A New Grid Structure for Domain Extension

Bo Zhu* Stanford University
Wenlong Lu* Stanford University
Matthew Cong* Stanford University, Industrial Light + Magic
Byungmoon Kim† Adobe Systems Inc.
Ronald Fedkiw* Stanford University, Industrial Light + Magic

Figure 1: *Our far-field grid structure provides an extended domain for fluid simulations of smoke, fire, and water. (Left) A fine grid follows the sphere in order to resolve the fine scale details of the smoke due to object interaction while the extended grid allows the smoke to rise until it is off camera – see Figure 5. (Center) A torch that moves through a large extent of the domain uses a fine grid to track the torch motion while grid extension allows for larger camera angles – see Figure 6. (Right) A large advantage of our extended grid is that it allows outgoing waves to avoid reflecting off of grid boundaries thus allowing for a large amount of detail and grid resolution near the region of interest without reflected waves.*

Abstract

We present an efficient grid structure that extends a uniform grid to create a significantly larger far-field grid by dynamically extending the cells surrounding a fine uniform grid while still maintaining fine resolution about the regions of interest. The far-field grid preserves almost every computational advantage of uniform grids including cache coherency, regular subdivisions for parallelization, simple data layout, the existence of efficient numerical discretizations and algorithms for solving partial differential equations, etc. This allows fluid simulations to cover large domains that are often infeasible to enclose with sufficient resolution using a uniform grid, while still effectively capturing fine scale details in regions of interest using dynamic adaptivity.

CR Categories: I.3.3 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

Keywords: fluid simulation, grids, boundary conditions

Links: DL PDF

*e-mail: {boolzhu,wenlongl,mdcong,fedkiw}@cs.stanford.edu
†e-mail: bmkim@adobe.com

1 Introduction

Computer graphics researchers have utilized a number of interesting data structures for fluid simulation including run-length encoded (RLE) grids [Houston et al. 2006; Irving et al. 2006; Chentanez and Müller 2011], octrees [Losasso et al. 2004], particle-based discretizations [Adams et al. 2007; Solenthaler and Pajarola 2009; Solenthaler and Gross 2011], velocity-vorticity domain decompositions [Golas et al. 2012], tetrahedral meshes [Feldman et al. 2005; Klingner et al. 2006], and Voronoi diagrams [Sin et al. 2009; Brochu et al. 2010]. However, uniform grids still remain a mainstay because of a number of advantages: a cache coherent memory layout, regular domain subdivisions suitable for parallelization, fast iterative solvers such as preconditioned conjugate gradient for solving partial differential equations, higher-order interpolation schemes which are are often difficult and computationally costly to generalize to unstructured data, and the ability to accelerate ray tracing algorithms for axis-aligned voxel data. Techniques such as adaptive mesh refinement (AMR) [Berger and Oliger 1984; Berger and Colella 1989; Sussman et al. 1999] and chimera grids [Benek et al. 1983; Benek et al. 1985; Dobashi et al. 2008] have remained prevalent because they use a number of Cartesian uniform grids. Similarly, the FLIP and PIC methods [Zhu and Bridson 2005; Losasso et al. 2008] use a background uniform grid for projection.

We focus on a single uniform grid structure as opposed to the multiple uniform grid structures used in AMR and chimera grid methods noting that for a variety of applications the added computational cost and complexity of many grids is unwarranted. However, there *are* many applications where one would want to use multiple uniform grids. Considering a single uniform grid, one could still add generality by changing the physical layout of the grid to be a modification of computational grid along the lines of curvilinear grids [Anderson et al. 1997]. Typically, a uniform grid exists in computational space and is mapped in some complex way to the physical domain where it can be boundary-fitted, stretched, compressed, etc., and Jacobians of the mapping are stored throughout the grid.

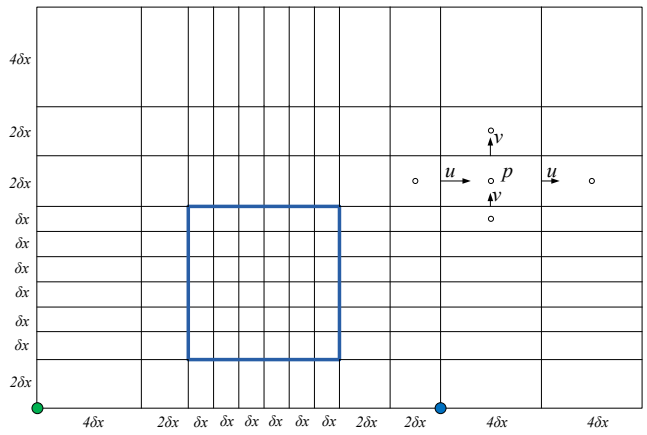

Figure 2: *Our far-field grid structure has a uniform Cartesian grid enclosing the viewer's domain of interest (outlined in blue) and extends much further in each of the other three spatial dimensions to provide a large simulation domain. There is a one to one mapping from the far-field grid structure to the standard uniform Cartesian grid which allows for exceptional cache coherency, ease of parallelism, and a natural mapping to the GPU – just as in the case of a uniform Cartesian grid. We have made some choices in limiting the far-field grid structure such as restricting the edge lengths of extended cells to be a power of two times the edges lengths along each axis of the uniform Cartesian grid in order to aggressively minimize overhead as compared to a standard uniform Cartesian grid.*

One drawback of these methods is that given a point in physical space, it is still difficult and computationally expensive (not $\mathcal{O}(1)$ in general) to find out which grid cell contains that point.

We propose a simple modification of the uniform grid where one can simply slide the vertical and horizontal planes left and right maintaining a Cartesian grid that allows for simple $\mathcal{O}(1)$ lookups of arbitrary points. Our grid structure can be seen as a subset of the general curvilinear/rectilinear grids used in other areas, while many optimizations (see Section 2) are made to this special case to gain significant computational efficiencies. Related work includes the far-field grid of Sussman [Sussman and Smereka 1997], where one layer of grid cells around the outside of the grid was extended in order to add relaxation to a pressure solver. We found this work highly motivating and generalize it to extending any number of layers but restrict those extensions in certain ways in order to make the lookups more efficient. Another related work is the Soroban grid [Yabe et al. 2004; Takizawa et al. 2007] where planes are solid in one dimension, lines in another, and grid points in another. While also motivating, this Soroban grid structure also struggles to identify the cell in which an arbitrary point is located. In particular, we stress that the cost of identifying the cells in which arbitrary locations are located as well as all other costs of our far-field grid structure only incur about a 10% overhead compared to a standard uniform grid.

2 Grid Structure

A far-field grid in three spatial dimensions is constructed by independently choosing a number of points in each of the three spatial dimensions and then taking the Cartesian product to obtain the locations of the corners of the grid cells. As illustrated in Figure 2, we store our data on this grid in the usual fashion with the scalar pressure at cell centers and velocities at the faces. Computing the cell that contains an arbitrary point in this general representation

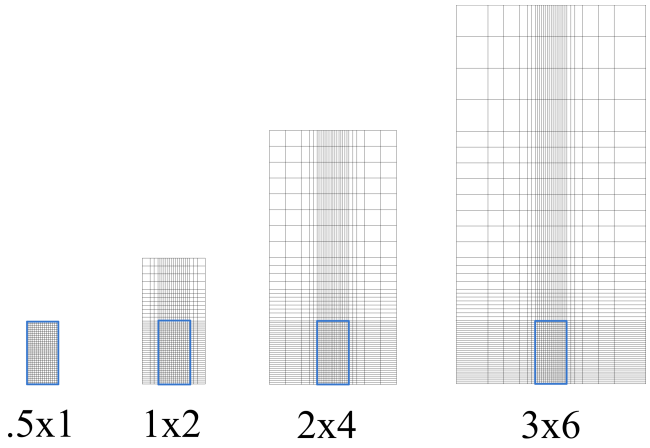

Figure 3: *Grid structures for the performance test covering domain sizes of* $.5 \times 1$, 1×2, 2×4, *and* 3×6. *The blue outline shows the fine grid domain of* $.5 \times 1$.

of the far-field grid requires a binary search along each axis of the grid and therefore requires $\mathcal{O}(\log N_x + \log N_y + \log N_z)$ runtime where N_x, N_y, and N_z are the number of points in each dimension. We can improve the asymptotic runtime of this data access by changing the layout as follows. First, the stretched cells in each dimension are organized hierarchically into n different layers. The edge length of a cell in the ith layer where $1 \leq i \leq n$ is given by $2^{i-1}\delta x$ where δx denotes the edge length of the finest cells on the grid. For each layer $i > 1$, we place k_{ij}^- stretched cells on the negative side of the previous layer along the jth axis ($j \in \{1, 2, 3\}$) and k_{ij}^+ stretched cells on the positive side of the previous layer. This particular implementation allows one to efficiently compute the cell which contains an arbitrary point in $\mathcal{O}(1)$ time. Recall that on a uniform grid, locating this cell can be achieved by simply dividing by δx and truncating in each dimension. On the far-field grid, this is accomplished with the aid of one precomputed one-dimensional array for each axis.

If we construct a uniform grid with the finest resolution, then each cell in the uniform grid will be contained entirely within a cell on the far-field grid. For example, the cell in the bottom left of Figure 3 would contain eight uniform fine grid cells. Thus, in each spatial dimension we can construct a simple one dimensional array which contains for each uniform grid cell the far-field cell that would contain it. In Figure 2, the x-dimension array would be $\{1, 1, 1, 1, 2, 2, 3, 4, 5, \ldots\}$ allowing us to map the uniform cell number to the far-field cell number. Once we identify the cell that contains an arbitrary point, we can interpolate values from the corners of that cell without any further information. However, MAC grids store information at faces and cell centers requiring a slightly modified approach. In order to determine the four (eight in three spatial dimensions) nearest neighbors for interpolating velocity or pressure, one needs to know whether a given location lives in the first or second half of the grid cell in each of the spatial dimensions. One simple way of encoding this information would be to make our one dimensional array twice as long and encode the terms in the array based on whether each half-size fine uniform cell is contained in the first or second half of the far-field cell. For example, using a "$-$"-sign for the left half and a "$+$"-sign for the right half of a cell one obtains $\{-1, -1, -1, -1, 1, 1, 1, 1, -2, -2, 2, 2, -3, 3, \ldots\}$ for the aforementioned case. These arrays can be precomputed and stored in each dimension separately and therefore incur a negligible memory cost – and is especially elegant for a GPU implementation.

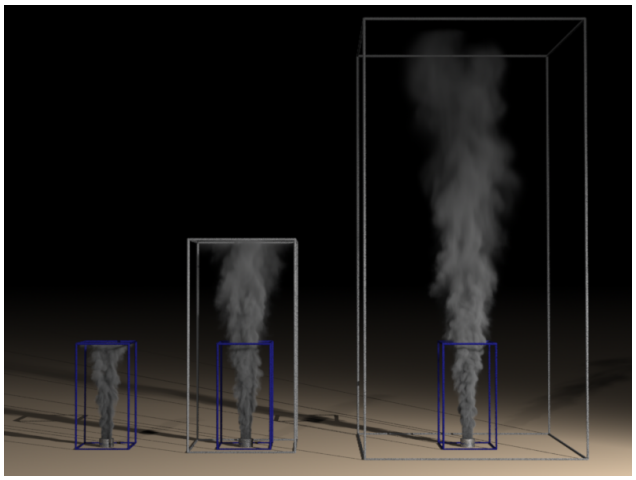

Figure 4: *Smoke simulations are performed in domains of sizes* $.5 \times 1 \times .5$*,* $1 \times 2 \times 1$ *and* $2 \times 4 \times 2$ *(outlined by the gray wireframes) obtained by extending the uniform domain of* $.5 \times 1 \times .5$ *(outlined by the blue wireframes) with our new grid structure. Note how the simulation on the domain with size* $2 \times 4 \times 2$ *allows for large camera angles without distracting grid boundaries. As seen in Table 1, using the far-field grid only increases the simulation time by a factor of 6 as opposed to using a fine grid for the entire domain which would increase the simulation time by a factor of 160.*

Since the overhead of the far-field grid is so small (around 10% in our final implementation), we took a slightly different approach to the precomputed array in order to have access to more information. We modify the array to tell us the level $i = 1, \ldots, n$ of the grid cell that contains the uniform grid cell, i.e. then our array becomes $\{3, 3, 3, 3, 2, 2, 1, 1, 1, \ldots\}$. This means that it requires slightly more work to calculate the cell that contains an arbitrary point. For example, given a position x, we compute the index $I(x)$ via

$$I(x) = \left\lfloor \frac{x - x_i^0}{2^{i-1}\delta x} \right\rfloor + I_i^0 \tag{1}$$

where $\lfloor \cdot \rfloor$ is the floor function and x_i^0 and I_i^0 are the precomputed location and index offsets for layer i respectively. For example, if x lies in the first cell in Figure 2, then x_i^0 and I_i^0 come from the left hand corner of the grid as indicated by the green dot. However, if x lies in the last cell, then one needs to obtain x_i^0 and I_i^0 from the blue dot lying two-thirds of the way to the right in Figure 2. This requires an "if"-statement to decide if the location in question is to the right of the second x_i^0 or not. In fact, this "if"-statement can be avoided by precomputing and storing a "+"-sign or "−"-sign in the array indicating which of the two x_i^0s to use. Note that one does not need to use an "if"–statement on the "±"-sign, but rather can compute x_i^0 and I_i^0 quickly by adding or subtracting the unsigned values stored in the array and multiplying those quantities with the appropriate x_i^0 and I_i^0 before combining them. For example, if the value indexed in the precomputed array is $m = \pm 3$, then $\frac{|m|-m}{2|m|} x^0_{|m|-} + \frac{|m|+m}{2|m|} x^0_{|m|+}$gives us either x^0_{3-} or x^0_{3+}.

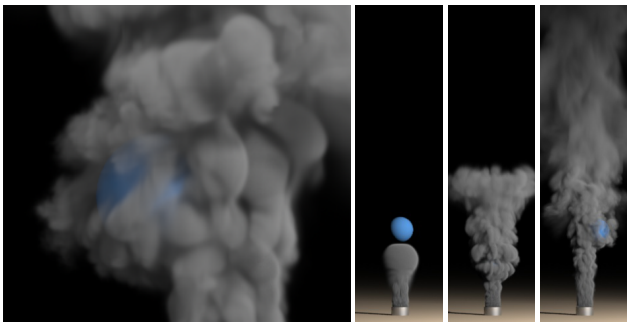

Figure 5: *The fine grid dynamically follows the kinematic sphere resolving the fine details due to its interactions with the smoke. Fine cells are allocated with a resolution of* 192^3 *around the sphere, and the entire far-field grid has a resolution of* $320 \times 448 \times 320$*.*

The converse of Equation 1 is useful for computing the location corresponding to a given index via $x(I) = 2^{i-1}\delta x(I - I_i^0) + x_i^0$where I is the input index and $x(I)$ is the output cell center location. Note that one needs to use the larger x_i^0 and I_i^0 pair that produces a positive $I - I_i^0$ when computing $x(I)$. We experimented with many implementations and discovered that it is actually quite quick to calculate on the fly which half cell contains a location for interpolating quantities such as velocities and pressure, and therefore only store the normal length (as opposed to double length) array.

Our strategy is to track the interesting regions of the flow field with the fine grid interior while still capturing a quite large computational domain with the stretched cells. Since these interesting regions

Grid	Domain	Resolution	DOF Increase	Advection	Projection	Total
Uniform	$.5 \times 1 \times .5$	$128 \times 256 \times 128$	1.0	1.0	1.0	1.0
Far-field	$1 \times 2 \times 1$	$176 \times 352 \times 176$	2.6	2.4	3.2	3.1
Uniform	$1 \times 2 \times 1$	$256 \times 512 \times 256$	8.0	5.6	13	12
Far-field	$2 \times 4 \times 2$	$208 \times 416 \times 208$	4.3	3.8	6.4	6.2
Uniform	$2 \times 4 \times 2$	$512 \times 1024 \times 512$	64	37	173	160
Far-field	$3 \times 6 \times 3$	$224 \times 448 \times 224$	5.4	4.5	11	10
Uniform	$3 \times 6 \times 3$	$768 \times 1536 \times 768$	216	-	-	-

Table 1: *Timing data for smoke simulations ran with a CFL number of .99 on both far-field and uniform grids for domains of varying sizes in three spatial dimensions. Timings are given as the ratio of the time required for a simulation as compared to that for the* $.5 \times 1 \times .5$ *uniform grid. Since only coarser cells are added, the number of timesteps per frame is constant over all simulations. The simulations were run on a single machine using a single thread for two spatial dimensions and 16 threads for three spatial dimensions. The results are only shown for three spatial dimensions noting that the results in two spatial dimensions were similar. Whereas the fourth column of the table, i.e. DOF Increase, shows the theoretical increase in degrees of freedom, the actual increase in runtime given in the last column can be significantly higher for the uniform grid due to the slower nature of convergence for the divergence free projection – meanwhile, the far-field grid not only has a much lower theoretical DOF increase, but also behaves similarly in performance. In fact, on the* $2 \times 4 \times 2$ *simulations the uniform grid is over* 25 *times slower than the far-field grid. The* $3 \times 6 \times 3$ *simulation was impractical to run on the uniform grid in three spatial dimensions; however, the results in two spatial dimensions indicate that it scales even more poorly.*

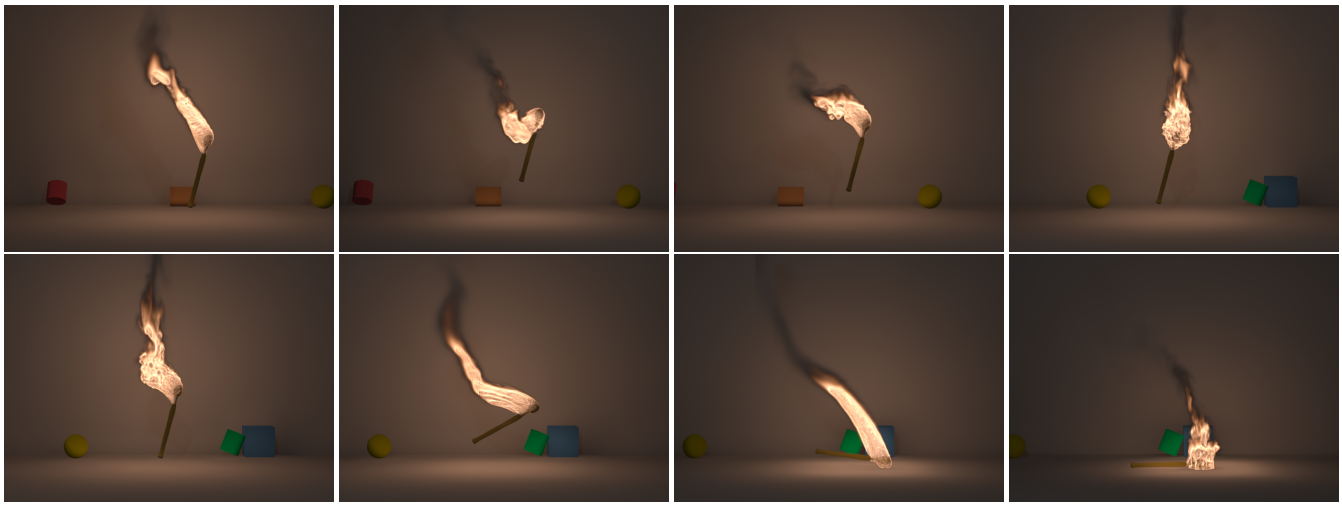

Figure 6: *A detailed flaming torch is moved and used to illuminate different sections of a large domain (top). The torch is then dropped on the ground generating trailing flame details as it falls (bottom). The fine grid has a resolution of* 128^3 *and dynamically tracks the level set in order to resolve the visually appealing fine scale flame wrinkles, while the entire far-field grid of resolution* $208 \times 288 \times 208$ *efficiently resolves the global motion of the fire and smoke.*

are usually around objects or at the point of camera focus, they can potentially move and change size, which we handle by dynamically changing the structure of the far-field grid. This is accomplished by moving grid lines in each dimension (see Section 4) as motivated by [Yabe et al. 2004; Takizawa et al. 2007]. One can also add and delete grid lines in each dimension noting that one needs to be careful that the total number of cells does not increase beyond the original allocation of the array because this would incur the extra cost of reallocating large arrays for three dimensional data. We stress that this does not mean that the grid can never grow in the number of cells; it just means that one should preallocate a buffer which is large enough to include the maximum number of cells.

3 Incompressible Flow

Our smoke solver follows the general procedure of [Fedkiw et al. 2001]. The inviscid incompressible Navier-Stokes equations are given by

$$\mathbf{u}_t + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{1}{\rho}\nabla p + \mathbf{f}, \tag{2}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{3}$$

where $\mathbf{u}$ is the velocity field, ρ is the density, p is the pressure, and $\mathbf{f}$ is the sum of any external forces (such as vorticity confinement, buoyancy, and gravity) scaled by ρ. First we compute the intermediate velocity $\mathbf{u}^*$ ignoring the pressure term via $(\mathbf{u}^* - \mathbf{u}^n)/\Delta t + (\mathbf{u}^n \cdot \nabla)\mathbf{u}^n = \mathbf{f}$ making use of semi-Lagrangian advection [Stam 1999; Kim et al. 2005; Selle et al. 2008] before computing the final velocity as $\mathbf{u}^{n+1} = \mathbf{u}^* - \Delta t \nabla p/\rho$. The pressure p is calculated by solving a Poisson equation of the form

$$V_{\text{cell}} \nabla \cdot \left(\frac{\nabla \hat{p}}{\rho} \right) = V_{\text{cell}} \nabla \cdot \mathbf{u}^*. \tag{4}$$

where $\hat{p} = p\Delta t$ is a scaled pressure, V_{cell} is the volume of a cell, and we have used the volume weighted divergence as outlined in [Losasso et al. 2004] in order to obtain a symmetric positive definite system on our stretched grid. Equation 4 can be solved efficiently using the preconditioned conjugate gradient method with an incomplete Cholesky preconditioner.

Invoking the second vector form of Green's theorem on both sides of Equation 4, we obtain

$$\sum_{\text{faces}} \frac{\nabla \hat{p}}{\rho} \cdot \mathbf{dA}_{\text{face}} = \sum_{\text{faces}} \mathbf{u}^*_{\text{face}} \cdot \mathbf{dA}_{\text{face}} \tag{5}$$

where $\mathbf{dA}_{\text{face}}$ is the area-weighted normal of the face. Note that although the faces are not equidistant between pressure locations where the edge length along a particular axis changes, we still compute the pressure gradients in the usual way subtracting the adjacent values and dividing by the actual distance between the cell centers. Furthermore, we demonstrate that this pressure discretization on the far-field grid achieves second-order accuracy as on uniform grid (see Table 2).

We demonstrate the efficacy of our method by simulating smoke on the grids outlined in Figure 3 where the initial fine grid is padded from left to right with a larger and larger far-field grid demonstrating the significant domain extension obtainable using this approach (see Figure 4 and Table 1).

4 Dynamic Tracking

Solid objects are handled in the usual manner by setting the velocity of cells covered by the object to the object velocity and setting Neu-

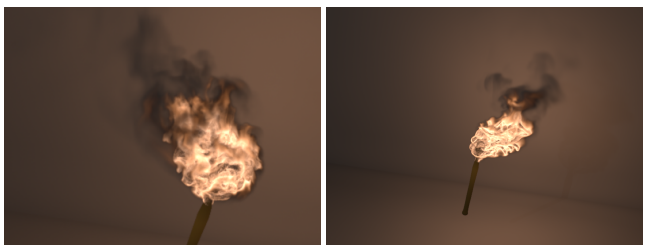

Figure 7: *A dynamically resizing fine grid with a resolution up to* 160^3 *dynamically tracks the level set in order to resolve local flame details of a torch moving in the wind. The far-field grid has a resolution of* $210 \times 300 \times 210$.

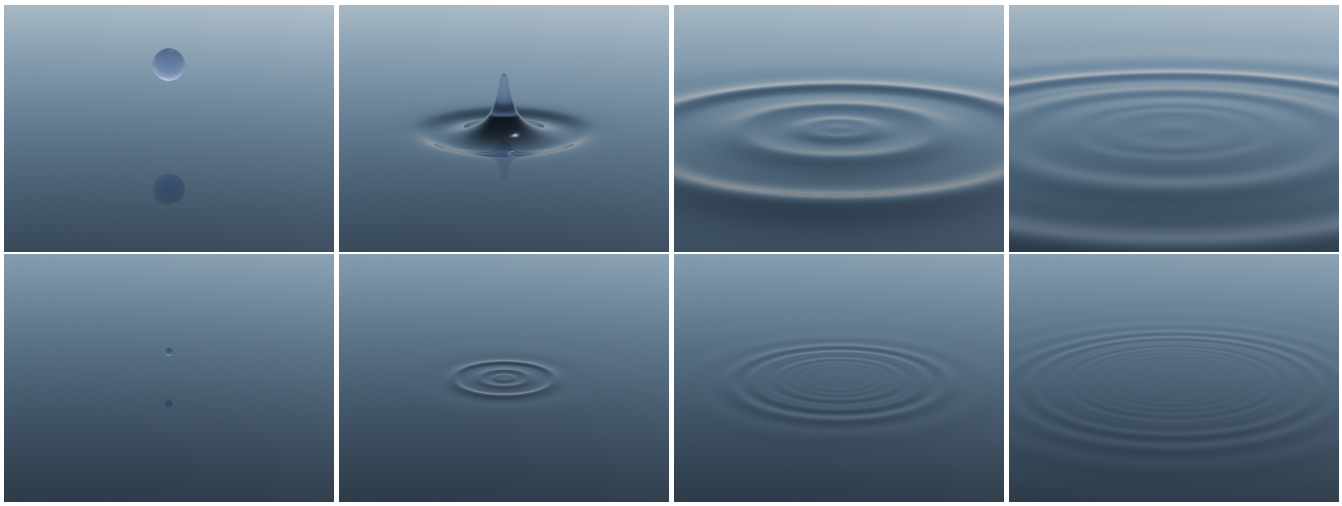

Figure 8: *A spherical water drop falls into a large body of water 40 times larger than the spherical drop. The stretched grid cells allow the resulting radially outgoing waves to continue propagating into the coarser domain as opposed to reflecting back into the fine domain. A fine grid with a resolution of* $384 \times 96 \times 384$ *is placed near the sphere, and the far-field grid has resolution* $672 \times 120 \times 672$*. This allows the grid to resolve the initial splash (top) and capture surface details such as the resulting radially outgoing waves that propagate throughout the large domain (bottom).*

mann boundary conditions inside the object for the pressure solve. The far-field grid can adaptively place the finest discretizations in a moving and resizing region of interest that is a subset of the entire computational domain and dynamically track it during the course of a simulation. Figure 5 shows smoke interacting with a kinematically driven sphere. At the beginning of each simulation, we choose a bounding box in the vicinity of the object and precompute a translation from the center of the bounding box to the center of the object maintaining that offset throughout the simulation. The fine region of the far-field grid lies entirely within this bounding box, and the grid lines are redistributed such that the resulting new stretched cells encompass the rest of the domain. This allows us to resolve fine scale fluid details in the vicinity of the object.

Solving the Navier-Stokes equations on a grid that dynamically tracks an object requires storing the grid structure both at time t^n and t^{n+1} so that the velocity field can be interpolated from the time t^n grid to the time t^{n+1} grid in order to trace the semi-Lagrangian rays backward in time and interpolate from the time t_n grid. The projection step does not require modification.

We demonstrate the use of the far-field grid for fire simulation in Figures 6 and 7 using the thin flame model [Nguyen et al. 2002] (see also [Hong et al. 2007]). A dynamic surface where the chemical reaction occurs is represented with a level set, and Rankine-Hugoniot jump conditions for velocity and pressure are enforced across the interface in order to conserve mass and momentum. Reacted and unreacted materials are distinguished with densities of ρ_r and ρ_u respectively in Equation 2. The Poisson equation with pressure jumps across the interface is solved on the far-field grid using the volume weighted divergence discussed in Section 3. In each timestep the level set interface is evolved by advecting the signed distance function ϕ and applying the fast marching method. All of these discretizations are straightforward to apply since the far-field grid is also a Cartesian grid.

Grid	L^1 Error	Order	L^∞ Error	Order
128×128	1.68×10^{-4}	–	1.02×10^{-3}	–
256×256	4.17×10^{-5}	2.01	2.54×10^{-4}	2.01
512×512	1.04×10^{-5}	2.01	6.33×10^{-5}	2.01
1024×1024	2.59×10^{-6}	2.00	1.58×10^{-5}	2.00
2048×2048	6.58×10^{-7}	1.98	3.93×10^{-6}	2.01

Table 2: *Poisson solver accuracy on the far-field grid. A Poisson equation with an analytic solution of* $\phi(x, y) = \sin(\pi x)\sin(\pi y)$ *is solved on a far-field grid.*

However, in order resolve the detailed flame wrinkles around the level set interface, it is preferable to enclose the level set which can move independently of the torch (e.g. due to wind). We accomplish this in two steps. First, we track the torch by translating the entire far-field grid by adding or removing cells as in [Rasmussen et al. 2004] (see also [Shah et al. 2004; Cohen et al. 2010]). Second, the fine grid dynamically moves around in the interior of the far-field grid in order to track the level set representation of the flame. This is accomplished by computing the level set's bounding box at each time step and clamping it to a maximum and minimum size along each dimension.

5 Non-Reflecting Waves

Our fluid solver for free surface flows uses the particle level set method of [Enright et al. 2002]. A constant number of positive and negative particles are seeded per cell within a distance of $3\delta x$ (stressing that this is the fine grid δx) from the interface. Since we would like to resolve fine scale details on the fine grid, we prefer a large number of particles on the fine grid. In contrast, we place a coarse grid where we expect to lose detail due to numerical dissipation and therefore require fewer particles in the coarse regions. In the projection step, we use the pressure modifications of [Enright et al. 2003] to achieve second order accuracy at the free surface. The fast marching method is used to maintain the signed distance property of ϕ in the vicinity of the interface and velocity extrapolation is performed using closest point extrapolation to ensure that the interface moves unhindered by the air. Again, we stress that the Cartesian nature of our underlying grid makes all of these discretizations straightforward to apply. We demonstrate our solver for free surface flows in Figures 8, 9, and 10. (Figure 10 uses vortex particles [Selle et al. 2005].)

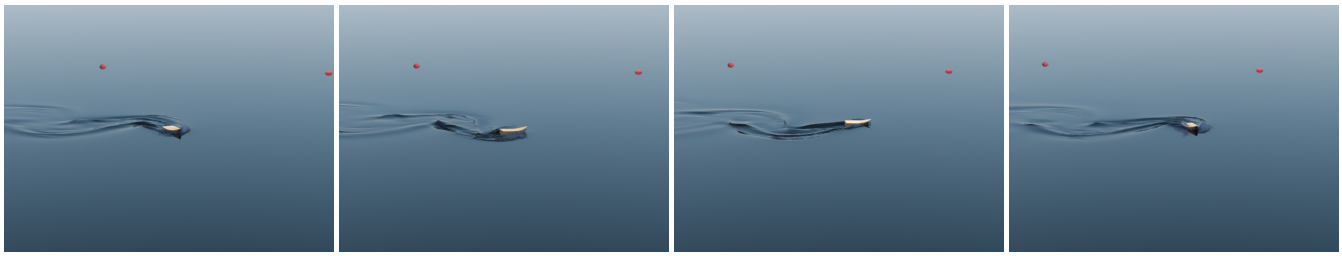

Figure 10: *A fine grid dynamically follows a boat allowing the simulation to capture fine scale details of the wake in the presence of far-field boundary conditions. Vortex particles are seeded behind the boat in order to generate additional fine scale details. The fine grid has a resolution of* $192 \times 32 \times 192$ *and the far-field grid has a resolution of* $384 \times 58 \times 320$.

It is often computationally intractable to sufficiently approximate far-field boundary conditions on a uniform grid while preserving a high-degree of detail. The domain needs to be large enough to enclose the fluid flow without boundaries influencing the fluid motion while a sufficient number of cells still need to be placed throughout the domain in order to resolve the fine scale details in the regions of interest. Thus, researchers have explored various methods for approximating far-field boundary conditions such as perfectly matched layers [Söderström et al. 2010]. Our grid structure is able to efficiently approximate far-field boundary conditions by progressively extending the grid cells as one moves farther away from the fine uniform grid covering the region of interest. Since numerical viscosity often damps out fine scale features in the fluid flow before reaching the boundary, the extended grid cells can sufficiently resolve the flow far away from the highest resolution domain surrounding the region of interest at a much lower computational cost compared to placing uniform cells throughout the domain. Furthermore, we have verified that the outgoing wave propagation speed for the simulation shown in Figure 8 is the same on both the far-field grid and the corresponding uniform grid (where the cell size on the uniform grid is the same as that in the fine region of the far-field grid).

We have noticed that if the fine grid is insufficiently large compared to the initial splash and transitions to the coarser grid before the outgoing waves are sufficiently dissipated, the lower frequency components of the waves are transmitted into the coarser domain while the higher frequency components of the waves can be reflected back into the fine domain. This results in artificial interference patterns which are especially noticeable in the case of the falling spherical water drop due to the radial nature of the outgoing waves. An alternative solution (that we do not follow) for alleviating these reflected waves is to dynamically resize the fine grid such that the radially outward propagating initial wave generated by the splash is always enclosed in the fine region. This approach causes the speed of the simulation to decrease significantly as the simulation progresses since the fine grid needs to be expanded along two of the three dimensions. On the other hand, these extra grid lines can be removed after the splash dissipates.

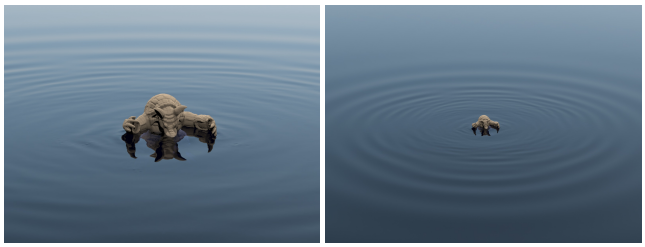

Figure 9: *A solid armadillo falls into a large body of water which enforces a far-field boundary condition. The fine grid has a resolution of* $384 \times 96 \times 384$ *and the far-field grid has a resolution of* $672 \times 168 \times 672$.

6 Extensions and Conclusions

We have presented an efficient grid structure that extends a uniform grid to create a significantly larger far-field grid that is a heavily-optimized special case of existing curvilinear grids. Our approach preserves almost every computational advantage of uniform grids including cache coherency, regular subdivisions for parallelization, simple data layout, and the existence of efficient numerical algorithms for solving partial differential equations due to the underlying Cartesian nature of our grid. We demonstrate that our grid structure allows us to simulate significantly larger domains than a uniform grid thus allowing us to capture far-field boundary conditions while maintaining the same resolution in regions of interest. Efficient algorithms for ray tracing axis-aligned voxel data on uniform grids are straightforward to adapt to the far-field grid which we demonstrate by performing all of our lighting precomputation and rendering directly on the far-field grid.

While our specific implementation is unable to simultaneously track multiple regions of interest with fine grids on a single far-field grid, one could instead track multiple regions by overlaying multiple far-field grids that each have a single region of interest. This adds additional fine regions at the cost of maintaining multiple hierarchical subdivisions. In fact, an AMR or chimera grid implementation could readily make use of one or more far-field grids. Another approach is to relax our restrictions on the general far-field grid and allow for multiple fine regions on a single far-field grid by choosing multiple regions of fine grid points along each axis. However, this approach leads to additional fine regions in areas that are not regions of interest since the fine regions in the far-field grid are determined by the Cartesian product of the fine regions along each axis. For example, if we wanted to track two objects with fine uniform grids located at (x_1, y_1) and (x_2, y_2) in two spatial dimensions, we would also be forced to resolve the regions surrounding (x_1, y_2) and (x_2, y_1) resulting in four fine regions.

There are also visual artifacts that can arise due to the stretched cells. In smoke simulations, there is an increased amount of numerical dissipation in regions with stretched cells which can manifest when the smoke density field is rendered. In fire simulations, a similar problem exists with the temperature field but can be resolved by enclosing the region of interest (level set wrinkles) within the finest discretization since a fire simulation is often localized. In water simulations, the increased dissipation in stretched regions tends to rapidly smooth out the sharp features of the free surface. However, outside of the region of interest, there are usually fewer of

these sharp features and the increased numerical dissipation is less noticeable. Thus, the far-field grid is more suitable to modeling localized fluid phenomena with a bounded region of interest such as fire and water as opposed to smoke where the density tends to distribute itself throughout the entire domain.

Another interesting direction for future work is the implementation of the far-field grid on the GPU towards real time applications. The far-field grid can be represented in three spatial dimensions using three small one-dimensional arrays or textures which can be bound either as constants or small texture or stored in group shared memory. Due to the Cartesian nature of the far-field grid, the data stored on the far-field grid has a direct mapping to three-dimensional textures or arrays and therefore numerical algorithms optimized for uniform grids on the GPU can be easily adapted to operate on the far-field grid. The far-field grid's ability to dynamically resolve details inside a region of interest within a very large domain while maintaining almost every computational advantage of a uniform grid has the potential to facilitate the use of fluid simulations in numerous real time applications.

Acknowledgements

Research was supported in part by ONR N00014-09-1-0101, ONR N-00014-11-1-0027, ONR N00014-11-1-0707, ARL AHPCRC W911NF-07-0027, and the Intel Science and Technology Center for Visual Computing. Computing resources were provided in part by ONR N00014-05-1-0479. We would like to thank Jure Leskovec and Christos Kozyrakis for additional computing resources as well as Andrej Krevl and Jacob Leverich for helping us use those resources.

References

ADAMS, B., PAULY, M., KEISER, R., AND GUIBAS, L. J. 2007. Adaptively sampled particle fluids. *ACM Trans. Graph. (SIGGRAPH Proc.) 26*, 3.

ANDERSON, D. A., TANNEHILL, J. C., AND PLETCHER, R. H. 1997. *Computational Fluid Mchanics and Heat Transfer*. Taylor & Francis, 531.

BENEK, J. A., STEGER, J. L., AND DOUGHERTY, F. C. 1983. A flexible grid embedding technique with applications to the euler equations. In *6th Computational Fluid Dynamics Conference*, AIAA, 373–382.

BENEK, J. A., BUNING, P. G., AND STEGER, J. L. 1985. A 3-d chimera grid embedding technique. In *7th Computational Fluid Dynamics Conference*, AIAA, 322–331.

BERGER, M., AND COLELLA, P. 1989. Local adaptive mesh refinement for shock hydrodynamics. *J. Comput. Phys. 82*, 64–84.

BERGER, M., AND OLIGER, J. 1984. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comput. Phys. 53*, 484–512.

BROCHU, T., BATTY, C., AND BRIDSON, R. 2010. Matching fluid simulation elements to surface geometry and topology. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 47:1–47:9.

CHENTANEZ, N., AND MÜLLER, M. 2011. Real-time eulerian water simulation using a restricted tall cell grid. *ACM Trans. Graph. (SIGGRAPH Proc.) 30*, 4, 82:1–82:10.

COHEN, J. M., TARIQ, S., AND GREEN, S. 2010. Interactive fluid-particle simulation using translating eulerian grids. In *Proc. of the 2010 ACM SIGGRAPH Symp. on Interactive 3D Graphics and Games*, 15–22.

DOBASHI, Y., MATSUDA, Y., YAMAMOTO, T., AND NISHITA, T. 2008. A fast simulation method using overlapping grids for interactions between smoke and rigid objects. *Comput. Graph. Forum 27*, 2, 477–486.

ENRIGHT, D., MARSCHNER, S., AND FEDKIW, R. 2002. Animation and rendering of complex water surfaces. *ACM Trans. Graph. (SIGGRAPH Proc.) 21*, 3, 736–744.

ENRIGHT, D., NGUYEN, D., GIBOU, F., AND FEDKIW, R. 2003. Using the particle level set method and a second order accurate pressure boundary condition for free surface flows. In *Proc. 4th ASME-JSME Joint Fluids Eng. Conf., number FEDSM2003–45144. ASME*.

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

FELDMAN, B., O'BRIEN, J., KLINGNER, B., AND GOKTEKIN, T. 2005. Fluids in deforming meshes. In *Proc. of the ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 255–259.

GOLAS, A., NARAIN, R., SEWALL, J., KRAJCEVSKI, P., DUBEY, P., AND LIN, M. 2012. Large-scale fluid simulation using velocity-vorticity domain decomposition. *ACM Trans. Graph. 31*, 6, 148:1–148:9.

HONG, J., SHINAR, T., AND FEDKIW, R. 2007. Wrinkled flames and cellular patterns. *ACM Trans. Graph. (SIGGRAPH Proc.) 26*, 3, 47.

HOUSTON, B., NIELSEN, M., BATTY, C., NILSSON, O., AND MUSETH, K. 2006. Hierarchical RLE level set: A compact and versatile deformable surface representation. *ACM Trans. Graph. 25*, 1, 1–24.

IRVING, G., GUENDELMAN, E., LOSASSO, F., AND FEDKIW, R. 2006. Efficient simulation of large bodies of water by coupling two and three dimensional techniques. *ACM Trans. Graph. (SIGGRAPH Proc.) 25*, 3, 805–811.

KIM, B.-M., LIU, Y., LLAMAS, I., AND ROSSIGNAC, J. 2005. Flowfixer: Using BFECC for fluid simulation. In *Eurographics Workshop on Natural Phenomena 2005*.

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

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

LOSASSO, F., TALTON, J., KWATRA, N., AND FEDKIW, R. 2008. Two-way coupled sph and particle level set fluid simulation. *IEEE TVCG 14*, 4, 797–804.

NGUYEN, D., FEDKIW, R., AND JENSEN, H. 2002. Physically based modeling and animation of fire. *ACM Trans. Graph. (SIGGRAPH Proc.) 21*, 721–728.

RASMUSSEN, N., ENRIGHT, D., NGUYEN, D., MARINO, S., SUMNER, N., GEIGER, W., HOON, S., AND FEDKIW, R. 2004. Directable photorealistic liquids. In *Proc. of the 2004 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 193–202.

SELLE, A., RASMUSSEN, N., AND FEDKIW, R. 2005. A vortex particle method for smoke, water and explosions. *ACM Trans. Graph. (SIGGRAPH Proc.) 24*, 3, 910–914.

SELLE, A., FEDKIW, R., KIM, B., LIU, Y., AND ROSSIGNAC, J. 2008. An Unconditionally Stable MacCormack Method. *J. Sci. Comp. 35*, 2, 350–371.

SHAH, M., COHEN, J. M., PATEL, S., LEE, P., AND PIGHIN, F. 2004. Extended galilean invariance for adaptive fluid simulation. In *Proc. of the 2004 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 213–221.

SIN, F., BARGTEIL, A. W., AND HODGINS, J. K. 2009. A point-based method for animating incompressible flow. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, ACM, New York, NY, USA, SCA '09, 247–255.

SÖDERSTRÖM, A., KARLSSON, M., AND MUSETH, K. 2010. A pml-based nonreflective boundary for free surface fluid animation. *ACM Trans. Graph. 29*, 5, 136:1–136:17.

SOLENTHALER, B., AND GROSS, M. 2011. Two-scale particle simulation. *ACM Trans. Graph. (SIGGRAPH Proc.) 30*, 4, 81:1–81:8.

SOLENTHALER, B., AND PAJAROLA, R. 2009. Predictive-corrective incompressible sph. *ACM Trans. Graph. (SIGGRAPH Proc.) 28*, 3, 40:1–40:6.

STAM, J. 1999. Stable fluids. In *Proc. of SIGGRAPH 99*, 121–128.

SUSSMAN, M., AND SMEREKA, P. 1997. Axisymmetric free boundary problems. *J. Fluid Mech. 341*, 269–294.

SUSSMAN, M., ALGREM, A. S., BELL, J. B., COLELLA, P., HOWELL, L. H., AND WELCOME, M. L. 1999. An adaptive level set approach for incompressible two-phase flow. *J. Comput. Phys 148*, 81–124.

TAKIZAWA, K., YABE, T., TSUGAWA, Y., TEZDUYAR, T. E., AND MIZOE, H. 2007. Computation of free-surface flows and fluid object interactions with the cip method based on adaptive meshless soroban grids. *Comput. Mech. 40*, 1, 167–183.

YABE, T., MIZOE, H., TAKIZAWA, K., MORIKI, H., IM, H.-N., AND OGATA, Y. 2004. Higher-order schemes with cip method and adaptive soroban grid towards mesh-free scheme. *J. Comput. Phys. 194*, 1.

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