quad + \Delta x^2 (z-Z)(A'_{i+\frac{1}{2},j,k} - A'_{i-\frac{1}{2},j,k}),\end{aligned} \tag{12}$$

$$\begin{aligned}[J_{\text{rot}}]_3^{i,j,k} &= -\Delta x^2 (y-Y)(A'_{i+\frac{1}{2},j,k} - A'_{i-\frac{1}{2},j,k}) \\ &\quad + \Delta x^2 (x-X)(A'_{i,j+\frac{1}{2},k} - A'_{i,j-\frac{1}{2},k}),\end{aligned} \tag{13}$$

where $A' = 1 - A$ and $[X, Y, Z] = \boldsymbol{X}_{\text{com}}$. (x, y, z) is the center position of a cell (i, j, k). $[J]_n^{i,j,k}$ denotes a matrix entry of $[J]$ at n-th row and (i, j, k)-th column with respect to the cell (i, j, k). (i, j, k)-th column is usually interpreted via a topological encoder e.g., $m = N_x N_z k + N_x j + i$ where (N_x, N_y, N_z) is a grid resolution.

Algorithm 1: Our Simulation Loop

```
1  Function Advance_Step():
2      Save Current Grid and Variables
3      Evaluate Sizing Function                    // Sec. 4.1
4      Subdivide Quadtree
5      Extend Cells Vertically
6      Advect Velocity and Level Set               // Sec. 4.5
7      Advance Rigid Body
8      Convert Cells into Tall Cells               // Sec. 4.2
9      Solve Pressure from (1) or (14) // Sec. 4.3, Sec. 4.4
10     Map Pressure onto Cubical Cells
11     Update Velocity and Rigid Body
```

$A'_{i+\frac{1}{2},j,k}$ is the area fraction subtracted from one $1-A$ of two adjacent cells, (i, j, k) and the cell to its right.

Notably, this is identical to one in Batty et al. [2007], with a difference being that Δx varies as in (4). Finally, we solve the following new linear system to enable strongly two-way coupled dynamics

$$\begin{aligned}\Delta t \left([\nabla]^T [V][A][F][\nabla] + [J]^T [M_s]^{-1} [J] \right) \{p\} \\ = [\nabla]^T [V][A]\{\boldsymbol{u}^*\} - [J]^T \{\boldsymbol{w}^*\},\end{aligned} \tag{14}$$

where $[J] = [J_{\text{trans}}^T \; J_{\text{rot}}^T]^T$ and $\boldsymbol{w}^*$ is a 6×1 vector encoding both the linear and angular velocity of the rigid body. $[M_s]$ is a 6×6 mass matrix corresponding to a rigid body touching a fluid as in Batty et al. [2007]. Notice that we multiply Δt on the left-hand side of (14) to ensure unit consistency. The resulting pressure is subsequently used for both enforcing incompressibility $\{\boldsymbol{u}\} = \{\boldsymbol{u}^*\} - \Delta t [F][\nabla]\{p\}$ and updating the rigid body dynamics using (6) and (7).

4.5 Advection

Our simulation loop adopts operator splitting [Bridson 2008]; thus requires the updating of two variables: velocity and the level set, each defined at specific locations on staggered grids. More specifically, they undergo updates through advection [Foster and Fedkiw 2001]. For the sake of ease of implementation, we employ the semi-Lagrangian advection method [Stam 1999].

To make this work with our dynamically changing grids, we employ the strategy proposed in Klinger et al. [2006]; that is, we keep the grid and variables from the previous step. After constructing new quadtree tall-cell grids, at every respective sample location (e.g., cell centers and face centers), we interpolate velocity from the grid in the previous step and use it to backtrace the position, which is then used for interpolating variables of interest. Extrapolation of variables is handled in the same manner as described in Ando and Batty [2020].

When advection, the overall structure of cells becomes identical to that of octree cells. From this stage onwards, we can run the semi-Lagrangian advection and assign intermediate velocities to each cell face.

4.6 Simulation Loop

In the above, we described how to evaluate a sizing function, construct quadtree tall-cell grids, advect the level set, compute pressure, and update velocity. These components are now combined together to complete our simulation pipeline, which is laid in Algorithm 1. Note that quadtree tall-cells are mainly used in pressure solve (Line 9 in Algorithm 1) and we share many components with Ando and Batty [2020] for the rest of operations.

5 RESULTS

For evaluation purposes, we have performed three different techniques: uniform grids [Bridson 2008], tall cells [Irving et al. 2006], and our quadtree tall cells, all of which are based on our implementation. Note that we have implemented the method of Irving et al. [2006] slightly different from the original in that we employ our variational discretization to facilitate implementation.

Choosing an appropriate pressure solver to evaluate our method needs careful analysis, as we discuss below.

First, our method ensures the symmetric positive definite linear system by design through a variational formulation. However, this does not necessarily mean that our system forms an M-matrix, where off-diagonal entries are all negative while the diagonal terms remain positive. In our examples, we find that the MICCG solver by [Bridson 2008] was not the best choice, as we find it works most effectively when the coefficient matrix is M-Matrix. The ICCG solver provided by AMGCL [Demidov 2019] works in most cases, but we observe that, during highly complex moments, it converges poorly and is prone to stall. Ultimately, we have chosen an ICCG implemented with minor modifications from [Bridson 2008].[1] We employed this solver to all of our examples with the relative residual of 10^{-4}.

In our implementation, we leveraged the data layout as described in [Ando and Batty 2020] where all data is stored in layered sparse grids of different resolutions. Within this structure, all cells in the vertical direction reside in the same sparse grids. Hence, cache coherence is maximized for vertical scans. We also incorpolate extended narrow-band FLIP (EXNBFLIP) [Sato et al. 2018] to enrich visual expression.

Table 1 summarizes the runtime of the above three methods. As expected, the tall-cell method outperforms uniform grids due to a significant reduction in the number of degrees of freedom to be solved, resulting in accelerated convergence. Our method delivers additional performance gains by further reducing the degrees of freedom in the xz-plane in a quadtree fashion, which adapts to the scene's complexity.

Table 2 summarizes the details of the projection routine for the three methods mentioned above. We find that uniform tall-cell method consumes more iterations to converge than the uniform grid, but we also find that the uniform tall-cell method converges faster in practice, due to the reduced grid cell count. The performance of our method consistently outperforms the above two, achieving projection times that are 5× to 10× faster.

Figure 7 is the time-varying timing of our method and the reference simulations. Due to the adaptive nature, the number of grids

[1] We addressed some missing variables from their MICCG code in the process.

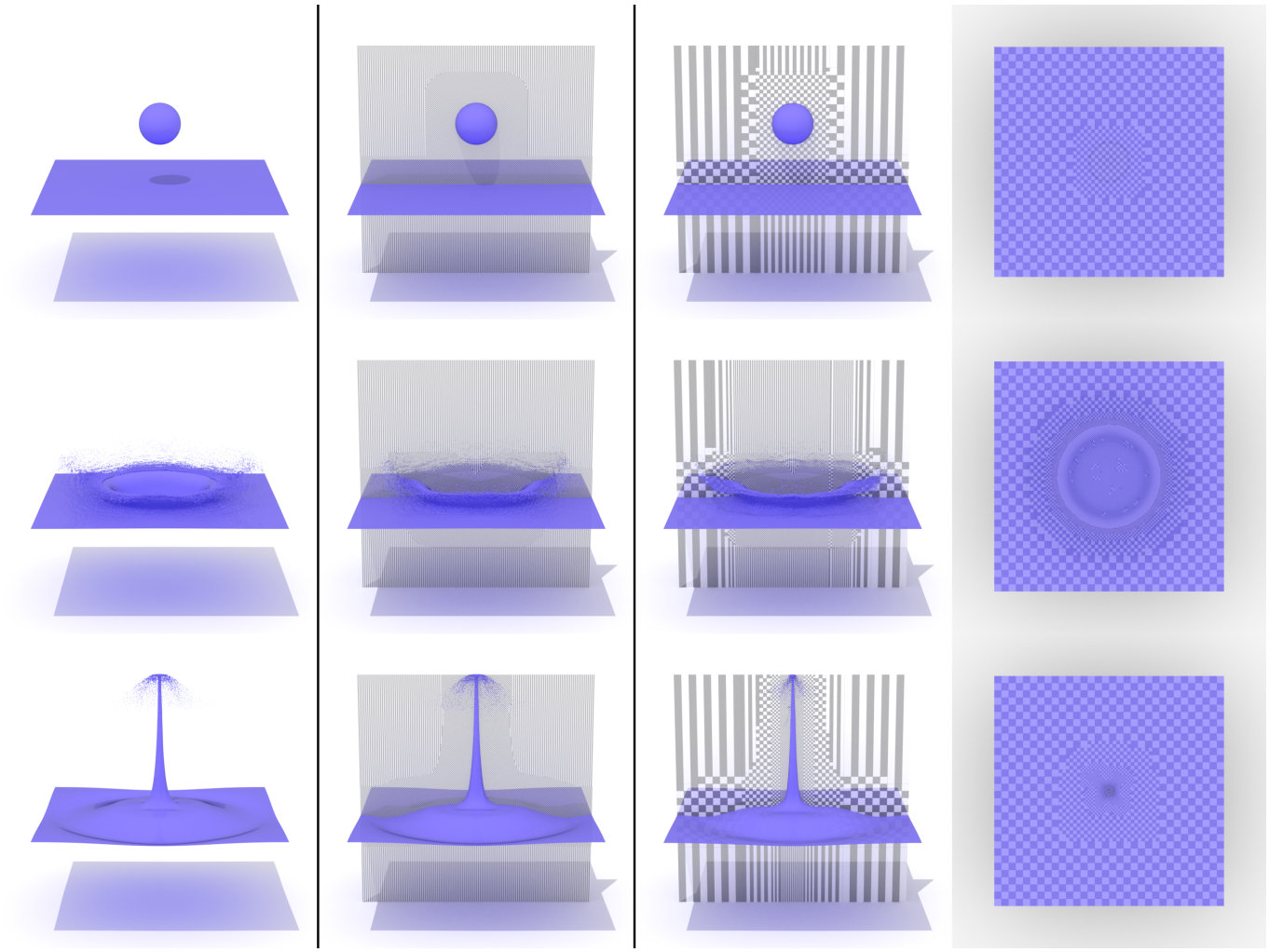


Fig. 5. Water drop. From left to right: uniform grids, uniformly populated tall cells, our quadtree tall cells, and the top view of our method. Effective resolution of 256^3. Notice that the shapes of both the crown splash and the rising spindles closely resemble each other among the three simulations. On the rightmost view, cell subdivision is carried out only where it is necessary, resulting in a significant reduction in the number of degrees of freedom.

and the quadtree structure dynamically changes, and thus the computation time per frame also fluctuates accordingly. Here, we would like to highlight that even at the most challenging moments, our method remains faster or as fast as tall cells.

Water Drop. Figure 5 shows a common testcase of a spherical liquid falling into a static pool, creating isotropic splashes. As can be seen, our method successfully resolves thin crowns and spindles rising at the center after the impact. On the rightmost column of Figure 5 is an xz-plane view of the same scene, providing a top view of the quadtree cell subdivision. This shows that the cells near the center are more likely to be subdivided as desired. Eventually, after a certain period of time, the motion of surfaces settles, and the variation in subdivision also diminishes. This scene was simulated with Ryzen 9 5950X.

Breaking Dam. Figure 6 is a breaking dam of water with the crest colliding with multiple static pillars. In this setup, we dictate the subdivision of cells not only on the visually important surfaces but also near the static pillars to more effectively reproduce the water splashes incurred by the waves crashing at the pillars, which produces animations that are closely similar to that of uniform grids with a notable performance gain. The right most is a top-down view of the same scene, illustrating that the subdivision is diagonally symmetric from the left-bottom to the right-top. We simulated this scene with Ryzen 9 5950X.

Two-Way Coupled Rigid Bodies. Our representative example of Figure 1 is two-way coupled rigidbodies gently bobbing at the edge of water tub. Toward the beginning of the scene, cells near the wave crashing are more likely to be subdivided, followed by the vicinity of rigid bodies when the crests reach them, creating intricate waves

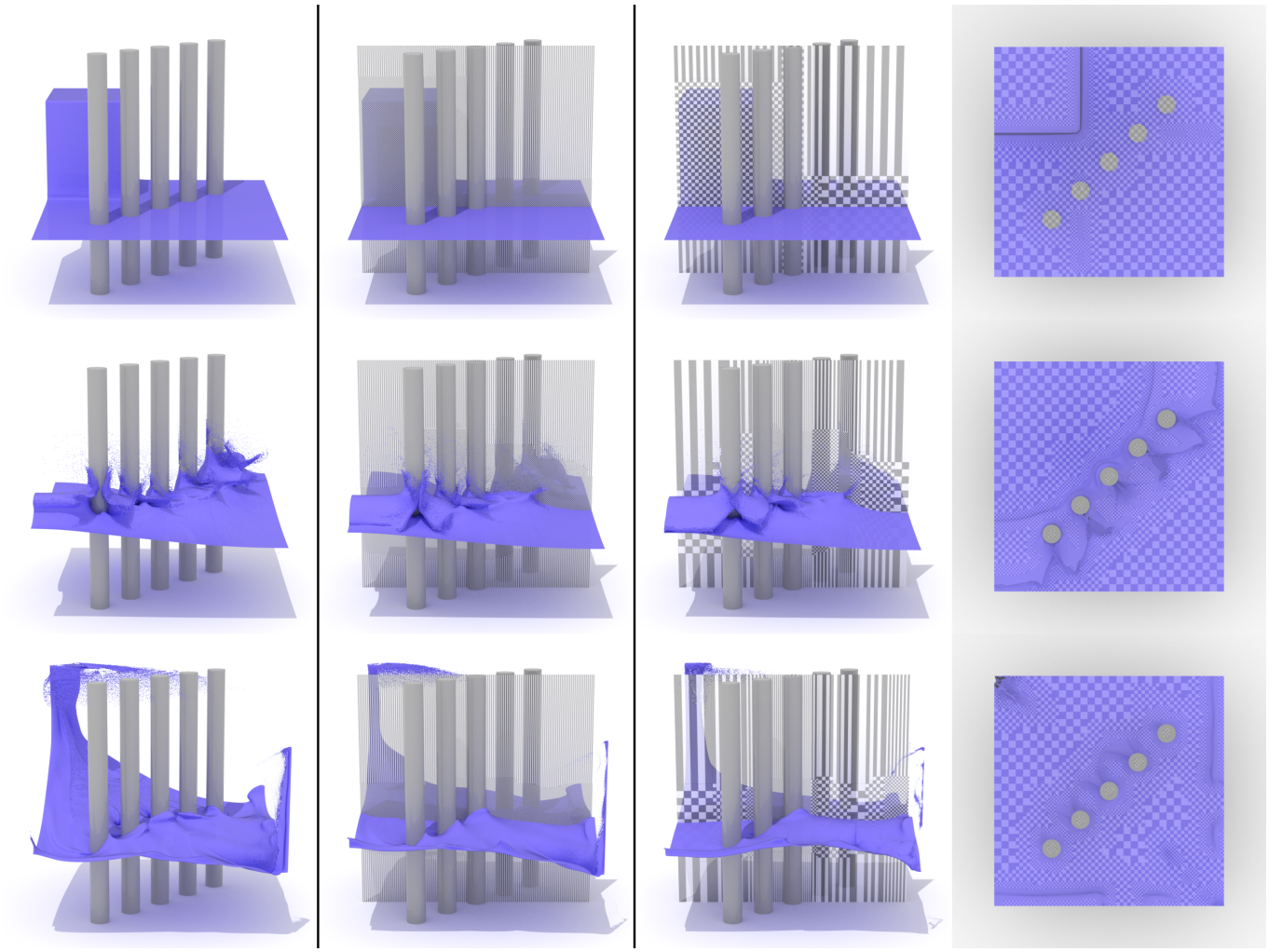


Fig. 6. Breaking dam. From left to right: uniform grids, uniformly populated tall cells, our quadtree tall cells, and the top view of our method. Effective resolution of 256^3. Our method dynamically subdivides cells, focusing on areas near the pillars and other visually significant regions, yet it still produces animations similar to two reference simulations, all the way from the dam collapse to the returning splashes at a corner.

near the rigid bodies waved by the two-way coupled dynamics. Toward the end, the rigid bodies undergo calm motion in harmony with the motion of water, occasionally intersecting with tall cells of different sizes. At this stage, cells on the left tends to be coarsened since waves are smooth and nearly static. This scene was also simulated with Ryzen 9 5950X. We used BulletPhysics [Coumans and Bai 2021] for handling collisions between rigid bodies.

Merging Animals. Figure 8 illustrates a scenario where liquid, shaped as various characters, falls into a static pool, creating complex thin sheets and splashes. Despite the relatively large scale of the scene, with a resolution of 512^3, our method enables efficient simulation within a reasonable computation time. This simulation was performed on a Ryzen 9 5950X processor.

6 DISCUSSIONS

Numerical Damping. Our method exhibits visually satisfactory results for many practical scenarios, except that it inevitably introduces noticeable damping. This limitation is consistent with previous spatially adaptive methods such as [Ando and Batty 2020] in that coarse grid approximations tend to wipe out detail flow that cannot be captured with the cells.

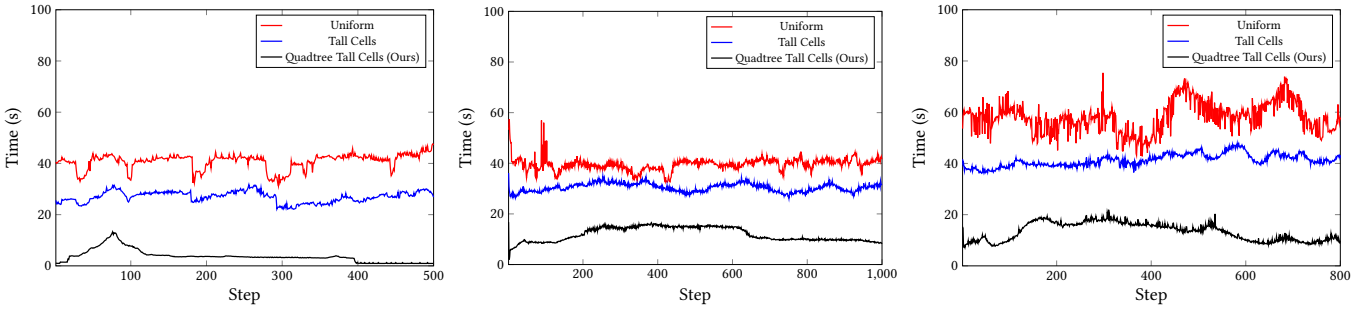
For example, subtle variations in velocity and surface detail may be lost at cell transitions near T-junctions, partly due to the Nyquist limit. While not exactly the same, similar artifacts, such as ghost reflections, are discussed in the Perfectly Matched Layer (PML) configuration [Söderström et al. 2010b]. If preserving both total energy and surface detail is critical, one may either globally reinject kinetic energy [Lentine et al. 2011] or maintain high-resolution level set surfaces, while retaining the coarse grid representations [Goldade et al. 2016].

Table 1. Timing breakdown of three methods. The numbers represent seconds on average per step for each computational stage.

Scene	Resol.	Method	Sizing value evaluation	Quadtree construction	Advection	Tall cells construction	Projection	Extrapolation	EXNBFLIP operation	Total
Fig. 5	256^3	Uniform	-	-	4.472	-	32.413	1.609	2.853	41.716
		Tall	-	-	4.557	0.970	16.475	2.130	2.865	27.364
		Ours	0.070	0.504	0.770	0.549	0.480	0.177	0.836	3.462
Fig. 6	256^3	Uniform	-	-	4.893	-	30.370	1.847	2.768	40.246
		Tall	-	-	4.829	0.987	18.521	2.271	3.153	30.123
		Ours	0.395	0.975	1.962	0.696	2.207	1.049	1.517	8.895
Fig. 1	512×192^2	Uniform	-	-	5.319	-	38.414	2.824	2.992	57.479
		Tall	-	-	5.266	1.038	22.080	2.845	3.322	41.483
		Ours	0.230	1.030	1.755	0.662	4.476	0.765	1.508	12.275

Table 2. The breakdown of projection steps of three representative methods. All numbers are measured as averages per step. The assembled matrix is the sum of all processes involved in the matrix assembly, including gradient evaluation, divergence computation, and multiplications related to two-way coupling with rigid bodies (if needed). Residual numbers refers to the absolute residuals in PCG, all recorded using the $\|L\|_\infty$ norm.

Scene	Resol.	Method	Total cells	Assemble matrix (seconds)	Solve poisson's equation (seconds)	Initial residual	Residual Returned	Iterations
Fig. 5	256^3	Uniform	6779709	2.897	28.456	1.0×10^{-2}	1.0×10^{-6}	229
		Tall	2556400	1.795	12.108	3.3×10^{-2}	3.0×10^{-6}	264
		Ours	126031	0.061	0.318	9.9×10^{-2}	8.0×10^{-6}	95
Fig. 6	256^3	Uniform	6054119	2.918	26.452	3.5×10^{-2}	3.0×10^{-6}	241
		Tall	2828746	2.527	13.529	3.0×10^{-1}	2.6×10^{-5}	269
		Ours	568377	0.339	1.416	4.1	3.5×10^{-4}	140
Fig. 1	512×192^2	Uniform	5347016	5.125	30.777	2.9×10^{-2}	3.0×10^{-6}	220
		Tall	2288191	3.171	15.618	8.3×10^{-2}	6.0×10^{-6}	252
		Ours	316432	0.351	3.529	5.2×10^{-1}	4.9×10^{-5}	169



(a) Water drop scene (Fig. 5). (b) Breaking dam scene (Fig. 6). (c) Two way coupling scene (Fig. 1).

Fig. 7. Timing comparison of three methods. Uniform grids are represented in red, tall cells in blue, and our method in black.

Refinement Oracle. We used surface curvature and variations in velocity near the liquid surface because our primary interest is the motion on the surface. We did not consider inner turbulence, as our tall cells by construction approximate both pressure and velocity variance linearly with depth. We note that the strategy of omitting inner turbulence is reportedly effective for visual simulation [Da et al. 2016; Zhai et al. 2020]. Consequently, the refinement oracle should be simpler than those that consider volumetric variance [Ando and Batty 2020; Ando et al. 2013].

FLIP. We used an extended narrowband FLIP [Sato et al. 2018], which resorts to traditional level set methods for coarse grids. Although a more common FLIP or APIC [Jiang et al. 2015] is possible, distributing uniform particles across the surfaces would not fully leverage the benefits of our quadtree surface adaptivity. Alternatively, we may seed particles with radii and spacing proportional to the grid cell size [Ando et al. 2013], but doing so would result in *particle bumpiness* when visualized without care, as demonstrated by [Sato et al. 2018].

Optimal Thickness. The thickness of the thin layer beneath the liquid surfaces is a major factor in determining the trade-off between runtime and the overall accuracy of the animation, both for advection and pressure computation. For example, a thinner layer results in faster performance, while a thicker layer yields more accurate results. As suggested by Irving et al. [2006], we have opted for a thickness of $1/4$ of the depth. Selecting an optimal thickness remains a heuristic, and the designing a dynamically changing thickness for improved efficiency is a topic we leave for future work.

Linear System Inversion. As we discussed, our linear system solver departs from an ideal M-matrix, which may limit the choice of preconditioner for the CG method. Also, we note that our system

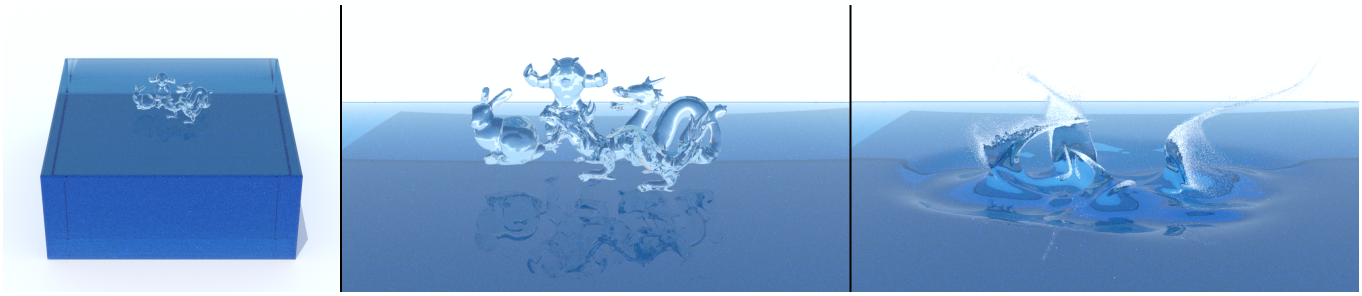


Fig. 8. Merging animals: Various characters fall into the liquid, forming complex thin sheets and splashes. The simulation achieves an effective resolution of 512^3 with an average computation time of 41.3 seconds per step.

could be more challenging to invert compared to the uniform tall cells from the view of condition number, due to the newly introduced horizontal adaptivity, as the variance of eigenvalues tends to deviate. Therefore, we believe that designing a good preconditioner would be a promising future work in further performance improvements. In particular, our method is based on the quadtree, which should facilitate the design of a geometric multigrid solver [McAdams et al. 2010]. Nevertheless, we highlight that despite the above challenges, our approach, with its significantly reduced degrees of freedom, offers various advantages, including reduced memory allocation impact, faster advection and a more efficient surface reconstruction.

Memory Allocation Impact. We base our implementation on Ando and Batty [2020] to fully leverage the benefits of surface reconstruction and seamless interpolation near T-junctions. For tall cells, as we discussed in Section 4.2, we concatenate cubical cells to create tall cells as needed, and conversely, we split them back to the cubical cells after the pressure solve, enabling the use of Moving Least Squares (MLS) interpolation [Ando and Batty 2020]. Of course, this comes at the cost of increased memory allocation and subdivision overheads. We have chosen this for the sake of implementation simplicity, but it should be straightforward to optimize it to minimize any explicit allocation and subdivision. For instance, one can subdivide the tall-cells locally on the fly and discard them once either the interpolation or averaging is completed. Nonetheless, we highlight that despite the above overheads, our method is still remarkably faster than alternatives, as seen in Table 1 and Figure 7.

7 CONCLUSIONS

Our quadtree tall-cells can be seen as a sweet spot between the general adaptive methods, which offer adaptivity but often come with algorithmic complexity, and tall-cell grids, which are efficient but lack the horizontal adaptivity needed for certain scenarios. Our numerical experiments demonstrate that our method is more efficient than tall-cell methods while still maintaining algorithmic simplicity.

Our variational pressure solver is more affordable compared to existing tall-cell methods because once the gradient operator is computed, the divergence operator becomes readily available as the transpose of the gradient operator, which facilitates implementation even without the need for adaptivity. We have also taken advantage of this variational view to incorporate monolithic two-way coupled rigid bodies, which to the best of our knowledge has not been achieved before for tall-cell grids.

We believe that our variational approach opens up the way for several research avenues in the realm of tall-cell methods, including coupling with deformables and air dynamics. We are also interested to explore GPU acceleration for real-time applications.

ACKNOWLEDGMENTS

We would like to thank Tetsuya Takahashi for the helpful discussions on pressure solvers, and the anonymous reviewers for their valuable comments and constructive suggestions, which greatly contributed to improving the quality of this paper. This work is supported in part by JSPS KAKENHI (25K15401).

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 (July 2017), 12 pages. https://doi.org/10.1145/3072959.3073625

Bart Adams, Mark Pauly, Richard Keiser, and Leonidas J. Guibas. 2007. Adaptively Sampled Particle Fluids. *ACM Trans. Graph.* 26, 3 (July 2007), 48–es. https://doi.org/10.1145/1276377.1276437

Pierre Alliez, David Cohen-Steiner, Mariette Yvinec, and Mathieu Desbrun. 2005. Variational Tetrahedral Meshing. *ACM Trans. Graph.* 24, 3 (July 2005), 617–625. https://doi.org/10.1145/1073204.1073238

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

Ryoichi Ando, Nils Thürey, and Chris Wojtan. 2013. Highly Adaptive Liquid Simulations on Tetrahedral Meshes. *ACM Trans. Graph.* 32, 4, Article 103 (July 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. *Computer Graphics Forum* 34, 2 (2015), 473–480. https://doi.org/10.1111/cgf.12576

Vinicius C. Azevedo and Manuel M. Oliveira. 2013. Efficient Smoke Simulation on Curvilinear Grids. *Computer Graphics Forum* 32, 7 (2013), 235–244. https://doi.org/10.1111/cgf.12231

Christopher Batty, Florence Bertails, and Robert Bridson. 2007. A Fast Variational Framework for Accurate Solid-Fluid Coupling. *ACM Trans. Graph.* 26, 3 (July 2007), 100–es. https://doi.org/10.1145/1276377.1276502

Christopher Batty and Ben Houston. 2011. A Simple Finite Volume Method for Adaptive Viscous Liquids. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM, New York, NY, USA, 111–118. https://doi.org/10.1145/2019406.2019421

Christopher Batty, Stefan Xenos, and Ben Houston. 2010. Tetrahedral Embedded Boundary Methods for Accurate and Flexible Adaptive Fluids. *Computer Graphics Forum* 29, 2 (2010), 695–704. https://doi.org/10.1111/j.1467-8659.2009.01639.x

Robert Bridson. 2008. *Fluid Simulation*. A. K. Peters, Ltd., USA.

José A. Canabal, David Miraut, Nils Thuerey, Theodore Kim, Javier Portilla, and Miguel A. Otaduy. 2016. Dispersion Kernels for Water Wave Simulation. *ACM Trans. Graph.* 35, 6, Article 202 (November 2016), 10 pages. https://doi.org/10.1145/

2980179.2982415

Nuttapong Chentanez, Bryan E. Feldman, François Labelle, James F. O'Brien, and Jonathan R. Shewchuk. 2007. Liquid Simulation on Lattice-Based Tetrahedral Meshes. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association, Goslar, DEU, 219–228. https://doi.org/10.2312/SCA/SCA07/219-228

Nuttapong Chentanez and Matthias Müller. 2011. Real-Time Eulerian Water Simulation Using a Restricted Tall Cell Grid. *ACM Trans. Graph.* 30, 4, Article 82 (July 2011), 10 pages. https://doi.org/10.1145/2010324.1964977

Nuttapong Chentanez, Matthias Müller, and Tae-Yong Kim. 2014. Coupling 3D Eulerian, Heightfield and Particle Methods for Interactive Simulation of Large Scale Liquid Phenomena. In *Proc. Eurographics/ ACM SIGGRAPH Symposium on Computer Animation*. The Eurographics Association, Goslar, DEU, 1–10. https://doi.org/10.2312/sca.20141117

Pascal Clausen, Martin Wicke, Jonathan R. Shewchuk, and James F. O'Brien. 2013. Simulating Liquids and Solid-Liquid Interactions with Lagrangian Meshes. *ACM Trans. Graph.* 32, 2, Article 17 (April 2013), 15 pages. https://doi.org/10.1145/2451236.2451243

Erwin Coumans and Yunfei Bai. 2016–2021. PyBullet, a Python module for physics simulation for games, robotics and machine learning. http://pybullet.org.

Fang Da, David Hahn, Christopher Batty, Chris Wojtan, and Eitan Grinspun. 2016. Surface-Only Liquids. *ACM Trans. Graph.* 35, 4, Article 78 (July 2016), 12 pages. https://doi.org/10.1145/2897824.2925899

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 (July 2015), 11 pages. https://doi.org/10.1145/2766901

Denis Demidov. 2019. AMGCL: An Efficient, Flexible, and Extensible Algebraic Multigrid Implementation. *Lobachevskii Journal of Mathematics* 40, 5 (01 May 2019), 535–546. https://doi.org/10.1134/S1995080219050056

Yoshinori Dobashi, Yasuhiro Matsuda, Tsuyoshi Yamamoto, and Tomoyuki Nishita. 2008. A Fast Simulation Method Using Overlapping Grids for Interactions between Smoke and Rigid Objects. *Computer Graphics Forum* 27, 2 (2008), 477–486. https://doi.org/10.1111/j.1467-8659.2008.01145.x

R. Elliot English, Linhai Qiu, Yue Yu, and Ronald Fedkiw. 2013. Chimera Grids for Water Simulation. In *Proc. 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM, New York, NY, USA, 85–94. https://doi.org/10.1145/2485895.2485897

Bryan E. Feldman, James F. O'Brien, and Bryan M. Klingner. 2005. Animating Gases with Hybrid Meshes. *ACM Trans. Graph.* 24, 3 (July 2005), 904–909. https://doi.org/10.1145/1073204.1073281

Florian Ferstl, Rüdiger Westermann, and Christian Dick. 2014. Large-Scale Liquid Simulation on Adaptive Hexahedral Grids. *IEEE Trans. Vis. Comput. Graph.* 20, 10 (2014), 1405–1417. https://doi.org/10.1109/TVCG.2014.2307873

Nick Foster and Ronald Fedkiw. 2001. Practical Animation of Liquids. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*. Association for Computing Machinery, New York, NY, USA, 23–30. https://doi.org/10.1145/383259.383261

Xifeng Gao, Jin Huang, Kaoji Xu, Zherong Pan, Zhigang Deng, and Guoning Chen. 2017. Evaluating Hex-mesh Quality Metrics via Correlation Analysis. *Computer Graphics Forum* 36, 5 (2017), 105–116. https://doi.org/10.1111/cgf.13249

Abhinav Golas, Rahul Narain, Jason Sewall, Pavel Krajcevski, Pradeep Dubey, and Ming Lin. 2012. Large-Scale Fluid Simulation Using Velocity-Vorticity Domain Decomposition. *ACM Trans. Graph.* 31, 6, Article 148 (November 2012), 9 pages. https://doi.org/10.1145/2366145.2366167

Ryan Goldade, Christopher Batty, and Chris Wojtan. 2016. A Practical Method for High-Resolution Embedded Liquid Surfaces. *Computer Graphics Forum* 35, 2 (2016), 233–242. https://doi.org/10.1111/cgf.12826

Ryan Goldade, Yipeng Wang, Mridul Aanjaneya, and Christopher Batty. 2019. An Adaptive Variational Finite Difference Framework for Efficient Symmetric Octree Viscosity. *ACM Trans. Graph.* 38, 4, Article 94 (July 2019), 14 pages. https://doi.org/10.1145/3306346.3322939

Yixin Hu, Teseo Schneider, Bolun Wang, Denis Zorin, and Daniele Panozzo. 2020. Fast Tetrahedral Meshing in the Wild. *ACM Trans. Graph.* 39, 4, Article 117 (July 2020), 18 pages. https://doi.org/10.1145/3386569.3392385

Libo Huang and Dominik L. Michels. 2020. Surface-only ferrofluids. *ACM Trans. Graph.* 39, 6 (2020), 174:1–174:17. https://doi.org/10.1145/3414685.3417799

Libo Huang, Ziyin Qu, Xun Tan, Xinxin Zhang, Dominik L. Michels, and Chenfanfu Jiang. 2021. Ships, Splashes, and Waves on a Vast Ocean. *ACM Trans. Graph.* 40, 6, Article 203 (dec 2021), 15 pages. https://doi.org/10.1145/3478513.3480495

Hikaru Ibayashi, Chris Wojtan, Nils Thuerey, Takeo Igarashi, and Ryoichi Ando. 2020. Simulating Liquids on Dynamically Warping Grids. *IEEE Trans. Vis. Comput. Graph.* 26, 6 (2020), 2288–2302. https://doi.org/10.1109/TVCG.2018.2883628

Geoffrey Irving, Eran Guendelman, Frank Losasso, and Ronald Fedkiw. 2006. Efficient Simulation of Large Bodies of Water by Coupling Two and Three Dimensional Techniques. *ACM Trans. Graph.* 25, 3 (July 2006), 805–811. https://doi.org/10.1145/1141911.1141959

Tobin Isaac, Carsten Burstedde, and Omar 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

Stefan Jeschke and Chris Wojtan. 2015. Water Wave Animation via Wavefront Parameter Interpolation. *ACM Trans. Graph.* 34, 3, Article 27 (May 2015), 14 pages. https://doi.org/10.1145/2714572

Stefan Jeschke and Chris Wojtan. 2017. Water Wave Packets. *ACM Trans. Graph.* 36, 4, Article 103 (July 2017), 12 pages. https://doi.org/10.1145/3072959.3073678

Chenfanfu Jiang, Craig Schroeder, Andrew Selle, Joseph Teran, and Alexey Stomakhin. 2015. The Affine Particle-in-Cell Method. *ACM Trans. Graph.* 34, 4, Article 51 (July 2015), 10 pages. https://doi.org/10.1145/2766996

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

Michael Kass and Gavin Miller. 1990. Rapid, Stable Fluid Dynamics for Computer Graphics. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques* (Dallas, TX, USA) *(SIGGRAPH '90)*. Association for Computing Machinery, New York, NY, USA, 49–57. https://doi.org/10.1145/97879.97884

Todd Keeler and Robert Bridson. 2014. Ocean Waves Animation using Boundary Integral Equations and Explicit Mesh Tracking. In *Proc. urographics/ ACM SIGGRAPH Symposium on Computer Animation*. The Eurographics Association, Goslar, DEU, 11–19. https://doi.org/10.2312/sca.20141118

Theodore Kim, Jerry Tessendorf, and Nils Thürey. 2013. Closest Point Turbulence for Liquid Surfaces. *ACM Trans. Graph.* 32, 2, Article 15 (April 2013), 13 pages. https://doi.org/10.1145/2451236.2451241

Bryan M. Klingner, Bryan E. Feldman, Nuttapong Chentanez, and James F. O'Brien. 2006. Fluid Animation with Dynamic Meshes. *ACM Trans. Graph.* 25, 3 (July 2006), 820–825. https://doi.org/10.1145/1141911.1141961

François Labelle and Jonathan Richard Shewchuk. 2007. Isosurface Stuffing: Fast Tetrahedral Meshes with Good Dihedral Angles. *ACM Trans. Graph.* 26, 3 (July 2007), 57–es. https://doi.org/10.1145/1276377.1276448

Anita T. Layton and Michiel van de Panne. 2002. A numerically efficient and stable algorithm for animating water waves. *The Visual Computer* 18, 1 (Feb. 2002), 41–53. https://doi.org/10.1007/s003710100131

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* (Vancouver, British Columbia, Canada) *(SCA '11)*. Association for Computing Machinery, New York, NY, USA, 91–100. https://doi.org/10.1145/2019406.2019419

Michael Lentine, Wen Zheng, and Ronald Fedkiw. 2010. A Novel Algorithm for Incompressible Flow Using Only a Coarse Grid Projection. *ACM Trans. Graph.* 29, 4, Article 114 (July 2010), 9 pages. https://doi.org/10.1145/1778765.1778851

Steve Lesser, Alexey Stomakhin, Gilles Daviet, Joel Wretborn, John Edholm, Noh-Hoon Lee, Eston Schweickart, Xiao Zhai, Sean Flynn, and Andrew Moffat. 2022. Loki: A Unified Multiphysics Simulation Framework for Production. *ACM Trans. Graph.* 41, 4, Article 50 (jul 2022), 20 pages. https://doi.org/10.1145/3528223.3530058

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 (August 2004), 457–462. https://doi.org/10.1145/1015706.1015745

Jörn Loviscach. 2002. A Convolution-Based Algorithm for Animated Water Waves. In *Eurographics 2002 - Short Presentations*. Eurographics Association, 7 pages. https://doi.org/10.2312/egs.20021032

Pierre-Luc Manteaux, Chris Wojtan, Rahul Narain, Stéphane Redon, François Faure, and Marie-Paule Cani. 2017. Adaptive Physically Based Models in Computer Graphics. *Computer Graphics Forum* 36, 6 (2017), 312–337. https://doi.org/10.1111/cgf.12941

Aleka McAdams, Eftychios Sifakis, and Joseph Teran. 2010. A Parallel Multigrid Poisson Solver for Fluids Simulation on Large Grids. In *Proc. Eurographics/ ACM SIGGRAPH Symposium on Computer Animation*. The Eurographics Association, Goslar, DEU, 65–74. https://doi.org/10.2312/SCA/SCA10/065-073

Marek Krzysztof Misztal, Robert Bridson, Kenny Erleben, Jakob Andreas Bærentzen, and François Anton. 2010. Optimization-based Fluid Simulation on Unstructured Meshes. In *Workshop in Virtual Reality Interactions and Physical Simulation (VRIPHYS)*. The Eurographics Association, 11–20. https://doi.org/10.2312/PE/vriphys/vriphys10/011-020

Neil Molino, Robert Bridson, Joseph Teran, and Ronald Fedkiw. 2003. A Crystalline, Red Green Strategy for Meshing Highly Deformable Objects with Tetrahedra. In *Proc. 12th International Meshing Roundtable*. 103–114.

David M. Mount and Sunil Arya. 1998. ANN: A Library for Approximate Nearest Neighbor Searching. https://www.cs.umd.edu/~mount/ANN/

Michael B. Nielsen and Robert Bridson. 2016. Spatially Adaptive FLIP Fluid Simulations in Bifrost. In *ACM SIGGRAPH 2016 Talks* (Anaheim, California) *(SIGGRAPH '16)*. Association for Computing Machinery, New York, NY, USA, Article 41, 2 pages.

https://doi.org/10.1145/2897839.2927399

Ziyin Qu, Minchen Li, Fernando De Goes, and Chenfanfu Jiang. 2022. The Power Particle-in-Cell Method. *ACM Trans. Graph.* 41, 4, Article 118 (jul 2022), 13 pages. https://doi.org/10.1145/3528223.3530066

Takahiro Sato, Chris Wojtan, Nils Thuerey, Takeo Igarashi, and Ryoichi Ando. 2018. Extended Narrow Band FLIP for Liquid Simulations. *Computer Graphics Forum* 37, 2 (2018), 169–177. https://doi.org/10.1111/cgf.13351

Camille Schreck, Christian Hafner, and Chris Wojtan. 2019. Fundamental Solutions for Water Wave Animation. *ACM Trans. Graph.* 38, 4, Article 130 (July 2019), 14 pages. https://doi.org/10.1145/3306346.3323002

Camille Schreck and Chris Wojtan. 2022. Coupling 3D Liquid Simulation with 2D Wave Propagation for Large Scale Water Surface Animation Using the Equivalent Sources Method. *Computer Graphics Forum* 41, 2 (2022), 343–353. https://doi.org/10.1111/cgf.14478

Hang Si. 2015. TetGen, a Delaunay-Based Quality Tetrahedral Mesh Generator. *ACM Trans. Math. Softw.* 41, 2, Article 11 (feb 2015), 36 pages. https://doi.org/10.1145/2629697

Tomas Skrivan, Andreas Soderstrom, John Johansson, Christoph Sprenger, Ken Museth, and Chris Wojtan. 2020. Wave Curves: Simulating Lagrangian Water Waves on Dynamically Deforming Surfaces. *ACM Trans. Graph.* 39, 4, Article 65 (July 2020), 12 pages. https://doi.org/10.1145/3386569.3392466

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

Andreas Söderström, Matts Karlsson, and Ken Museth. 2010b. A PML-based nonreflective boundary for free surface fluid animation. *ACM Trans. Graph.* 29, 5, Article 136 (Nov. 2010), 17 pages. https://doi.org/10.1145/1857907.1857912

Barbara Solenthaler and Markus Gross. 2011. Two-Scale Particle Simulation. *ACM Trans. Graph.* 30, 4, Article 81 (July 2011), 8 pages. https://doi.org/10.1145/2010324.1964976

Tommaso Sorgente, Silvia Biasotti, Gianmarco Manzini, and Michela Spagnuolo. 2023. A Survey of Indicators for Mesh Quality Assessment. *Computer Graphics Forum* 42, 2 (2023), 461–483. https://doi.org/10.1111/cgf.14779

Jos Stam. 1999. Stable Fluids. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99)*. ACM Press/Addison-Wesley Publishing Co., USA, 121–128. https://doi.org/10.1145/311535.311548

Nils Thürey, Matthias Müller-Fischer, Simon Schirm, and Markus Gross. 2007. Real-time Breaking Waves for Shallow Water Simulations. In *15th Pacific Conference on Computer Graphics and Applications (PG'07)*. 39–46. https://doi.org/10.1109/PG.2007.33

Nils Thürey, Ulrich Rüde, and Marc Stamminger. 2006. Animation of Open Water Phenomena with coupled Shallow Water and Free Surface Simulations. In *Proc. ACM SIGGRAPH / Eurographics Symposium on Computer Animation*. The Eurographics Association, Goslar, DEU, 157–164. https://doi.org/10.2312/SCA/SCA06/157-165

Huamin Wang, Gavin Miller, and Greg Turk. 2007. Solving General ShallowWave Equations on Surfaces. In *Proc. Eurographics/SIGGRAPH Symposium on Computer Animation*. The Eurographics Association, Goslar, DEU, 229–238. https://doi.org/10.2312/SCA/SCA07/229-238

Jun Wang and Zeyun Yu. 2011. Adaptive and Quality Tetrahedral Mesh Generation. In *2011 IEEE International Conference on Computer Science and Automation Engineering*, Vol. 3. 323–327. https://doi.org/10.1109/CSAE.2011.5952690

Rene Winchenbach and Andreas Kolb. 2021. Optimized Refinement for Spatially Adaptive SPH. *ACM Trans. Graph.* 40, 1, Article 8 (jan 2021), 15 pages. https://doi.org/10.1145/3363555

Yuwei Xiao, Szeyu Chan, Siqi Wang, Bo Zhu, and Xubo Yang. 2020. An adaptive staggered-tilted grid for incompressible flow simulation. *ACM Trans. Graph.* 39, 6 (2020), 171:1–171:15. https://doi.org/10.1145/3414685.3417837

Cem Yuksel, Donald H. House, and John Keyser. 2007. Wave Particles. *ACM Trans. Graph.* 26, 3 (July 2007), 99–es. https://doi.org/10.1145/1276377.1276501

Xiao Zhai, Fei Hou, Hong Qin, and Aimin Hao. 2020. Fluid Simulation with Adaptive Staggered Power Particles on GPUs. *IEEE Transactions on Visualization and Computer Graphics* 26, 6 (2020), 2234–2246. https://doi.org/10.1109/TVCG.2018.2886322

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 (July 2013), 12 pages. https://doi.org/10.1145/2461912.2461999