An Adaptive Staggered-Tilted Grid for Incompressible Flow Simulation

YUWEI XIAO, Shanghai Jiao Tong University
SZEYU CHAN, University of Pennsylvania
SIQI WANG, New York University
BO ZHU*, Dartmouth College
XUBO YANG*, Shanghai Jiao Tong University

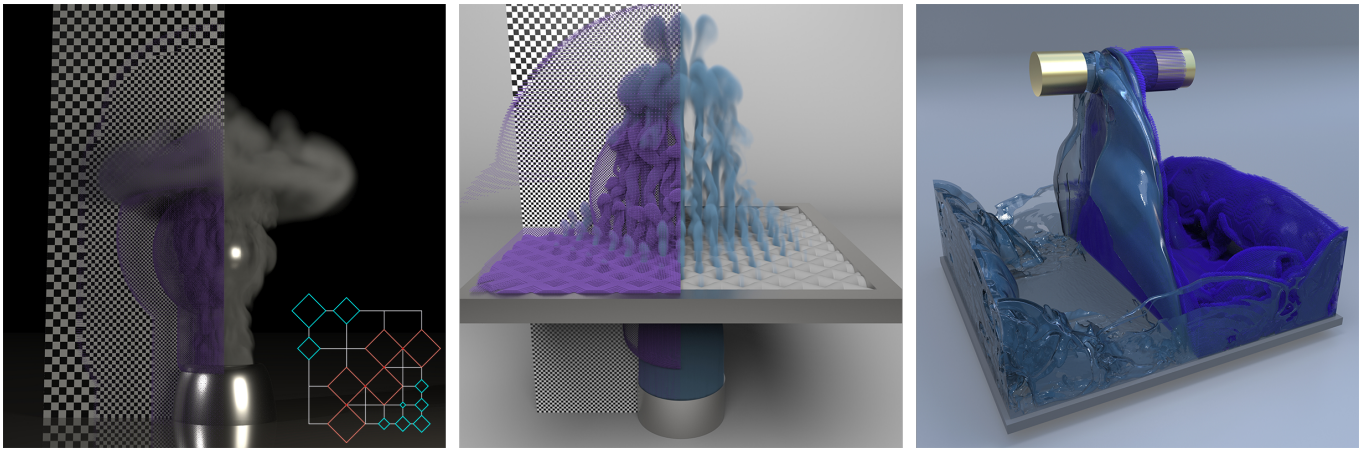


Fig. 1. The AST grid structure provides a new regular pattern for the adaptive fluid simulation that is able to be integrated with existing grid structures such as an Octree. Active tilted cells in the AST grid are rendered in purple and darker blue color. (Left) A smoke plume passes around a sphere. Tilted cells and the four-level Octree pattern are visualized. (Middle) A smoke plume passes through a solid porous grate. The AST grid places extra DoFs around smoke and in the vicinity of the porous grate to capture detailed flow features. (Right) Two sources inject liquid to form a thin sheet. Tilted cells are activated in the narrowband of the level set that is evolving with the liquid.

Enabling adaptivity on a uniform Cartesian grid is challenging due to its highly structured grid cells and axis-aligned grid lines. In this paper, we propose a new grid structure – the adaptive staggered-tilted (AST) grid – to conduct adaptive fluid simulations on a regular discretization. The key mechanics underpinning our new grid structure is to allow the emergence of a new set of tilted grid cells from the nodal positions on a background uniform grid. The original axis-aligned cells, in conjunction with the populated axis-tilted cells, jointly function as the geometric primitives to enable adaptivity on a regular spatial discretization. By controlling the states of the tilted cells both temporally and spatially, we can dynamically evolve the adaptive discretizations on an Eulerian domain. Our grid structure preserves almost all the computational merits of a uniform Cartesian grid, including the cache-coherent data layout, the easiness for parallelization, and the existence of high-performance numerical solvers. Further, our grid structure can be integrated into other adaptive grid structures, such as an Octree or a sparsely populated grid, to accommodate the T-junction-free hierarchy. We demonstrate the efficacy of our AST grid by showing examples of large-scale incompressible flow simulation in domains with irregular boundaries.

Additional Key Words and Phrases: fluid simulation, grid structure, adaptive simulation

ACM Reference Format:
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, Article 171 (December 2020), 15 pages. https://doi.org/10.1145/3414685.3417837

*Corresponding authors

Authors' addresses: Yuwei Xiao, Shanghai Jiao Tong University, xiaoyuwei@sjtu.edu.cn; Szeyu Chan, University of Pennsylvania, sychan@seas.upenn.edu; Siqi Wang, New York University, siqi.wang@nyu.edu; Bo Zhu, Dartmouth College, bo.zhu@dartmouth.edu; Xubo Yang, Shanghai Jiao Tong University, yangxubo@sjtu.edu.cn.

0730-0301/2020/12-ART171 $15.00
https://doi.org/10.1145/3414685.3417837

1 INTRODUCTION

Regularity is essential for building an Eulerian solver for large-scale physical simulations. A regular Eulerian structure, e.g., a Cartesian grid, is distinguished from its unstructured counterparts (e.g., particle or mesh) by its random access ability, cache coherent layout, and easiness in hosting large-scale parallel solvers. By requiring scarcely any additional large-bulk memory storage or acceleration data structures, we can randomly access a degree of freedom or a quadrature shape defined on a regular grid structure by performing an $O(1)$ operation. Such discretization regularity can be typically

encoded in an analytical expression [Azevedo and Oliveira 2013; Houston et al. 2006; Ibayashi et al. 2018], a cacheable index lookup table [Zhu et al. 2013], or a hardware-accelerated page access mechanism [Setaluri et al. 2014], etc. However, on the minus side, a regular Cartesian discretization suffers from a handful of weaknesses. In particular, compared with its mesh and particle counterparts, a regular grid structure lacks the inherent ability for adaptivity and dynamic Lagrangian structure tracking, limiting its capability in capturing intricate dynamics and complicated geometries of a flow system that exhibit multi-scale and heterogeneous features evolving rapidly in space. To overcome these limits, two streams of previous efforts have been successfully devoted to tackling the adaptivity challenge. First, a variety of hierarchical or transformable geometric primitives have been created on an Eulerian grid structure to enhance adaptivity. Most of the efforts within this category conduct a set of specific geometric operations on Cartesian grid cells to enhance adaptivity, e.g., by splitting cells (Octree [Losasso et al. 2004]), deleting cells (SPGrid [Setaluri et al. 2014]), stretching cells (far-field grid [Zhu et al. 2013]), merging cells (RLE grid[Chentanez and Müller 2011; Irving et al. 2006]), deforming cells (curvilinear grid [Azevedo and Oliveira 2013]), or rigidly translating and overlapping cells (Chimera grid [English et al. 2013]). These adaptive grid structures realize their layout regularity by assuming a connectivity that each grid cell should be a square (in 2D) or a cube (in 3D) adjacent to up to six neighbors (on the same adaptive level). Second, a surge of approaches has been proposed to regularize unstructured mesh discretizations to benefit from both its Lagrangian tracking and structured Eulerian data processing capability. For example, Chentanez et al. [2007] build a body-centered cubic (BCC) lattice using isosurface stuffing[Labelle and Shewchuk 2007], achieving the storage saving by having only four shapes of the tetrahedron in the interior of the mesh. Brochu et al. [2010] build a Voronoi mesh by applying geometric-aware adaptive samples upon a regular lattice mesh. Furthermore, particles can be used to create volumetric parcels that partition the computation domain into Voronoi cells [Sin et al. 2009] or power diagrams [De Goes et al. 2015].

Our proposed discretization strategy falls in the first category by enriching adaptivity on a regular grid structure. In particular, we aim to answer the question raised by the stringent regularity requirement for a Cartesian data structure: "Can we find another geometric primitive, or a set of combinatorial basis primitives, to enable a regular domain discretization that can naturally support adaptivity while preserving the structural regularity?" Inspired by methods from both (semi-)regular meshes such as a BCC and adaptive structured grids such as an Octree, we conducted preliminary research to find the mechanism of regular-and-adaptive discretization from a new perspective, by exploring the possibility of combining multiple (2 in our case) overlapping regular primitives to generate a new regular Eulerian grid discretization to support adaptive and dynamic tracking. As shown in Figure 3, the key mechanics of our adaptive staggered-tilted (AST) grid structure is to modify a uniform grid by placing an additional, tilted grid cell (e.g., rotated by 45-degree in 2D) on each grid node with locally varying cell size. When the size of a tilted cell is zero, the grid is locally unmodified; when the size of a tilted cell reaches its maximum, a new set of grid nodes (i.e., the intersection of tilted cells at edge center) is created, which allows another level of DoFs to emerge at those grid nodes.

Our AST grid structure is motivated by a thread of previous research on a rotated Cartesian grid [Edwards and Zheng 2008; van Es et al. 2014, 2016], pioneered by the foundational work [Saenger et al. 2000] and followed by a series of variations and modifications [Gao and Huang 2017; Liu and Sen 2009; Saenger and Bohlen 2004]. Most of these works were focused on capturing the anisotropic diffusion or wave propagation by establishing a higher-order differential stencil taking the additional grid nodes on the diagonal directions. In our case, we keep the design philosophy of these rotated staggered-grids by involving more degrees of freedom in the differential discretization. The defining features of our AST grid distinguishing itself from the previous ones consist in its staggered layout of the tilted DoFs on grid nodes and its adaptively controlled tilted cell status which is essential for dynamic feature tracking. Our grid structure can support large-scale simulations in both two and three-dimensional spaces, enabled by the software engineering practices to fit the grid structure into the various modern data structures such as Octree and parallel iterative solvers such as algebraic multigrid.

The proposed AST grid exhibits the following computational merits. First, the cell-tilting mechanics is simple to implement on a Cartesian grid, in particular, by overlapping two Cartesian grids with different grid origin and spacing. In this sense, our tilted grid structure can be seen as a plugin to an existing standard data structure to enable adaptivity in a straightforward fashion. Second, the tilted grid structure has excellent flexibility while being a regular discretization. We employ the cell status control on every tilted cell independently to enable fast and cheap adaptive operations in the computational domain. Last, this cell-tilting mechanism can be seamlessly integrated into a hierarchical data structure such as an Octree to eliminate the T-junctions. For example, an AST-modified Octree can recover orthogonality [Aanjaneya et al. 2017] across different levels by constraining the status of tilted cells in a local region.

We summarize our main contributions as:

- We present a new tilted-staggered grid structure composed of two sets of grid cells to enable adaptive simulation of large-scale incompressible flow, which opens up the possibility of creating new regular discretizations with combinatorial geometric primitives.
- We present a novel way for the dynamic tracking of complex Lagrangian features by switching the status of local tilted cells with negligible additional cost (around 1%).
- Our AST grid presents a computational paradigm to accommodate the traditional adaptive grid structure, such as an Octree, to preserve its discretization orthogonality and enhance its capability in controlling the adaptivity.

2 RELATED WORK

Structured methods. Computer graphics researchers invented a broad spectrum of regular grid structures targeting adaptive fluid simulation. The efforts on improving a Cartesian grid structure are solely motivated by its various appealing benefits regarding the computation performance, such as the cache-friendly accessing pattern

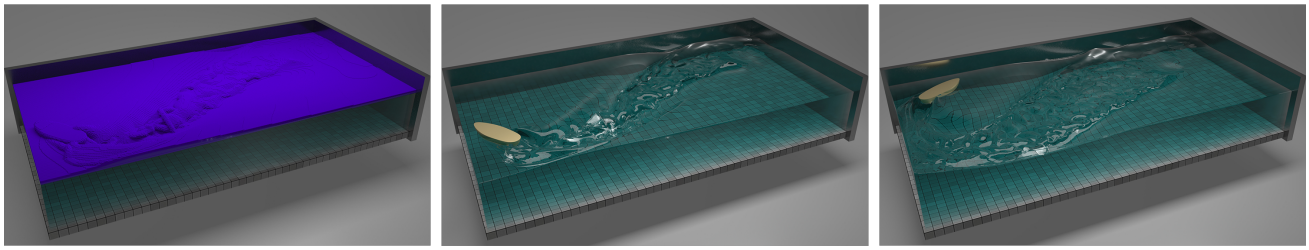


Fig. 2. Tilted cells dynamically evolve along with the boat and the wake to capture fine-scale details of fluid. The leftmost figure visualizes tilted cells. The effective horizontal resolution is 1200×600 and the narrow-band level set has horizontal resolution of 2400×1200.

and the easy parallelization for the modern hardware. Octree, as an enhancement to a uniform grid by splitting and deleting cells, is one of the most potent regular grid structure and has been well-studied over the past two decades. Originating from [Popinet 2003] and [Roussel et al. 2003], Losasso et al. [2004] proposed an Octree data structure accommodated by a set of symmetric differential operators to simulate incompressible flow adaptively. Losasso et al. [2006] improved their work to achieve second-order accuracy in pressure solving by a new gradient discretization over T-junctions. Setaluri et al. [2014] recovered optimal cache performance for Octree by introducing SPGrid, which reformulates the conventional pointer-based Octree to a pyramid of sparsely populated uniform grids. Nielsen et al. [2018] differed Setaluri for using velocities collocated at voxel corners, allowing sharper and faster interpolation. Aanjaneya et al. [2017] improved pressure projection accuracy on Octree by exploiting the power diagram to eliminate T-junctions in the cross-level region. Goldade et al. [2019] introduced an efficient viscosity solver for Octree with an adaptive variational finite-difference methodology. Ando et al. [2020] presented an easy-to-implement algorithm for high quality adaptive liquid simulation on Octree. Another class of regular adaptive grid is obtained through merging or stretching cells, forming stretched cell elements. For example, cells out of regions of interest are replaced with tall cells stretched in single-axis directions [Chentanez and Müller 2011; Irving et al. 2006] or are represented by compressed encodings [Houston et al. 2006]. Zhu et al. [2013] also employed stretched cells to achieve dimension-by-dimension adaptivity by translating entire grid lines. Aside from adjusting a Cartesian grid by applying operations on the unit of a cell, the curved grid line (curvilinear grid) [Azevedo and Oliveira 2013] was proposed for smoke simulation, maintaining grid regularity while achieving optimal alignment between grid and obstacle boundary. Multiple regular grids of different resolutions could also be grouped by rigidly translating and overlapping, forming a more descriptive grid structure (Chimera grid) [Cohen et al. 2010; Dobashi et al. 2008; English et al. 2013; Golas et al. 2012]. The spatially sparse nature of the fluid simulation task also drives researchers to propose more unified solutions, such as a high resolution sparse dynamic data structure (VDB) supporting $O(1)$ data access [Museth 2013], a computational framework for sparsely populated grid [Liu et al. 2018] and a data-oriented programming language for spatial sparse structure [Hu et al. 2019].

Unstructured and semi-structured methods. Mesh structures accommodate adaptive fluid simulation thanks to its inherent flexibility in allocating degrees of freedom in a computational domain. Feldman et al. [2005] adapted the stable fluid approach [Stam 1999] on the tetrahedral mesh, allowing underlying meshes to deform in an arbitrary Lagrangian-Eulerian (ALE) way that could be independent of the fluid motion. Klingner et al. [2006] evolved the discretized tetrahedron to align with the irregular boundaries and place more mesh elements in visually important regions. Wojtan and Turk [2008] proposed a tetrahedral finite element solver making use of the high-resolution surface mesh, capturing viscoelastic behavior with thin features. The unstructured tetrahedral mesh can also be advected with the fluid flow by maintaining a high-quality discretization with the help of a deformable simplicial complex [Misztal et al. 2010, 2012] or dynamic local remeshing [Clausen et al. 2013]. Though mesh-based methods are generally more natural to align with complex boundaries, their strength comes with the cost of an unstructured layout, which incurs non-trivial work related to explicit topological representation and thus suffers from poor memory footprint. To overcome the overhead of remeshing algorithm and data structure, the semi-structured lattice-based tetrahedral mesh is proposed for better memory storage and faster point location computation while maintaining mesh quality [Chentanez et al. 2007; Labelle and Shewchuk 2007; Teran et al. 2005]. Batty et al. [2010] extended these methods with embedded boundary techniques so that neither air nor solid boundaries need to align with the mesh in geometry, enabling simpler geometry and faster mesh generation. Ando et al. [2013] improved mesh discretization by placing velocity vectors at the barycenter of each cell and pressure values at each node. By taking advantage of sample points of fluid flow, Voronoi diagram has been used to construct the mesh on simulation particles [Sin et al. 2009] or to enable geometric-aware adaptive sampling on a background uniform mesh [Brochu et al. 2010]. Power diagram, a generalization of Voronoi diagram, can be used to treat fluid particles not only as material points but also as volumetric parcels, offering accurate pressure computation and removal of damping arising from the kernel-based evaluation [De Goes et al. 2015; Zhai et al. 2018]. These methods fall into the category of hybrid methods that benefit from both Eulerian and Lagrangian methods. More works such as [Nielsen and Bridson 2016] and material point methods (MPM) [Gao et al. 2018; Hu et al. 2018; Stomakhin et al. 2013]

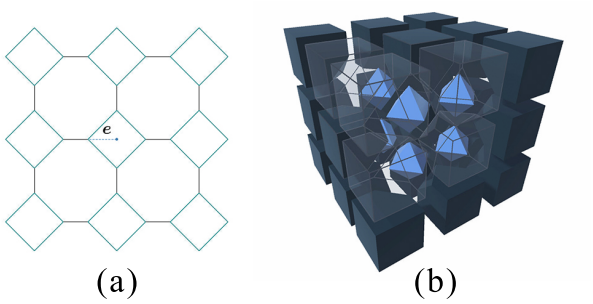
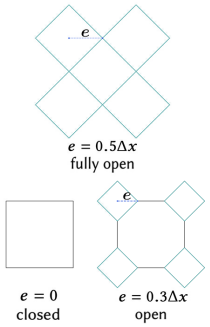


Fig. 3. (a) Setup of the AST grid in 2D. The 45-degree-tilted grid cells are placed at grid nodes of the uniform grid. (b) Setup of the AST grid in 3D. Tilted cells are rendered as blue octahedra, and they truncate the original cells of the uniform grid.

also demonstrate the balance between adaptivity and computational efficiency.

3 GRID STRUCTURE

Geometric primitives. We construct an AST grid by introducing two sets of grid cells – one set as the cells on a background uniform grid with its axes aligned with the world frame (the axis-aligned grid) and the other set as the cells from an overlapped grid with tilted axes in the world frame (the axis-tilted grid). As shown in Figure 3, the center of a tilted cell is overlapped with the original grid nodes, and the diagonals of a tilted cell are aligned with the original grid lines, exhibiting a *tilted-staggered* cell layout. We use half-diagonal length (e in the right figure) to measure the size of an axis-tilted cell, and it can vary continuously within the range of $[0, \frac{\Delta x}{2}]$, with Δx as the size of a uniform grid cell. According to the cell size e, we define three states of an axis-tilted cell: *closed*, *open*, and *fully open* shown in the figure inset. An axis-tilted cell is closed if its $e = 0$, in which case it is excluded from the structure and the local behavior of the grid will be identical to a uniform grid. An axis-tilted cell is open when $e \in (0, \frac{\Delta x}{2})$, in which case the axis-aligned cells and the axis-tilted cells co-exist. An axis-tilted cell is fully open when $e = \frac{\Delta x}{2}$, in which case the local primitives are fully tilted in 2D where the original axis-aligned cells of size Δx degenerate to another set of tilted cells. Practically, in most of our numerical examples (except the anisotropic test in Figure 12), we choose to restrict the control strategy for the tilted cells to be binary, which means a tilted cell can switch its status only between closed and fully open. At the same time, we also want to note that a continuous representation of the tilted cell size e can benefit the computation in different scenarios, e.g., anisotropic diffusion, by allowing the smooth flux transition along different axes (see Section 7). By dynamically switching cell states temporally, the AST grid enables us to design a host of control strategies to achieve different simulation goals (see Section 5). The axis-aligned cells and the axis-tilted cells together form a hybrid, tight spatial discretization,

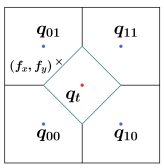


acting as two fundamental geometric primitives of the AST grid in a joint way. We use the resolution of the axis-aligned grid (i.e., the uniform grid) to denote the AST grid's resolution. Computationally, an AST grid has up to twice as many DoFs as the background uniform grid. For example, for an $n \times n$ uniform grid, its AST version adds $(n+1)^2$ extra adaptive DoFs along with the original fixed n^2 DoFs.

The domain discretization by the two sets of primitives is defined by the intersection of the two sets, removing the axis-aligned node underneath each tilted cell. In two spatial dimensions, each axis-tilted cell is adjacent to four axis-aligned cells neighbors. Each axis-aligned cell, which is an octagon, is adjacent to eight cells, including four original neighboring axis-aligned cells and additional four axis-tilted cells. In three spatial dimensions (see Figure 3b), the AST grid subdivides the space using a combination of equilateral octahedrons (axis-tilted cells) and truncated cubes (axis-aligned cells). Each axis-tilted cell has eight axis-aligned cells neighbors, and each axis-aligned cell (truncated cube) has extra eight neighbors of axis-tilted type in addition to original six axis-aligned neighbors.

Data storage and memory access. The data storage for the AST grid mainly resembles the one executed on a uniform Cartesian grid. Specifically, the quantities stored at tilted cells could be viewed as node-centered data in a standard Cartesian grid. We adopt the staggered discretization, i.e., the MAC grid [Harlow and Welch 1965], where scalar quantities are stored at the cell centers (both aligned and tilted cells), and vector fields (e.g., velocity) are stored at all faces. A linear sequential array is used to store scalar values at tilted cells. For vector fields, xyz-components at axis-aligned cells are stored in three separate memory blocks. Components at faces of a tilted cell are grouped into a 4-tuple (or 8-tuple in 3D), and all defined tuples are then packed into a continuous array to represent the vector field on tilted cells. The storage above supports us to access data points in an indexing manner. Therefore, to obtain neighboring cells (both tilted and axis-aligned cells) around a random point, one only needs to divide its position by cell size Δx and truncate in each dimension.

Interpolation. To support random point data sampling on an AST grid, we develop an interpolation scheme to include tilted cells. For scalar quantities (located at cell centers) interpolation on arbitrary position, we take the value stored on the tilted cell as a correction to the bilinear (or trilinear) interpolation operated on the background grid. The main idea follows the previous work of SPGrid [Setaluri et al. 2014], which also shares common features with [Ando et al. 2013; Brochu et al. 2010]. As shown in the figure inset, we start with a bilinear interpolation $q_f^* = \text{bilinear}(q_{00}, q_{01}, q_{10}, q_{11}, f_x, f_y)$, where (f_x, f_y) is the normalized coordinate of the sample point having range $[0, 1]$. If the tilted cell in the center is closed, we take q_f^* as the return value directly. In the case where we have an active tilted cell, which means it has a valid value, we proceed by computing a correction $\delta q = q_t - \frac{1}{4}(q_{00} + q_{01} + q_{10} + q_{11})$. And the final result of the interpolation is



$$q_f = q_f^* + 2\delta q \cdot min(f_x, 1 - f_x, f_y, 1 - f_y) \tag{1}$$

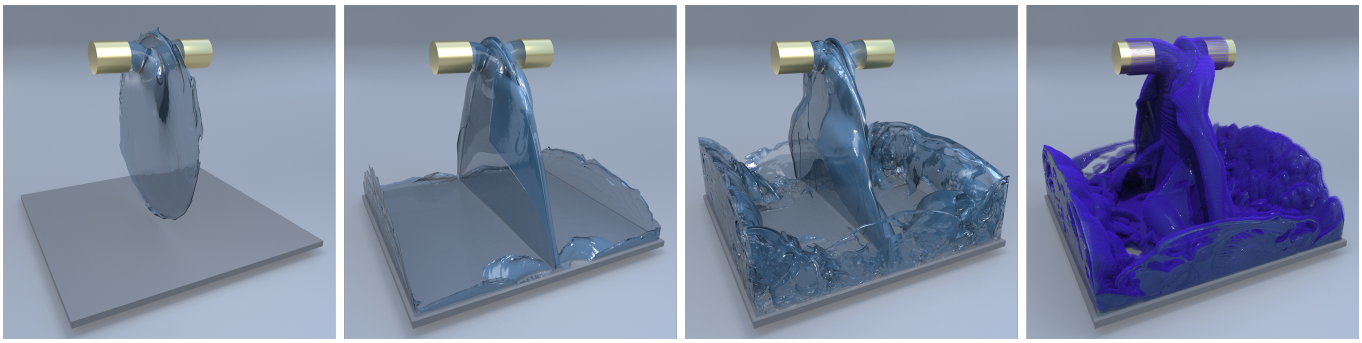


Fig. 4. Two sources of water inject into a basin, with 512^3 resolution and 1024^3 resolution for the narrow-band level set. The rightmost figure shows the tilted cell adaptivity.

The interpolation scheme can be naturally extended to 3D. Note that the computation of the index of a tilted cell is trivial because it offsets the index of point q_{00} by one, which is a known value in the bilinear interpolation pipeline. We show by experiments that the overhead of our interpolation scheme compared with a pure bilinear (or trilinear) interpolation is marginal (see Section 7.5).

For the vector quantity (e.g., velocity) interpolation, a naive way is to borrow the idea from the interpolation scheme on an unstructured mesh, by treating an AST grid as a mesh composed of a set of predefined primitives [Brochu et al. 2010; Chentanez et al. 2007]. On each cell (including both tilted and aligned), the vector components stored on faces are mapped back to the cell center to obtain an intermediate cell-centered vector field by solving a least-squares problem [Elcott et al. 2007]. This face-to-center mapping introduces numerical damping into the system. To alleviate this issue, we only map faces components of active tilted cells. The mapped vector values are used as a correction term rather than directly interpolated values, similar to our scalar interpolation. In particular, we first obtain the intermediate vector value $\vec{v}^*$ using the interpolation function I discretized on a uniform Cartesian grid. If the nearest tilted cell c_t is closed, then $\vec{v}^*$ will return as the final result directly. Otherwise, we compute $\delta\vec{v} = \vec{v}_{c_t} - I(P_{c_t})$ as a correction, where $\vec{v}_{c_t}$ is the remapped vector at the tilted cell c_t and P_{c_t} is the position of it. The final result is computed using $\vec{v}^*$ and the correction $\delta\vec{v}$ by Equation 1.

Relation to power diagram/BCC lattice. Besides having a straightforward geometric description, an AST grid can also be interpreted as a special case of a power diagram by taking the cell center as site and having the size of the tilted cell defined by weights. The topology of an AST grid also shares some similarities with the BCC lattice. Compared with Voronoi/power diagram based approaches [Brochu et al. 2010; De Goes et al. 2015; Sin et al. 2009] and BCC lattice based methods [Chentanez et al. 2007], our AST grid starts from a regular grid (Eulerian perspective), which promises a structured layout (both geometric and memory representation), allowing us to toggle DoFs dynamically during the simulation while avoiding the storage and the reconstruction of the mesh connectivity information. Aanjaneya et al. [2017] also applied this perspective, starting with a graded Octree. They embedded power diagrams only in the level-transition region of the Octree to recover primal-dual orthogonality, leaving the remaining part unchanged as a regular structured grid. In our case, we constrain the additional sites (i.e., tilted cells) to locate only at grid nodes to maintain the overall regularity of the grid, which also provides a way to eliminate T-junctions (see Section 6).

4 INCOMPRESSIBLE FLOW

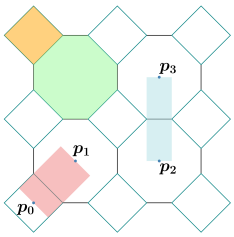
We solve the inviscid incompressible Navier-Stokes equations:

$$\begin{cases} \dfrac{\partial\vec{u}}{\partial t} + \vec{u}\cdot\nabla\vec{u} + \dfrac{1}{\rho}\nabla p = \vec{f} & (2) \\ \nabla\cdot\vec{u} = 0 & (3) \end{cases}$$

where $\vec{u}$ is velocity, ρ is density, p is pressure, and $\vec{f}$ is external forces. In each time step, the Equations above are solved in a time split fashion. First, the intermediate velocity $\vec{u}^*$ is computed by the semi-Lagrangian advection [Stam 1999], which is straightforward with the correction-based interpolation described in Section 3. Then, the body force is applied via $\vec{u}^{**} = \vec{u}^* + \Delta t\vec{f}$. The final velocity is computed by $\vec{u}^{n+1} = \vec{u}^{**} - \Delta t\nabla p/\rho$.

4.1 Discretized differential operators

Volume-weighted Gradient. As we have extra DoFs on tilted cells, we define gradient values on axis-tilted faces in addition to those on axis-aligned faces. The orthogonality between a cell face and the line connecting the centers of the two incident cells is guaranteed thanks to our cell tilting strategy, allowing us to build a volume-weighted gradient operator [Zhu et al. 2015]:



$$V\nabla p = V\frac{p_i - p_j}{l_{ij}}\vec{n} = (p_i - p_j)d\vec{A} \quad (4)$$

where V is the control volume (e.g., see pink and blue regions in the figure inset), l_{ij} is the distance between the centers of cell i and cell j, $\vec{n}$ is the unit face normal, and $d\vec{A}$ represents the area-weighted normal of the shared face between two cells.

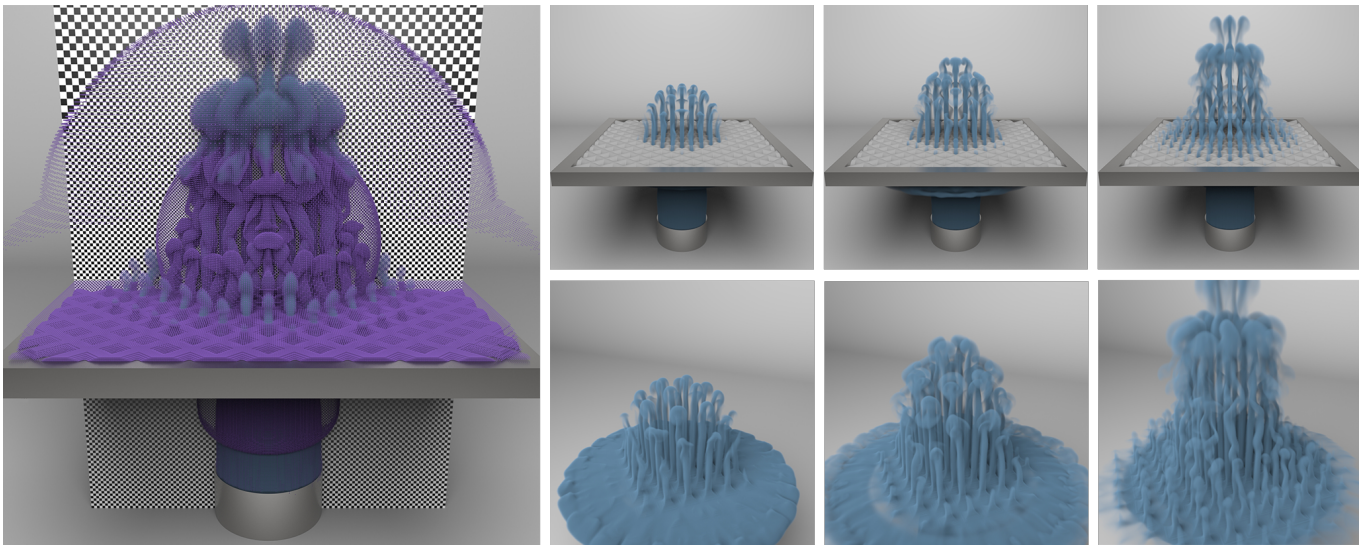


Fig. 5. Smoke plume passes through a solid porous grate using around 55 million voxels. The left figure shows the AST-modified Octree pattern and the tilted cell adaptivity. The right-bottom figures are the same frame sequence as the right-top figures but rendered in different camera setup and without the porous grate. Effective resolution is $512 \times 512 \times 768$.

Volume-weighted Divergence. Similar to the MAC grid [Harlow and Welch 1965], the divergence of a vector field is computed at the cell centers. Following the definition outlined in [Losasso et al. 2004], the discretized volume-weighted divergence formula can be written as

$$V_{cell} \nabla \cdot \vec{u} = \sum_{f \in F_c} \vec{u}_f \cdot d\vec{A} \tag{5}$$

where V_{cell} is the volume of the cell (shown as yellow and green regions in the above inset figure), F_c is the faces of the cell, $\vec{u}_f$ is the vector representation of the vector component containing direction information, and $d\vec{A}$ is the area-weighted normal pointing outwards. Note that the size of the tilted cell affects the area of faces, which provides a way to control flux through faces.

4.2 Poisson solver

Let's first consider a Poisson boundary value problem with spatially variable coefficients:

$$\begin{cases} \nabla \cdot \beta(\vec{x}) \nabla \phi(\vec{x}) = f(\vec{x}), & \vec{x} \in \Omega \\ \phi(\vec{x}) = g(\vec{x}), & \vec{x} \in \partial\Omega_D \\ \vec{n} \cdot \nabla \phi(\vec{x}) = h(\vec{x}), & \vec{x} \in \partial\Omega_N \end{cases} \tag{6}$$

where Ω denotes the interior domain, $\partial\Omega_D$ and $\partial\Omega_N$ represent the Dirichlet boundary and the Neumann boundary, and $\vec{n}$ is the unit normal pointing outwards. We follow the convention of a MAC grid discretization by defining $\phi(\vec{x})$, $g(\vec{x})$, and $f(\vec{x})$ at cell centers and $\beta(\vec{x})$ and $h(\vec{x})$ on faces. Equation 6 can be discretized on our grid structure as

$$(-\boldsymbol{G}^T)\boldsymbol{M}^{-1}\boldsymbol{G}\vec{\Phi} = \boldsymbol{V}_{cell}\vec{F} \tag{7}$$

where $\vec{\Phi}$ and $\vec{F}$ are vectors of ϕ and f values, $\boldsymbol{V}_{cell}$ is a diagonal matrix containing cell volume values, $\boldsymbol{G}$ and $-\boldsymbol{G}^T$ are the volume-weighted gradient and divergence operators, and $\boldsymbol{M}$ is a diagonal matrix assembled from the V/β values defined on faces. We enforce the boundary conditions the same way as on a uniform grid by modifying the right-hand side of the equation and the corresponding coefficients in the linear system on the left-hand side. The linear system defined in Equation 7 is sparse, symmetric, and positive-definite by construction. Such discretization enables a second-order accurate solving, as demonstrated by our convergence tests in Table 1.

Grid	L^1 Error	Order	L^∞ Error	Order
64×64	4.88×10^{-4}	-	1.08×10^{-3}	-
128×128	1.22×10^{-4}	2.00	2.73×10^{-4}	1.99
256×256	3.05×10^{-5}	2.00	6.87×10^{-5}	1.99
512×512	7.63×10^{-6}	2.00	1.72×10^{-5}	2.00
1024×1024	1.91×10^{-6}	2.00	4.31×10^{-6}	2.00

Table 1. Poisson solver accuracy on the AST grid. A Poisson equation with an analytic solution of $\phi(x, y) = sin(\pi x)sin(\pi y)$ is solved in the AST grid.

4.3 Pressure projection and free surface

To enforce incompressibility, we solve the Poisson equation of the form $\nabla \cdot \nabla p = \rho/\Delta t \nabla \cdot \vec{u}^{**}$ using the method above. We employ the level set method [Osher and Fedkiw 2003], ghost fluid method and cut-cell method [Batty et al. 2007, 2010; Enright et al. 2003; Gibou et al. 2002] to handle the free surface and solid boundaries on our grid structure. Surface tension is modeled as a pressure jump across the free surface boundary by enforcing a Dirichlet boundary condition $p_D = p_{air} + \gamma\kappa$ at the liquid-air interface, with γ as the surface tension coefficient and $\kappa = \nabla \cdot \nabla\phi$ as the local interface curvature, which is computed using a regular grid level set. A milk crown example is shown in Figure 6.

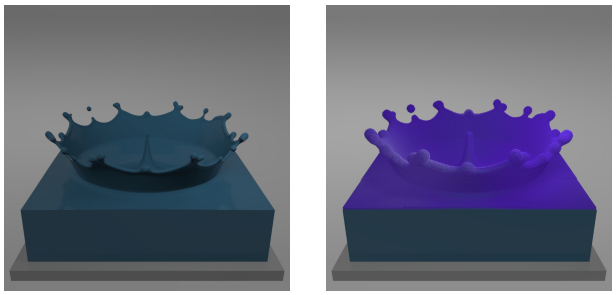


Fig. 6. Milk crown formation in the presence of surface tension, with the resolution of 256^3 and the tilted cell adaptivity. The right figure visualizes tilted cells.

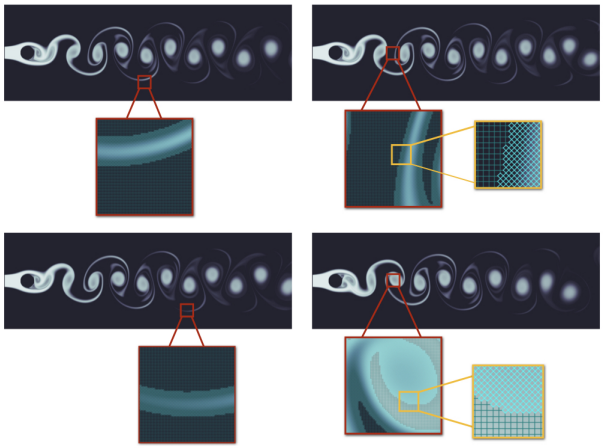


Fig. 7. Karman vortex street simulation. Left column figures visualize the status of tilted cells of the same fluid object at different times. Right column figures show the status of tilted cells of the same position at different times.

5 ADAPTIVITY AND DYNAMIC TRACKING

In this section, we present several strategies on an AST grid to enable adaptive simulation. The efficacy of these strategies is demonstrated by a set of simulation results in Section 7. In general, the spatial adaptivity on an AST grid is controlled by the local tilted cell states based on some specific policies, designed to track and capture flow feature dynamically during the course of the simulation.

Advective tilted-cell tracking. For the case of smoke simulation, the most basic dynamic tracking policy is based on density. As shown in Figure 7, we dynamically activate tilted cells following the density field along the Karman vortex street to provide locally refined discretizations. In particular, we open the tilted cells when the values of an advected scalar field that controls the local cell opening and closing exceed a predefined threshold. From the Lagrangian point of view, this adaptivity advection scheme amounts to the spatial continuous tracking of fluid material particles in an Eulerian way.

Narrowband-based tracking. Another tracking principle is to refine the grid near the regions of interest. In Figure 5 and 16, we initialize tilted cells in the vicinity of the boundary and keep them fully open while adjusting the remaining tilted cells based on the flow in a dynamic way. For liquid simulations shown in Figure 2, 4 and 8, the tilted cells evolve with the liquid free surface, which is straightforward to implement in the presence of the level set.

Vorticity tracking. As vorticity describes the local spinning motion of the fluid, we activate tilted cells in the region with high vorticity values. As shown in our two-dimensional smoke simulation comparison (Figure 14), the vorticity-based tracking captures the details of the rotational motion of a smoke plume by controlling the grid adaptivity based on the local vorticity.

Anisotropic tracking. The tilted cells can be adjusted to track global anisotropic features, which served as the original motivation of designing a rotated grid as in [Saenger et al. 2000; van Es et al. 2016]. The opened tilted cell introduces two (four in 3D) more axes besides the original xy axes (xyz axes in 3D), enabling better alignment of the grid faces with the characteristic flux. Continuous size control of the tilted cell further allows a smooth flux transition on different axes. We showcase the anisotropic tracking capability of our AST grid in the diffusion experiment (see Section 7.1).

Coarse-fine tracking. Integrating our AST grid with an existing adaptive data structure (e.g., Octree) can empower the tracking to have two different granularity (i.e., coarse tracking and fine tracking). The original adaptive structure provides a coarse-level control by updating its structure every several frames, and at the same time, the AST plugin provides a costless, fine-level control mechanism that can be updated in every time step with negligible overhead (around 1%). The example of plugging our AST discretization into a traditional Octree can be seen in Section 6.

Implementation. The status of the tilted cell scaling can be controlled either analytically or numerically. For an analytical control mechanism, the open or close status of the tilted cell (or cell size e) can be calculated on the fly without any memory storage using a predefined analytical function. For example, we could use the inequality $p_x^2 + p_y^2 < 1$ to fully open tilted cells inside the unit circle while closing all remaining cells. For a numerical control strategy, the status of the tilted cell is maintained in a separate memory block that is shared among different fields. When the cell control is a float operation, the block could be a sequential array of type float representing the size of every tilted cell. When the cell control is a binary operation, which is the case for our fluid examples, we exploit bitmap in our implementation to optimize memory overhead introduced by the grid structure. Including cached cell status (e.g., duplication at the axis-aligned cell to speed up computation), the overall memory overhead for recording grid topology is no more than 50MB in our fluid experiments. After updating the status of tilted cells, we still need to store the memory block both at time t_n and t_{n+1} in order to interpolate data for the new tilted setup in the advection step using semi-Lagrangian. As only a single memory block needs to be maintained to record cell status and the structured layout of the AST grid, the tracking process only adds around 1% overhead to a single step of simulation (see Table 3).

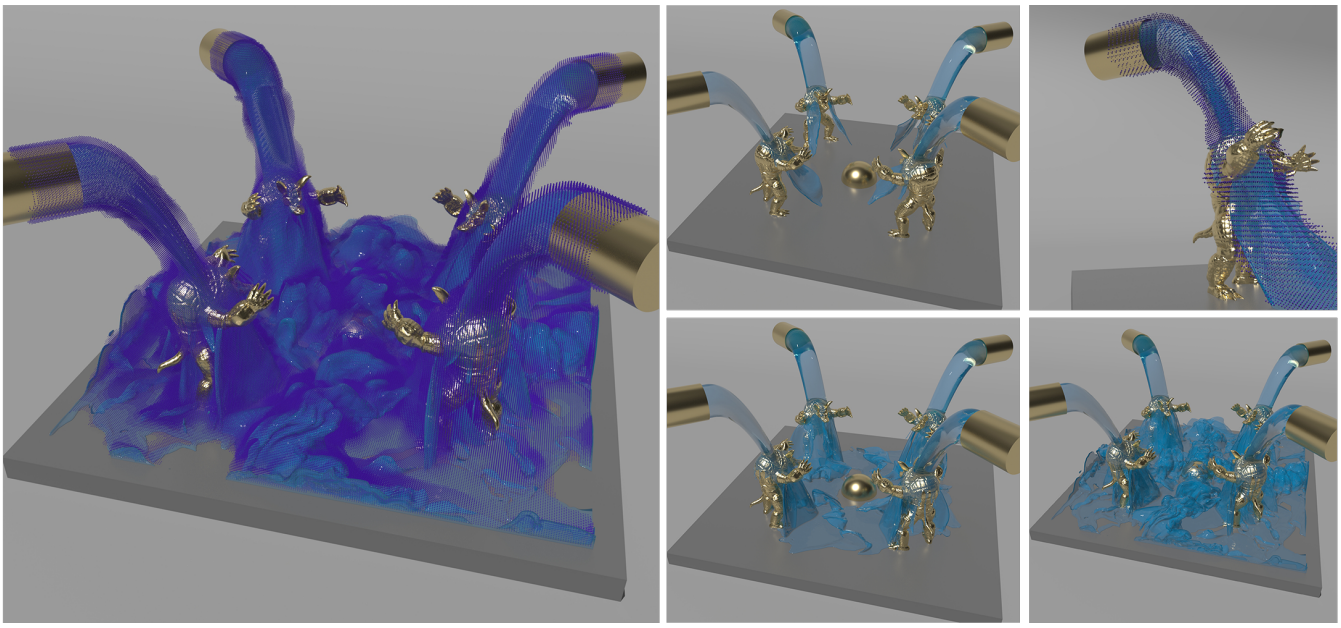
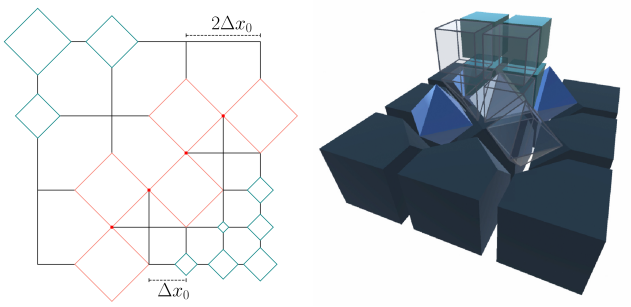


Fig. 8. Four sources pour liquid to four armadillo models with effective resolution 512^3. Tilted cells are opened in the vicinity of liquid free surface. A fine narrow-band level set with resolution 1536^3 is used for interface tracking.



(a) (b)

Fig. 9. Setup of the AST-modified Octree in 2D and 3D. (a) T-junctions (red dots) are naturally eliminated by maximized tilted cells (red tilted cells). (b) The half-tilted cell (white prism) is introduced to handle T-junctions.

6 OCTREE PLUGIN

The emergence of the tilted cell only relies on the existence of intersection nodes of orthogonal grid lines in the uniform grid. It is a fundamental characteristic of the uniform grid and can be found in many other regular grids, such as far-field grid [Zhu et al. 2013] and Octree [Losasso et al. 2004]. So the cell-tilting mechanism could be applied to *any* of these grid structures that exhibit regular spatial discretizations. We provide one example of marrying our AST grid with an Octree to showcase its ability to enable another level of adaptivity control and eliminate the T-junctions.

6.1 AST-Modified Octree

Similar to reformulation to the uniform grid, the AST grid modifies an Octree grid by placing tilted cells in every grid node of it. The topology of an AST-modified Octree is similar to the AST grid described in Section 3. The size of the tilted cell follows the same rules described in Section 3, but Δx, which was the uniform grid's cell size, now represents the size of the largest uniform cell adjacent to a tilted cell. Our AST grid can eliminate T-junctions on an Octree by adjusting the status of tilted cells. The strategy is simple: as shown in Figure 9a, tilted cells over T-junctions are constrained to be closed ($e = 0$), and the remaining tilted cells in the level-transition region are set to maximum size ($e = \Delta x_0$). This tilting strategy will lead to a mix of primitives, including a set of triangles connecting the truncated cells and tilted cells without the T-junction transition.

In three spatial dimensions, the maximized tilted cells alone are not sufficient for T-junctions elimination. We further introduce a square pyramid structure called half-tilted cell with its base placed on the face of the coarser-level cell and apex on the vertex of the neighboring finer-level cell, as illustrated in Figure 9b. The treatment of the half-tilted cell is almost the same as the tilted cell except for an extra flag recording the direction in implementation, which ensures the topology simplicity and solvability of the pressure projection.

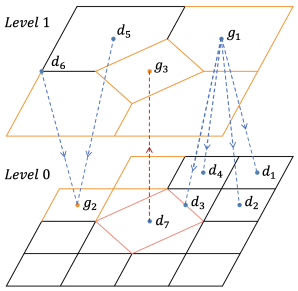
6.2 Linearized Storage

The main weakness of an Octree lies in its cache incoherency. We modify our data storage pattern introduced in Section 3 to fit an AST-modified Octree structure. As shown in Figure 10, data is stored in a linearized fashion, while the linearized index is computed by a hash table using the level and coordinate information of the cell as the key. We composite level and coordinate of a cell into a single 64-bit

integer, where the top 4 bits represent the level, and every following 20 bits represent the x, y, z coordinate, respectively. To enable data access locality, we divide the data layout into different layers for different Octree levels, and arrays allocated for each layer are stored separately. In each layer, cells are stored in a fashion similar to the uniform grid, where topologically adjacent cells are stored next to each other. With the linearized index and multi-layer storage, the AST-modified Octree preserves the cache coherent merits of a uniform grid on an Octree. For example, the addition of two fields (such as the force field and the velocity field) could be done directly for data arrays in each layer without calculating coordination. For more complex operations such as advection, where the position of cells is involved in computation, we keep a single copy of auxiliary data (e.g., coordination) to reduce the redundant access of the hash table.

6.3 Ghost Cell

Neighboring relation in level transition regions of the Octree is complex and computationally unfriendly. The introduction of tilted cells makes it even worse. For example, the interpolation scheme used in Section 3 would be ineffective because of the different data points layout. It is challenging to identify the neighboring cells in the level-transition region of an Octree if we choose the interpolation methods for the unstructured mesh. To solve this problem, we introduce a mechanism empowered by ghost cells that lie outside the boundary of valid cells in the transition region. Similar to [Setaluri et al. 2014], data of ghost cells are stored in the same way as valid cells within each level and are interpolated from adjacent levels.



The existence of tilted cells allows us to perform a precise average to get values at ghost cells, rather than extrapolation or approximation from known points nearby. The figure inset illustrates the ghost cell (draw in yellow color) update process:

(1) Borrow data from the adjacent level directly if the locations of data points coincide, e.g., $g_3 = d_7$.
(2) Perform a precise average of cells in the adjacent level using data stored on both tilted cells and axis-aligned cells, e.g., $g_1 = \frac{1}{4}(d_1 + d_2 + d_3 + d_4)$ and $g_2 = \frac{1}{2}(d_5 + d_6)$.

The structure of ghost cells turns cross-level interpolation and propagation into the interpolation within a single level. It ensures that the operations on a valid cell can always find either valid or ghost cell quantities within the same level and thus avoid the complex neighboring search. Also, the proposed interpolation scheme is still valid for an AST-modified Octree. As verified in our simulations (see Section 7), the interpolation and data acquisition methods for ghost cells successfully guarantee the data continuity across different levels of an AST-modified Octree, while reducing the implementation complexity significantly.

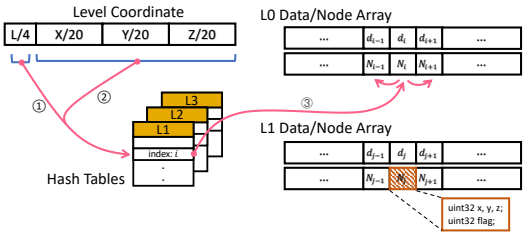


Fig. 10. Demonstration of the linearized storage for the AST-modified Octree. Each level has a hash table mapping the cell's level coordinate to its index (steps one and two), which is further used to fetch elements in the array (step three). Note that all data array and node array share the same index, which means map operations such as advection and addition of fields can be done without hash table lookup.

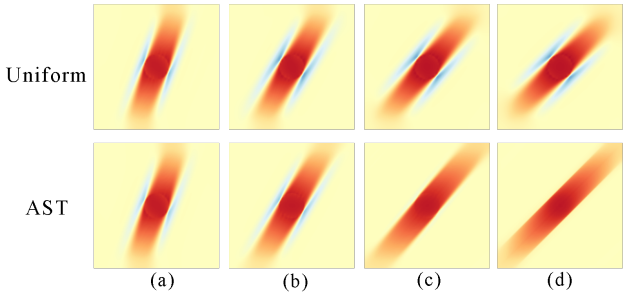
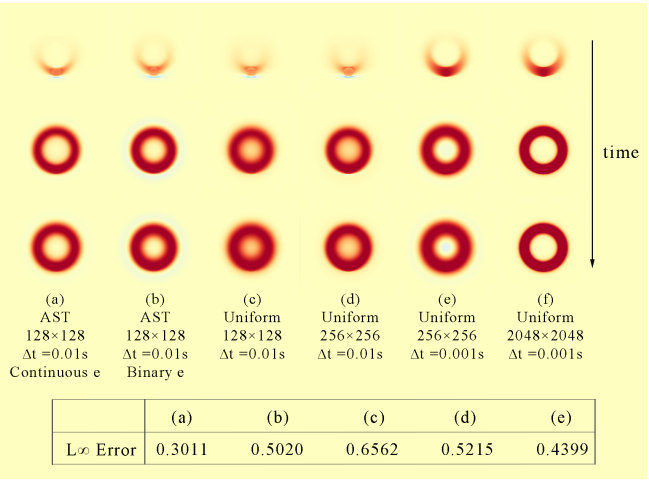


Fig. 11. Results of the directional anisotropic diffusion. From left to right, the columns correspond to diffusion directions of 20°, 30°, 40°, 45°, and the tilted cell sizes used in experiments are $0.01\frac{\Delta x}{2}$, $0.8\frac{\Delta x}{2}$, $0.99\frac{\Delta x}{2}$, $\frac{\Delta x}{2}$. In small diffusion angle cases, the benefits brought by the tilted cell are visually less observable as the diffusion direction is more aligned with the grid line of the uniform grid. The negative density value is visualized as blue and can be taken as a numerical error sourced from the supporting grid structure. The absolute summation of the negative value on the uniform grid is 892, 1515, 1288, and 1251 for each case, while our AST grid achieves a lower numerical error in all cases with value 820, 960, 90, and 0.

7 RESULTS

7.1 Anisotropic Diffusion

We conduct a numerical experiment to demonstrate a continuous representation of tilted cell size by preserving the diffusion features along the additional axes in our AST grid. We solve an anisotropic, generalized diffusion equation with the form $\frac{\partial q}{\partial t} = \nabla \cdot \boldsymbol{D} \nabla q(\vec{x}, t)$ on our AST grid, where q is a scalar diffusion quantity (e.g., heat) and $\boldsymbol{D}$ matrix is the collective diffusion coefficient defined by the strength of diffusion on principal axes (see Appendix A for the detailed formulation). The numerical error exists when there are misalignments between the principal axes and the axes of the underlying grid [van Es et al. 2016]. Our AST grid alleviates this issue by adaptively controlling non-axis-aligned tilted cells based on the local diffusion direction (i.e., principal axes). A continuously distributed tilted cell size e over the entire computational domain allows a smooth transition between zero-flux and full-flux on each axis. In the directional diffusion experiment (Figure 11), the domain is initialized with a



	(a)	(b)	(c)	(d)	(e)
L∞ Error	0.3011	0.5020	0.6562	0.5215	0.4399

Fig. 12. Results of the circular anisotropic diffusion with different underlying grids and setups. The figure shows diffusion results at different time frames (From top to bottom, $t = 0.02s, 1s, 2s$). The table shows the error computed against the high-resolution result (f).

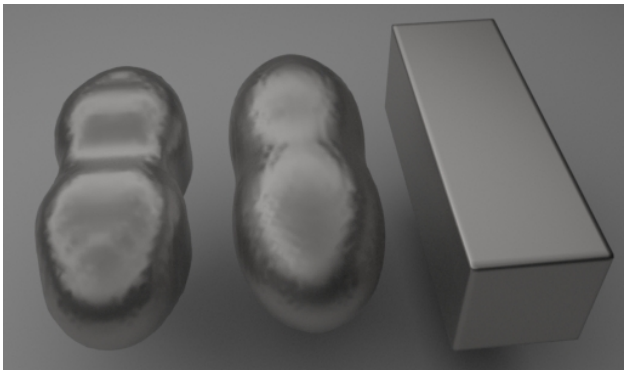


Fig. 13. Surface tension simulation of an initially block-shaped liquid. Objects from left to right are an AST grid using correction-based interpolation, an AST grid using interpolation of unstructured mesh, and the initial condition. Both examples have base resolution 64^3.

density value of one in the center and diffused in a specific direction. In the circular diffusion experiment (Figure 12), we define the diffusion direction of a given point as the tangent of the circle formed by the point and the domain center. As shown in both examples, the AST grid outperforms its uniform counterparts, and the continuous e setup reduces the error even further compared to the setup of the binary tilted cell states. All cases solve the same anisotropic diffusion equation in an explicit-implicit splitting manner [Rasmussen et al. 2004], which is suitable for the fast preconditioned conjugate gradient solver because the explicit integration needs an extremely small time step (e.g., $\Delta t = 0.0001s$) to remain stable.

7.2 Correction-based Interpolation

We validate the velocity interpolation scheme in Section 3 by showing a surface tension simulation with a level-set-based free surface in Figure 13. We compare the result from the correction-based scheme with the result from the face-to-center remapping scheme on the same AST grid. We show that the correction-based one can track the interface evolution and preserves the shape feature in a sharper way with the same grid resolution.

7.3 Visual Comparison

Figure 14 shows a set of 2D smoke simulations using the vorticity tracking strategy. We compared the results among a uniform grid, an AST grid, and a multi-level AST grid (Section 8). The additional number of DoFs we put on AST and multi-level AST are 11.3% and 12.3%, respectively, compared with the uniform grid with the coarsest resolution. The AST grid uses a two-layer hierarchy (aligned+tilted), and the multi-level AST grid uses a three-layer hierarchy (the third layer of adaptive cells are created on cell edges, as illustrated in Figure 14e). It is shown in Figure 14 that with a two-layer hierarchy (c) on our AST grid, we can obtain results comparable to the double-resolution uniform grid. With a three-layer hierarchy (d), but still at a very low computational cost, we can obtain the evolution of clear vortical structures developed from the refined AST regions, which was not possible even on a doubled-resolution uniform grid. Figure 15(a)-(e) compare 3D smoke simulations with different resolutions and initial conditions on a uniform grid and an AST grid. Our AST grid produces results with clearer and refined flow structures around the regions of interest, at the cost of around $20\% - 30\%$ increase of the total simulation time (Table 3), using around 10% additional tilted cells whose status are tracked based on the smoke density. We want to highlight that our AST grid can solve the visual artifact of axis-aligned smoke features and density gaps that have been existing as one of the major artifacts for Eulerian fluid simulation (as marked by red squares in the figure). These artifacts that are aligned to grid axis directions are due to the interpolation scheme on axis-aligned grid cells that are implemented for advection on a uniform grid. Our AST grid can naturally solve this problem by introducing additional DoFs in non-axis-aligned directions around the solid. Figure 15(f)(h) show another comparison with the same scene setup as Figure 5. The tilted cell is placed around the grate obstacle, which allows us to capture narrow flow passing through the porous grate without any staggered artifacts. Figure 15(i)(j) compare the two-level Octree and the AST-modified Octree. The simulation result obtained by the AST-modified Octree has more visual details, and is closer to the one generated on the uniform grid thanks to the recovered orthogonality. The Octree structure provides around 50% additional DoFs on the base resolution, and the tilted cell dynamically updates status to add around 10% more.

7.4 Examples

We demonstrate the effectiveness of our AST grid and its integration with Octree by simulating a number of examples. We illustrate our framework's dynamic tracking ability and adaptivity using two smoke examples to exemplify the interesting fluid flow features near a rigid body and solid porous boundaries. Figure 5 shows smoke flow travels through a solid porous grate, with three levels of Octree adaptivity. Figure 16 shows the smoke plume passes around a static metal sphere with a source at the bottom. We use tilted

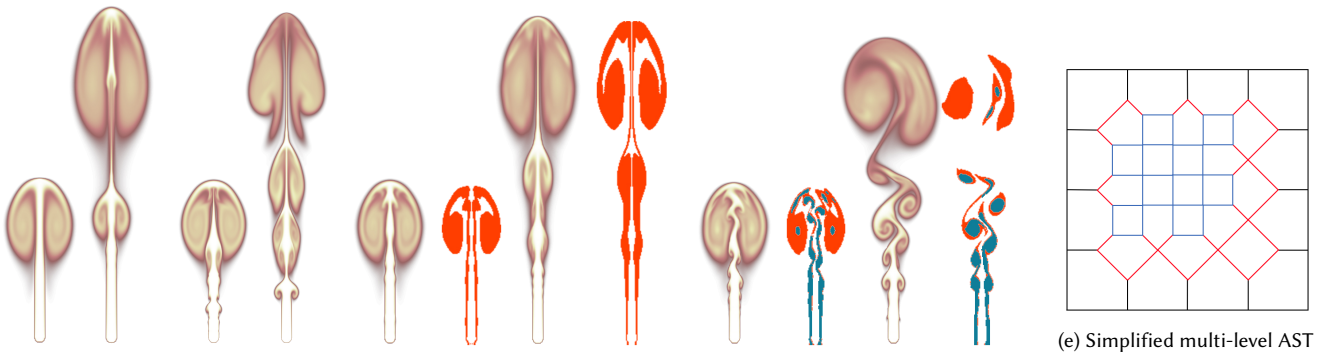


Fig. 14. 2D smoke simulation using different grid structures. The adaptive DoFs in images (c) and (d) are visualized beside the results, where tilted cells are rendered as red color, and edge nodes are denoted by dark blue color. (a)(b) Simulation result based on the uniform grid with two different resolutions. (c) Simulation result based on an AST grid. (d) Simulation result based on the simplified multi-level AST grid (Section 8). (e) Illustration of the simplified multi-level AST grid, where new cells (i.e., edge cells) are placed on the uniform grid's edge centers. It allows up to 4× DoFs increase on the background uniform grid.

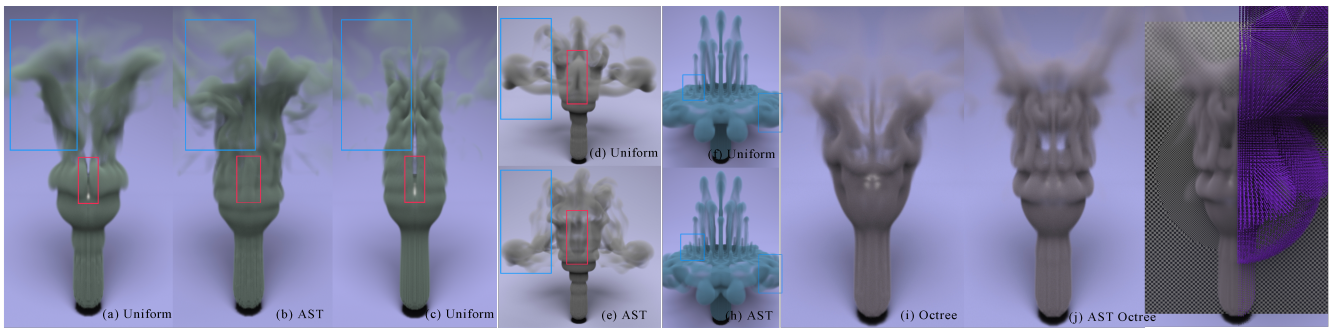


Fig. 15. Visual comparison of smoke simulations with different resolution and initial condition setup on the AST grid and its counterparts. (a)(b)(c) Simulation on the AST grid and the uniform grid with resolution $192 \times 192 \times 192$ (a)(b) and $242 \times 242 \times 242$ (c). (d)(e) Simulation with resolution of $256 \times 256 \times 256$. The time frame is different different from (a)(b), and the artifacts produced by the uniform grid are marked by red squares. (f)(h) Simulation with resolution of $256 \times 256 \times 256$ on the AST grid and the uniform grid. The porous grate is not rendered in the figure (refer Figure 5 for scene setup). (i)(j) Simulation on the Octree and the AST-modified Octree with base resolution of $128 \times 128 \times 128$. The Octree in both examples uses a two-level structure with around 50% additional DoFs placed around the sphere obstacle. The Octree structure pattern and the tilted cell adaptivity (12% of tilted cell is open in this time frame) are visualized in the rightmost figure.

adaptivity along with a four-level AST-modified Octree adaptive pattern in this example. In both examples, the Octree topology is finer near the flow and coarser away from it. The size of tilted cells maintains maximum within the narrow-band of the boundary and is dynamically updated based on the smoke density.

Examples of the liquid simulation are presented with different obstacle setups, where tilted cell adaptivity is applied to follow the liquid free surface and obstacle boundary. We use a regular grid level set with a higher resolution than the simulation grid to track and evolve the liquid free surface. Anonymous memory mapping [Setaluri et al. 2014] is used to store only the narrow-band region near the zero level set, accompanying with a bitmap marker to indicate inside/outside information for the whole domain. Figure 2 shows the dynamic adaptation of a scripted boat traveling in a still water. The tilted cells evolve dynamically along with the moving boat and the accompanying wake, where tilted cells within the narrow-band of the zero level set and the vicinity of obstacle boundary are opened. Figure 4 shows two sources of water pouring into a container, creating a thin sheet that wobbles as the instability accumulates. Figure 8 shows four sources pour the liquid onto four armadillo models in a container, exhibiting complex surface detail. Tilted cells are opened in the vicinity of the free surface and evolve along with the liquid. We use algebraic multigrid PCG solver and MICPCG solver in the projection step for our smoke examples and liquid examples. Table 2 shows the running time of a single time step for these examples. The liquid simulations use a serial level set reinitialization algorithm, which is the bottleneck in our implementation.

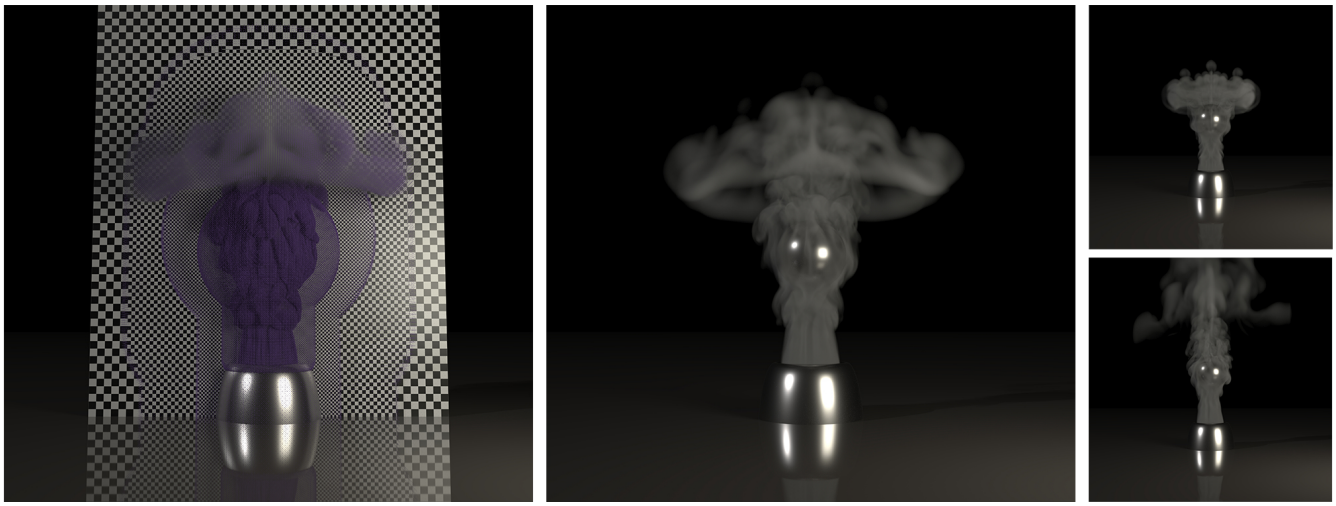


Fig. 16. Smoke plume passes around a static sphere using around 24 million voxels. The left figure shows the Octree adaptivity pattern and tilted cells, which reside in the smoke region to capture fine-scale details and the Octree level transition region to recover orthogonality. Effective resolution is $768 \times 768 \times 1024$.

7.5 Grid Overhead

The time overhead introduced by AST adaptivity control and the interpolation scheme is marginal, which is around 1% on average (see Table 2 and Table 3). It is achieved not only by the regular structure of the AST grid, which promises an efficient memory footprint and the random access ability but also by our well-performed dynamic tracking strategy. Those active tilted cells are placed at regions where DoFs are most needed thanks to our fine level adaptivity control. Our AST grid can be regarded as a continuous adaptive transition from a uniform grid with n^3 grid cells to a double-refined uniform grid with $2n^3$ grid cells. For an n^3 AST grid, the tilted cell adds another set of DoFs with the number of m (up to around n^3), and introduces $8m$ (up to $8n^3$) new faces to the existing $3n^3$ faces. From the *memory consumption per DoF* aspect, a tilted cell is more costly than a uniform cell because every active tilted cell needs to allocate eight faces compared with three for a uniform cell. However, if the inequality $3n^3 + 8m \leq 6n^3$ is satisfied, which describes the memory consumption of AST grid and the double-refined uniform grid, the AST grid outperforms the double-refined uniform grid with respect to the memory complexity. The inequality shows that at most 37.5% of the tilted grid cells can be open at the same time. We show that in practice, this number is a very generous upper bound for our simulation settings. For example, Figure 4 and Figure 5 open at most 10.3% and 6.3% of tilted cells, and Figure 16 has around 5% tilted cells opened.

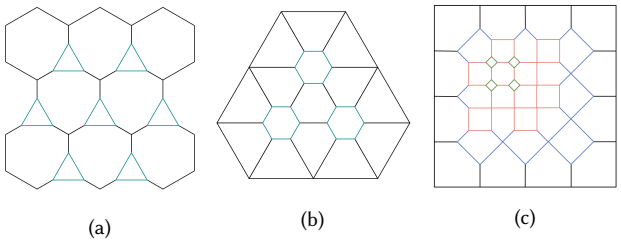


Fig. 17. (a) Triangle shape tilted cells grow on the regular honeycomb pattern. (b) Hexagon shape tilted cells grow on the regular triangle pattern. (c) The multi-level AST grid with recursively growing (axis-aligned or axis-tilted) cells.

	Fig 2	Fig 4	Fig 8	Fig 5	Fig 16
Advection	70	21	17	20	17
Projection	108	16	20	101	45
Level Set Reinitialization	146	50	42	-	-
Adaptivity Update Overhead	0.8%	1.7%	1.7%	1.1%	1.4%
Velocity Remapping Overhead	1.3%	1.7%	1.1%	1.4%	1.5%
Total	356	100	92	125	66

Table 2. Average timing breakdown (in seconds) for examples. Figure 5 was run on a machine with Intel Xeon 6136 at 3GHz, and the remaining examples were run on Intel Xeon E5-2620 v4 at 2.1GHz.

8 CONCLUSION, EXTENSION AND LIMITATIONS

We have presented the AST grid, an efficient data structure inheriting the merits of the Cartesian uniform grid while supporting adaptivity naturally. Our grid is quite flexible with respect to cell control. The capability of integrating with many regular grid structures makes the AST grid an effective plugin to enhance adaptivity control. We have shown the seamless combination of our tilted cells and Octree to remove the T-junctions. We demonstrate various simulations to illustrate that the AST grid structure can support the incompressible flow simulations with highly complicated features by providing flexible and dynamic tracking schemes with a small runtime overhead.

Grid	Resolution	DoFs Increase	Advection	Projection	Total	Adaptivity Update Overhead	Velocity Remapping Overhead
Uniform	192×384	1.0	1	1	1	-	-
AST	192×384	1.117	1.14	1.21	1.22	1.2%	1.1%
Simplified Multi-level AST	192×384	1.123	1.32	1.26	1.34	3.9%	1.4%
Uniform	272×543	2.0	1.54	1.93	1.86	-	-
Uniform	384×768	4.0	2.76	3.32	3.21	-	-
Uniform	$192 \times 192 \times 192$	1.0	1.0	1.0	1.0	-	-
AST	$192 \times 192 \times 192$	1.08	1.31	1.24	1.25	0.78%	0.33%
Uniform	$242 \times 242 \times 242$	2.01	2.14	1.84	1.88	-	-
Uniform	$256 \times 256 \times 256$	1.0	1.0	1.0	1.0	-	-
AST	$256 \times 256 \times 256$	1.08	1.39	1.32	1.33	0.76%	0.32%
Uniform	$323 \times 323 \times 323$	2.0	2.38	2.01	2.04	-	-

Table 3. Timing data of simulations in Figure 14 and Figure 15(a-e). Timings are given as the ratio required of a simulation on the uniform grid with base resolution. All simulations ran at a single machine with 16 threads (Intel Xeon E5-2620 v4 at 2.10GHz). The numbers in the DoFs increase column are given at a specific frame because DoFs changes during simulation due to our dynamic tracking algorithm. Overhead of AST adaptivity update includes the tilted cell (and the edge cell in multi-level AST) status update (i.e., dynamic tracking) and cached data update. Both AST adaptivity update and velocity remapping overheads are computed against the total time of a single step simulation.

Multi-level AST Grid. The proposed adaptive staggered-tilted grid provides a new adaptive-regular discretization perspective and can be successfully applied to the fluid simulation pipeline to offer both dynamic control and efficient computation. Though the AST grid is defined as the overlapping of two sets of cells for an implementation easiness, the tilted cell can also be viewed as a dynamic scaling process with a tilted cell emerging and growing from a node of the background uniform grid. Following this idea, we find that our AST grid can be naturally extended to a multi-level setting in two dimensions (see Figure 14e). New grid cells (i.e., edge cell) can be placed at the center of grid edges, where new grid nodes are formed by the tilted cell intersections. To validate its effectiveness, we implement a simplified version with up to four times more DoFs, and compare it against the AST grid and the uniform grid (Figure 14). The implementation uses the same memory layout (i.e., linear storage) with the AST grid and bilinear interpolation for regions where edge cells are open. The result suggests the multi-level AST grid is a promising structure and deserves further study (e.g., real multi-level AST grid and 3D version multi-level AST structure). Moreover, it is interesting that the tilted cell can also grow on different types of regular patterns, such as a regular triangle discretization and a honeycomb pattern, leading to the shapes combining a regular hexagon and a regular triangle (see Figure17a and 17b). It suggests that the majority of regular patterns have the potential to enable adaptive control and alleviate the strong regular constraint.

Limitations. The proposed AST grid structure also incurs several limitations. First, the default AST implementation provides two levels of adaptivity only, which can be insufficient for some applications. We demonstrated a simplified multi-level AST implementation providing up to four times DoFs, and more level of adaptivity can be achieved recursively (see Figure 17c). Although the current hierarchical scheme applies to 2D only, we envision its potential applications in other physics simulation settings, such as for adaptive cloth or interfacial flow phenomena. We also remedy the limited adaptivity by integrating the AST grid with an existing adaptive regular pattern (i.e., an Octree), forming a new regular pattern with coarse and fine level adaptivity tracking. Second, the extra axes introduced in the AST grid are currently fixed directions to keep the overall orthogonality of the grid structure, limiting the grid's potential to fully align with obstacles or anisotropic features. One possible solution to this is sacrificing orthogonality out of regions of interest, thus providing more alignment control. It is a challenging and exciting future work since arbitrary alignment is currently only possible with mesh-based schemes. Third, the presented AST-modified Octree does not allow a neighboring relation among cells with the level difference more than one. However, this constraint is satisfied in most Octree setups and can be ensured by introducing some transition levels when it is violated.

A ANISOTROPIC DIFFUSION ON AST GRID

The generalized diffusion equation is defined as follows:

$$\frac{\partial q}{\partial t} = \nabla \cdot \boldsymbol{D} \nabla q(\vec{x}, t) \tag{8}$$

where q is a scalar diffusion quantity (e.g., mass, energy, heat, etc.), and $\boldsymbol{D}$ is the collective diffusion coefficient. In a physical diffusion problem, $\boldsymbol{D}$ tensor is defined by:

$$\boldsymbol{D} = \boldsymbol{Q} \boldsymbol{\Lambda} \boldsymbol{Q}^T \tag{9}$$

$$\boldsymbol{Q} = [\vec{v}_1, \vec{v}_2], \boldsymbol{\Lambda} = \begin{bmatrix} D_1 & 0 \\ 0 & D_2 \end{bmatrix} \tag{10}$$

where $\vec{v}_1, \vec{v}_2$ are orthogonal axes (also known as principal axes), and D_1, D_2 are diffusion coefficients describing the strength of diffusion in these two axes. The diffusion process is isotropic when $\boldsymbol{D}$ is diagonal and anisotropic when $\boldsymbol{D}$ is a general symmetric matrix. Normally, numerical error exists when there are misalignments between principal axes and the grid line of the underlying grid

structure. We solve the above anisotropic diffusion problem in an explicit-implicit splitting manner [Rasmussen et al. 2004], where the $\boldsymbol{D}$ tensor is divided into a non-diagonal matrix $\boldsymbol{D}_N$ and a diagonal matrix $\boldsymbol{D}_D$. To discretize the equation on the tilted cell, we need to transform $\boldsymbol{D}$ to follow the tilted axes. We denoted the transformed tensor as $\boldsymbol{D}^{'}$, and it is obtained by $\boldsymbol{D}^{'} = \boldsymbol{R}^T\boldsymbol{D}\boldsymbol{R}$, where $\boldsymbol{R}$ is a matrix representing the rotation from the xy axes of the uniform grid to the tilted axes.

In our anisotropic experiment (Section 7.1), we constrain one of the diffusion coefficients to zero, and the other equals one, thus having an extreme anisotropic setup. To capture this anisotropy in the AST grid, we adjust the tilted cell status according to the principal axes. In the directional diffusion experiment (Figure 11), the tilted cell size is chosen heuristically to minimize the numerical error. In the circular diffusion experiment (Figure 12), the tilted cell size e is equal to $cos(2\theta)\frac{\Delta x}{2}$, where θ has the range $[0, \frac{\pi}{4}]$ representing the minimal angle between tilted axes and principal axes. This simple yet effective rule allows the tilted cell to be fully open when the diffusion direction is aligned with the tilted axes, and to smoothly transit to fully closed as the diffusion direction diverges.

ACKNOWLEDGMENTS

We thank all the reviewers for their valuable comments. SJTU (Shanghai Jiao Tong University) authors acknowledge the funding support from the National Key Research and Development Program of China (2018YFB1004902) and the National Natural Science Foundation of China (61772329, 61373085). Dartmouth author acknowledges the funding support from Neukom Institute CompX Faculty Grant, Burke Research Initiation Award, Toyota TEMA North America Inc, and ByteDance Inc. We credit the Houdini Education licenses for the video generations.

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 (July 2017).

Ryoichi Ando and Christopher Batty. 2020. A practical octree liquid simulator with adaptive surface resolution. *ACM Transactions on Graphics* 39, 4 (jul 2020). 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

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 (2007), 100. https://doi.org/10.1145/1276377.1276502

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

Tyson Brochu, Christopher Batty, and Robert Bridson. 2010. Matching Fluid Simulation Elements to Surface Geometry and Topology. In *ACM SIGGRAPH 2010 Papers (SIGGRAPH '10)*. ACM, New York, NY, USA, Article 47, 9 pages. https://doi.org/10.1145/1833349.1778784

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

Nuttapong Chentanez and Matthias Müller. 2011. Real-time Eulerian Water Simulation Using a Restricted Tall Cell Grid. In *ACM SIGGRAPH 2011 Papers (SIGGRAPH '11)*. ACM, New York, NY, USA, Article 82, 10 pages. https://doi.org/10.1145/1964921.1964977

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 (April 2013), 17:1–15. https://doi.org/0.1145/2451236.2451243 Presented at SIGGRAPH 2013.

Jonathan M Cohen, Sarah Tariq, and Simon Green. 2010. Interactive fluid-particle simulation using translating Eulerian grids. In *Proceedings of I3D 2010: The 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. 15–22. https://doi.org/10.1145/1730804.1730807

Fernando De Goes, Corentin Wallez, Jin Huang, Dmitry Pavlov, and Mathieu Desbrun. 2015. Power particles: An incompressible fluid solver based on power diagrams. *ACM Transactions on Graphics* 34, 4 (2015). https://doi.org/10.1145/2766901

Yoshinori Dobashi, Yasuhiro Matsuda, Tsuyoshi Yamamoto, and Tomoyuki Nishita. 2008. A Fast Simulation Method Using Overlapping Grids for Interactions between Smoke and Rigid Objects. *Comput. Graph. Forum* 27 (04 2008), 477–486. https://doi.org/10.1111/j.1467-8659.2008.01145.x

Michael G. Edwards and Hongwen Zheng. 2008. A quasi-positive family of continuous Darcy-flux finite-volume schemes with full pressure support. *J. Comput. Phys.* 227, 22 (2008), 9333 – 9364. https://doi.org/10.1016/j.jcp.2008.05.028

Sharif Elcott, Yiying Tong, Eva Kanso, Peter Schröder, and Mathieu Desbrun. 2007. Stable, Circulation-preserving, Simplicial Fluids. *ACM Trans. Graph.* 26, 1, Article 4 (Jan. 2007). https://doi.org/10.1145/1189762.1189766

R. Elliot English, Linhai Qiu, Yue Yu, and Ronald Fedkiw. 2013. Chimera grids for water simulation. In *Proceedings - SCA 2013: 12th ACM SIGGRAPH / Eurographics Symposium on Computer Animation*. 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 *In Proc. 4th ASME-JSME Joint Fluids Eng. Conf., number FEDSM2003-45144. ASME*. 2003–45144.

Bryan E. Feldman, James F. O'Brien, Bryan M. Klingner, and Tolga G. Goktekin. 2005. Fluids in Deforming Meshes. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation 2005*. http://graphics.berkeley.edu/papers/Feldman-FDM-2005-07/

Kai Gao and Lianjie Huang. 2017. An improved rotated staggered-grid finite-difference method with fourth-order temporal accuracy for elastic-wave modeling in anisotropic media. *J. Comput. Phys.* 350 (2017), 361–386. https://doi.org/10.1016/j.jcp.2017.08.053

Ming Gao, Xinlei Wang, Kui Wu, Andre Pradhana, Eftychios Sifakis, Cem Yuksel, and Chenfanfu Jiang. 2018. GPU optimization of material point methods. *SIGGRAPH Asia 2018 Technical Papers, SIGGRAPH Asia 2018* 37, 6 (2018). https://doi.org/10.1145/3272127.3275044

Frederic Gibou, Ronald P. Fedkiw, Li-Tien Cheng, and Myungjoo Kang. 2002. A Second-order-accurate Symmetric Discretization of the Poisson Equation on Irregular Domains. *J. Comput. Phys.* 176, 1 (Feb. 2002), 205–227. https://doi.org/10.1006/jcph.2001.6977

Abhinav Golas, Rahul Narain, Jason Sewall, Pavel Krajcevski, Pradeep Dubey, and Ming Lin. 2012. Large-scale fluid simulation using velocity-vorticity domain decomposition. In *ACM Transactions on Graphics*, Vol. 31. https://doi.org/10.1145/2366145.2366167

Ryan Goldade, Yipeng Wang, Mridul Aanjaneya, and Christopher Batty. 2019. An adaptive variational finite difference framework for efficient symmetric octree viscosity. *ACM Transactions on Graphics* 38, 4 (2019), 1–14. https://doi.org/10.1145/3306346.3322939

F. H. Harlow and J. E. Welch. 1965. Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface. *Physics of Fluids* 8 (Dec. 1965), 2182–2189. https://doi.org/10.1063/1.1761178

Ben Houston, Michael B. Nielsen, Christopher Batty, Ola Nilsson, and Ken Museth. 2006. Hierarchical RLE level set. *ACM Transactions on Graphics* 25, 1 (2006), 151–175. https://doi.org/10.1145/1122501.1122508

Yuanming Hu, Mit Csail, Tzu-Mao Li, U C Berkeley, Luke Anderson, Jonathan Ragan-Kelley, Berkeley Frédo Durand, and Frédo Durand. 2019. Taichi: A Language for High-Performance Computation on Spatially Sparse Data Structures. *ACM Trans. Graph* 38, 6 (2019), 16. https://doi.org/10.1145/3355089.3356506

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* 37, 4 (jul 2018), 1–14. https://doi.org/10.1145/3197517.3201293

H. Ibayashi, C. Wojtan, N. Thuerey, T. Igarashi, and R. Ando. 2018. Simulating Liquids on Dynamically Warping Grids. *IEEE Transactions on Visualization and Computer Graphics* (2018), 1–1. 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. In *ACM SIGGRAPH 2006 Papers (SIGGRAPH '06)*. ACM, New York, NY, USA, 805–811. https://doi.org/10.1145/1179352.1141959

Bryan M. Klingner, Bryan E. Feldman, Nuttapong Chentanez, and James F. O'Brien. 2006. Fluid Animation with Dynamic Meshes. In *ACM SIGGRAPH 2006 Papers (SIGGRAPH '06)*. ACM, New York, NY, USA, 820–825. https://doi.org/10.1145/1179352.1141961

François Labelle and Jonathan Richard Shewchuk. 2007. Isosurface stuffing: Fast tetrahedral meshes with good dihedral angles. *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics* (2007), 1–10. https://doi.org/10.1145/1275808.1276448

Haixiang Liu, Yuanming Hu, Bo Zhu, Wojciech Matusik, and Eftychios Sifakis. 2018. Narrow-band topology optimization on a sparsely populated grid. *SIGGRAPH Asia 2018 Technical Papers, SIGGRAPH Asia 2018* 37, 6 (2018), 1–14. https://doi.org/10.1145/3272127.3275012

Yang Liu and Mrinal K. Sen. 2009. A new time space domain high-order finite-difference method for the acoustic wave equation. *J. Comput. Phys.* 228, 23 (2009), 8779 – 8806. https://doi.org/10.1016/j.jcp.2009.08.027

Frank Losasso, Ronald Fedkiw, and Stanley Osher. 2006. Spatially adaptive techniques for level set methods and incompressible flow. *Computers and 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. In *ACM SIGGRAPH 2004 Papers (SIGGRAPH '04)*. ACM, New York, NY, USA, 457–462. https://doi.org/10.1145/1186562.1015745

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" (2010)*, Kenny Erleben, Jan Bender, and Matthias Teschner (Eds.). The Eurographics Association. https://doi.org/10.2312/PE/vriphys/vriphys10/011-020

M. K. Misztal, K. Erleben, A. Bargteil, J. Fursund, B. Bunch Christensen, J. A. Bærentzen, and R. Bridson. 2012. Multiphase flow of immiscible fluids on unstructured moving meshes. *Computer Animation 2012 - ACM SIGGRAPH / Eurographics Symposium Proceedings, SCA 2012* (2012), 97–106. https://doi.org/10.2312/SCA/SCA12/097-106

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

Michael B. Nielsen and Robert Bridson. 2016. Spatially adaptive FLIP fluid simulations in bifrost,robert bridson. In *SIGGRAPH 2016 - ACM SIGGRAPH 2016 Talks*. ACM Press, New York, New York, USA, 1–2. https://doi.org/10.1145/2897839.2927399

Michael B. Nielsen, Konstantinos Stamatelos, Morten Bojsen-Hansen, Duncan Brinsmead, Yannick Pomerleau, Marcus Nordenstam, and Robert Bridson. 2018. A collocated spatially adaptive approach to smoke simulation in bifrost. (2018), 1–2. https://doi.org/10.1145/3214745.3214749

Stanley Osher and Ronald Fedkiw. 2003. *Level Set Methods and Dynamic Implicit Surfaces*. Applied Mathematical Sciences, Vol. 153. Springer New York, New York, NY. https://doi.org/10.1007/b98879

Stephane 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

N. Rasmussen, D. Enright, D. Nguyen, S. Marino, N. Sumner, W. Geiger, S. Hoon, and R. Fedkiw. 2004. Directable Photorealistic Liquids. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '04)*. Eurographics Association, Goslar Germany, Germany, 193–202. https://doi.org/10.1145/1028523.1028549

Olivier Roussel, Kai Schneider, Alexei Tsigulin, and Henning Bockhorn. 2003. A conservative fully adaptive multiresolution algorithm for parabolic PDEs. *J. Comput. Phys.* 188, 2 (2003), 493 – 523. https://doi.org/10.1016/S0021-9991(03)00189-X

Erik H. Saenger and Thomas Bohlen. 2004. Finite difference modeling of viscoelastic and anisotropic wave propagation using the rotated staggered grid. *GEOPHYSICS* 69, 2 (2004), 583–591. https://doi.org/10.1190/1.1707078 arXiv:https://doi.org/10.1190/1.1707078

Erik H. Saenger, Norbert Gold, and Serge A. Shapiro. 2000. Modeling the propagation of elastic waves using a modified finite-difference grid. *Wave Motion* 31, 1 (2000), 77–92. https://doi.org/10.1016/S0165-2125(99)00023-2

Rajsekhar Setaluri, Mridul Aanjaneya, Sean Bauer, and Eftychios Sifakis. 2014. SPGrid: A sparse paged grid structure applied to adaptive smoke simulation. *ACM Transactions on Graphics (TOG)* 33, 6 (2014), 205.

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 (SCA '09)*. ACM, New York, NY, USA, 247–255. https://doi.org/10.1145/1599470.1599502

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., New York, NY, USA, 121–128. https://doi.org/10.1145/311535.311548

Alexey Stomakhin, Craig Schroeder, Lawrence Chai, Joseph Teran, and Andrew Selle. 2013. A material point method for snow simulation. *ACM Transactions on Graphics* 32, 4 (jul 2013), 1. https://doi.org/10.1145/2461912.2461948

J. Teran, Neil Molino, R. Fedkiw, and R. Bridson. 2005. Adaptive physics based tetrahedral mesh generation using level sets. *Engineering with Computers* 21, 1 (2005), 2–18. https://doi.org/10.1007/s00366-005-0308-8

Bram van Es, Barry Koren, and Hugo J. de Blank. 2014. Finite-difference schemes for anisotropic diffusion. *J. Comput. Phys.* 272 (2014), 526 – 549. https://doi.org/10.1016/j.jcp.2014.04.046

Bram van Es, Barry Koren, and Hugo J. de Blank. 2016. Finite-volume Scheme for Anisotropic Diffusion. *J. Comput. Phys.* 306, C (Feb. 2016), 422–442. https://doi.org/10.1016/j.jcp.2015.11.041

Chris Wojtan and Greg Turk. 2008. Fast viscoelastic behavior with thin features. *ACM Transactions on Graphics* 27, 3 (2008). https://doi.org/10.1145/1360612.1360646

Xiao Zhai, Fei Hou, Hong Qin, and Aimin Hao. 2018. Fluid Simulation with Adaptive Staggered Power Particles on GPUs. *IEEE Transactions on Visualization and Computer Graphics* (2018), 1–14. https://doi.org/10.1109/TVCG.2018.2886322

Bo Zhu, Minjae Lee, Ed Quigley, and Ronald Fedkiw. 2015. Codimensional non-Newtonian Fluids. *ACM Trans. Graph.* 34, 4, Article 115 (July 2015), 9 pages. https://doi.org/10.1145/2766981

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