EG UK Theory and Practice of Computer Graphics (2009)
Wen Tang, John Collomosse (Editors)

An Adaptive Sampling Approach to Incompressible Particle-Based Fluid

Woosuck Hong[1], Donald H. House[2] and John Keyser[1]

[1]Department of Computer Science, Texas A& M University
[2]Department of Computer Science, Clemson University

Abstract

We describe an adaptive particle-based technique for simulating incompressible fluid that uses an octree structure to compute inter-particle interactions and to compute the pressure field. Our method extends the hybrid Flip technique by supporting adaptive splitting and merging of fluid particles, and adaptive spatial sampling for the reconstruction of the velocity and pressure fields. Particle splitting allows a detailed sampling of fluid momentum in regions of complex flow. Particle merging, in regions of smooth flow, reduces memory and computational overhead. The octree supporting field-based calculations is adapted to provide a fine spatial reconstruction where particles are small and a coarse reconstruction where particles are large. This scheme places computational resources where they are most needed, to handle both flow and surface complexity. Thus, incompressibility can be enforced even in very small, but highly turbulent areas. Simultaneously, the level of detail is very high in these areas, allowing the direct support of tiny splashes and small-scale surface tension effects. This produces a finely detailed and realistic representation of surface motion.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Animation]: Fluid simulation and natural phenomena

1. Introduction

In the simulation of incompressible fluid, one is faced with two phenomena that seem to demand quite different simulation methodologies. One dominant factor in fluid flow is that moving fluid is transporting real material, and thus the properties of the fluid are being transported (advected) with this material. This seems to call for a Lagrangian or particle-based simulation scheme, which naturally tracks the motion of discrete "chunks" of fluid mass, and thus provides a vehicle for handling advection in a straightforward way. The other dominant factor in fluid flow is that it is shaped largely by pressure gradients. Unlike advection, pressure is a massless field property that is poorly represented by moving particles, but quite naturally represented by an Eulerian or spatial-grid scheme. The usual way to handle advection within an Eulerian simulation is by backtracing, as in the *semi-Lagrangian* method [Sta99], but this approach is well known to lead to considerable dissipation of flow complexity. It is possible to approximate the pressure effects in compressible flow using the metaphor of interparticle spring forces. However, this method is poorly suited to handling water flow, which is essentially incompressible, yielding an unnatural looking "bouncy" water surface.

Another important factor in fluid representation is that, within a flow, some regions will be highly complex while others are quite smooth. Similarly, for rendering purposes, having a detailed and convincing representation of the moving surface geometry may be considerably more important than having high detail below the surface. Thus, within a simulation, the ability to refine or coarsen the representation can be quite useful, regardless of whether an Eulerian or a Lagrangian computational methodology is used. To capture the complex flow and fine geometric detail at the surface, finer grid spacing or particle sampling can be used, while regions of smooth flow permit coarsening of grid or particle representations without loss of detail, saving computation time and space.

Finally, if the simulation level of detail is to be very fine, a representation of surface tension becomes important. This is especially true when it is desired to represent spray and

DOI: 10.2312/LocalChapterEvents/TPCG/TPCG09/069-076

splashes. A particle-based scheme naturally accounts for fine spray, but will not directly support the representation of surface tension, which is needed for the simulation of small water drops. Again, there is a need for both a particle-based representation of the material of the fluid, and a grid-based representation of the small-scale fluid surface.

Our work attempts to find a suitable simulation framework that exploits the right methodologies for the various stages of the simulation, while maintaining a representation that is appropriately adapted to the required simulation granularity. For our approach, we have developed an adaptive version of the hybrid Fluid-Implicit-Particle(FLIP) method [ZB05], which is essentially particle-based, but augmented by a computational grid for speeding the calculation of inter-particle interactions and for computing the pressure field. To make the approach adaptive, we have implemented a particle splitting and merging scheme using both depth from the surface and flow complexity to determine when particles should be split or merged. We use an octree spatial data structure to represent the computational grid, with octree cell size determined at each time step by the particle sampling in the neighborhood of the cell. Thus, the method is essentially particle based, with particle advection providing the transport mechanism in the fluid, and the pressure, inter-particle interaction and surface geometry being calculated on an octree whose resolution is locally determined by the particles.

The main contribution of this work is to demonstrate how to support a fully adaptive simulation of both particle and grid-based components for incompressible fluids. As part of this, we present a new method for determining spatial grid resolution based on adaptive particle sizes, and coupling this grid with the particles. We also present improvements to particle splitting and merging. Our approach allows us to have the resolution needed to incorporate surface tension effects into an adaptive particle-based simulation, and we present a new method for computing these forces.

Figure 1 shows two frames from a 2D simulation conducted using our approach, illustrating both the particle sampling and the octree structure. The detail at the surface and the coarsening of the representation below the surface can be clearly seen. Since the simulation is grounded in a particle representation, splashes and spray are handled directly, and actually account for true fluid mass. At the same time, the octree representation provides the fine detail needed around spray particles so that nearby particles continue to form a surface over which surface tension forces can be computed.

2. Related work

Fluid simulations are typically grounded in the Navier-Stokes equations, describing fluid flow. An Eulerian representation of the incompressible form of these equations is:

$$\frac{\partial u}{\partial t} = f - (u \cdot \nabla)u - \frac{1}{\rho}\nabla p + \nu\nabla^2 u \qquad (1)$$

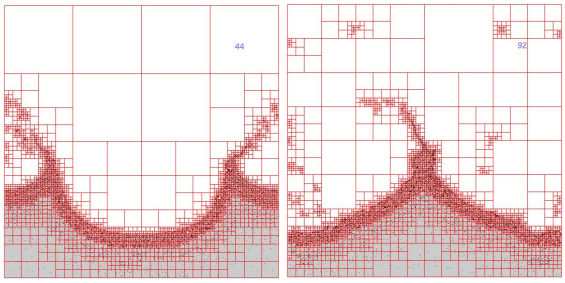

Figure 1: *Hybrid adaptive particle simulation (2D) showing particle and octree structure*

$$\nabla \cdot u = 0 \qquad (2)$$

where u is the velocity field, p is the pressure field, f are external forces, ρ is a density measure, and ν is kinematic viscosity. The $(u \cdot \nabla)u$ term accounts for advection of the fluid, the ∇p term accounts for acceleration due to pressure gradients, and the $\nu\nabla^2 u$ accounts for velocity diffusion due to the fluid's viscosity. The second equation ensures that the fluid is divergence free and thus incompressible.

2.1. Hybrid fluid simulation

Fluid simulations are divided into two main categories, Eulerian(Grid-based) and Lagrangian(Particle-based). The two approaches are combined in the hybrid fluid simulation technique in order to obtain the strength from each technique. The first truly hybrid fluid simulation method was introduced in computer graphics by Zhu and Bridson [ZB05]. They demonstrated how the Particle-in-Cell (PIC) [HW65], and Fluid-Implicit-Particle (FLIP) [BR86] methods could be used to simulate sand flow. Both PIC and FLIP are fundamentally particle-based, as particles carry the momentum of the fluid, but a MAC grid [HW65] is used for the efficient computation of the spatial interactions required to compute diffusion and to guarantee incompressibility. Using this auxiliary grid, incompressibility and boundary conditions can be enforced much more efficiently than in a pure Lagrangian scheme [KTO95] and [CEL06]. In both PIC and FLIP, mass particles have their own velocity and position, which are integrated numerically using velocity updates obtained from the grid. Since the methods are Lagrangian, the velocity backtracing step of the semi-Lagrangian method can be avoided, greatly reducing numerical dissipation and loss of flow detail. Kim and Ihm [KCC*06] adapted these approaches to water animation, demonstrating realistic turbulent splashing without volume loss.

2.2. Adaptive Fluid Simulation Methods

A number of authors have looked at adaptive methods for simulating fluid. Within the Eulerian simulation community,

Losasso et al. [LGF04] promoted the use of an adaptive octree data structure, instead of a fixed grid, to achieve high surface resolution in complex flows. Shi and Yu also used an adaptive Octree for simulation [SY04]. More recently, Irving et al. [IGLF06] used a simpler idea, based on the RLE Level set [HNB*06], simply to coarsen cells a larger distance from the surface. In Lagrangian methods, early work by Desbrun and Cani-Gasculel [DG98] looked at spliting and merging particles in an SPH(Smooth Particle Hydrodynamics) simulation to sample the fluid adaptively. Most recently, Adams et al. [APKG07] have proposed an improved adaptive method for SPH compressible fluid simulation, Song et al. proposed a method that combines a (non-adaptive) Lagrangian scheme with an octree-based solver [SKK07] and Hong et al. [HHK08] demonstrated adaptive particle sampling within an incompressible framework.

3. Overview of our adaptive fluid simulation

Our overall method is an extension to the hybrid FLIP method [ZB05]. We maintain the general FLIP approach of using particles to capture fluid advection and a grid to perform pressure calculations. In order to make the simulation more adaptive, we incorporate adaptive techniques in both key aspects: namely particle splitting and merging for Lagrangian calculation, and octree grids for Eulerian calculation. A surface tension force is added as an external force to more accurately capture small-scale fluid motions. While each of these ideas has been applied individually in prior work, we show that these techniques can be applied collectively in a unified manner. Specifically, we show how the adaptive particles can be used to drive the formation of the octree grid.

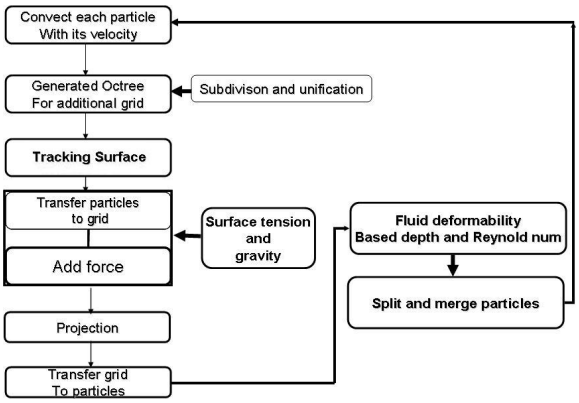

Figure 2: *Overall simulation structure*

Figure 2 presents a diagram illustrating our overall process. The following steps are iterated during the simulation.

- Update particle positions based on velocity.
- Generate an octree grid with resolution based on local particle radii.
- Identify surface cells from among the current fluid cells. Subdivide cells in the air based on the distance to the fluid surface.
- Construct a signed distance field from the particles on the octree grid, allowing identification of particles near the surface. Local surface curvature is computed for these particles to provide a surface tension force.
- Use the particle velocities to reconstruct a set of fields on the octree. Fields are maintained, at the center of each octree cell, for velocity, pressure, distance from the surface, and applied forces. Forces include gravity, surface tension, and any externally applied forces. A circular kernel is used for the reconstruction.
- Create a copy of the octree. Using two octrees avoids the additional steps and computation that would be needed to update the field in place.
- Calculate the pressure field, and project the flow field onto the nearest divergence free field [LGF04].
- Transfer velocity changes in the flow field back onto the particles' velocities.
- Compute a fluid deformability measure based on depth and local motion [HHK08]
- Split or merge the particles based on the local deformability measure.

Several of these steps are common to the hybrid FLIP method, or are explained in detail in other papers [ZB05] [HHK08] , and we will not discuss them further here. We will focus our discussion in the following sections on the key areas where we present new ideas or improvements on prior approaches. Specifically, we will describe the creation of octree cells based on particles (Section 4), splitting and merging particles (Section 5), communicating velocity information between the particles and the grid (Section 6), and computing surface tension (Section 7).

For illustrative purposes, most of our figures and examples will be presented in 2D, while the descriptions and formulae will be in 3D (e.g. octrees instead of quadtrees). However, all the methods discussed have been implemented and work equally well in both two and three dimensions.

4. Octree Generation

Key to our method is the generation of an octree data structure at each iteration. During simulation, both particle volume and octree cell size can be changed adaptively, conserving mass for a constant density incompressible fluid. We choose to have the octree structure adapt to both the positions and the sizes of the particles. Thus (as described in Section 5), we adapt the particle size/distribution to the needs of the simulation, and then the octree to the particles.

4.1. Relating Adaptive particles and Octree cells

First, we need to discuss the relation between adaptive particles and octree cells. In a sense, these values can be considered independently – the grid size and particle size/distribution do not necessarily have to be related. This is illustrated in prior adaptive particle methods that use fixed grids [HHK08]. However, in general the areas where we want additional (or less) detail in the pressure calculation are the same as those where we want more (less) detail in the mass advection, therefore it makes sense for the grid size to be related to the particle size.

In order to communicate information between the grid and the particles, we need to be able to map velocity information from the particles to the grid and vice-versa (see Section 6). When a size h cell is matched with a radius r particle, where $r = \sqrt{3}h$ (or $r = \sqrt{2}h$ in 2D), each particle can be expected to cover at least eight (four) nodal points on the octree (quadtree) grid. This coverage guarantees that (if all adjacent cells are the same size), a particle will affect all nodes of the grid within the cell in which it lies. This is illustrated in figure 3. We note that although this is a heuristic, it works very well in practice, and better than other values we tested. We discuss next how this heuristic is used to determine the actual octree subdivision.

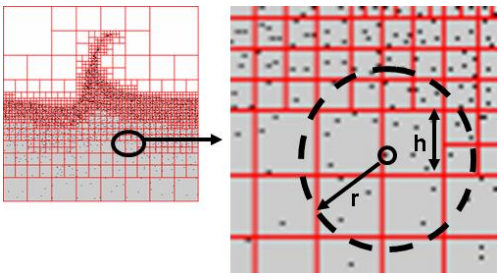

Figure 3: *The size of particles determines the size of an octree cell.*

4.2. Creating the Octree grid

It is straightforward to create an octree grid according to the "target" particle radius. All particles are placed into a single octree node for the simulation, and this node is recursively subdivided to form the octree. In the recursion, each node is examined to determine whether it contains a particle with radius less than the "target" value (i.e. in our implementation, with $r < h/\sqrt{3}$). If two particles in the same cell have different sizes, the cell will be subdivided based on the smaller particle radius. When a cell of the octree is of sufficiently small size, the cell is identified as a leaf node. Typically, there will be only a single particle per leaf node, though it is possible to have more.

Leaf nodes containing particles are identified as fluid cells and the physical values for the cell (velocity and pressure) are initialized from the particles (Section 6).

As the octree is created, the areas near the surface will thus be described by a fine grid (assuming the particles were of small radius). However, for the velocity field to be extrapolated accurately, we must also ensure a fine sampling of the velocity grid on the "air" side of the fluid-air interface. Essentially, we want neighboring air cells to be no larger than the fluid cells. So, we will further process the octree data to add detail in the air near the fluid.

Figure 4 demonstrates the relationship between the particles and the octree cells created.

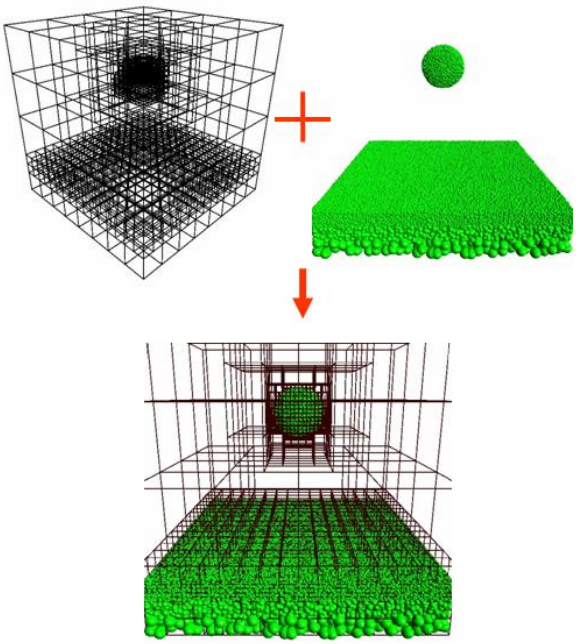

Figure 4: *Relation between adaptive particles and the Octree. cell*

All fluid cells are stored into a list and used to construct this extra detail. For each fluid cell, we determine any neighboring air cells, and repeatedly apply octree subdivision to those cells, until the neighboring cells are as small as the fluid cell. Within this process, we also identify which fluid and air cells form the fluid/air boundary, and store these in a list for subsequent computation.

5. Splitting and merging

In order to adapt our simulation to the areas of most interest, we implement a particle splitting and merging approach. Following previous approaches, we base our splitting and merging on both distance to the surface and on local fluid deformation similar to the Reynolds number. We summarize

our approach here, which offers an improvement over prior adaptive particle methods, and focus on the unique characteristics.

5.1. Splitting and Merging process

The process of splitting and merging particles must preserve mass and momentum. For a constant density incompressible fluid, mass and volume conservation are equivalent.

Merging particles is straightforward; the merged particle's mass and volume are the sum of the smaller particles' mass and volume, while position and velocity are chosen to conserve momentum and center of mass. When we decide to merge particles (Section 5.2), we use a greedy approach. We choose one particle arbitrarily, and merge it with the nearest particle within the particle's radius r. The process is continued with this particle, which will now have a larger radius, until its volume reaches the maximum volume allowed in the region of the particle, or no further particles are within its radius.

Splitting is somewhat more complicated. When splitting, a single particle is divided into two or more particles, each of equal volume/mass, and with velocity equal to that of the parent particle. Given the original particle position, we form a "box" around the particle position, of size equal to the size of a grid cell, h. The new subparticles are generated within this box. In practice, to ensure good distribution of child particles, we subdivide the box into a 2 by 2 by 2 grid of subboxes, and ensure that no particles are placed into the same subbox (we never subdivide into more than 8 particles).

When a particle is split near the surface, we modify the positions of the subparticles to ensure a good sampling remains near the surface. If the parent particle is located within a threshold of the surface, then the new subparticles are projected to surface using the gradient of the distance field (already computed):

$$\vec{N} = \frac{\nabla\phi}{|\nabla\phi|}. \tag{3}$$

5.2. Layers for splitting and merging

With the splitting and merging process available (similar to [DG98] and [APKG07]), we must determine exactly when to split and merge particles. We divide the fluid into a number of "layers," in each of which we determine whether to merge or split particles.

Layers are determined by distance from the surface. Within each layer, we determine whether to split/merge based in part on a local measure of fluid deformation based on the Reynolds number; this determination is similar to the approach of Hong et al. [HHK08], so we will not discuss it further. Generally, each layer has a nominal target particle size, with radius twice that of the layer nearer the surface. Local motion of the fluid within the layer can cause smaller particles to occur in those areas.

We create hysteresis around the layer transition boundary by defining a small range, ε, in which merging (in the "smaller layer") or splitting (in the "larger" layer) are not allowed. This is illustrated in Figure 5. The range of ε is determined by the nominal radius of particles in the farther layer. Specifically, we set $\varepsilon = h$ for the grid cell size, h of the layer with larger particles. This hysteresis is used to prevent particles newly split/merged from immediately being merged/split. For example, splitting a particle could cause some subparticles to be placed in the "larger" layer, and thus immediately merged. Likewise, a merged particle could be positioned within the "smaller" layer and immediately split.

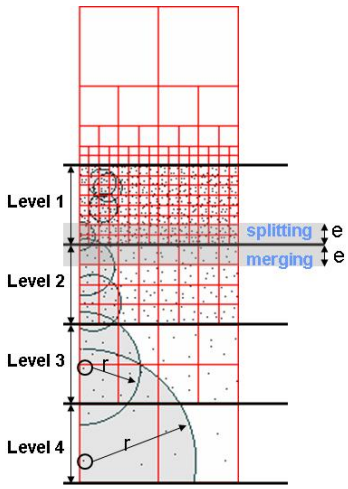

Figure 5: *Layers for splitting and merging.*

6. Kernel choice and transferring particle velocity to octree grid

Once the octree grid is formed, the velocities of particles must be transferred to the grid. To do this, a kernel function is associated with each particle, and the velocity value at each node point is determined by the weighted average of the kernels that overlap it. Likewise, this kernel is used to transfer velocities from the grid to the particle (velocities of the nodes are weighted by the kernel value).

We use a circular kernel function, w:

$$w(r) = (1 - \frac{r}{r_e})^3 \quad (0 \le r < r_e) \tag{4}$$

$$w(r) = 0 \qquad (r_e \le r) \tag{5}$$

where r is the distance from the current particle, and r_e is the current particle radius. The power of 3 in the kernel function was experimentally determined to give somewhat better results than a linear "tent" kernel or a quadratic kernel in

producing detailed splashing effects. Using a circular kernel, rather than a rectangular kernel, makes finding which cell centers are within the particle's radius simpler.

7. Surface tension

Surface tension is physically caused by attraction between molecules. Within the fluid, these attractive forces are canceled out, but near the surface they are unopposed in the direction of the surface. Surface tension J can be described as a force in the negative surface normal direction, with magnitude proportional to the surface curvature, κ_Γ:

$$J = -\sigma\kappa_\Gamma\vec{N} \tag{6}$$

In the hybrid FLIP method, we have two ways to apply surface tension. The prior work for incorporating surface tension, presented by Hong and Kim [HK05], is based on computing these external forces on an Eulerian grid. Likewise, we could compute the tension force by generating an additional MAC grid [HW65] in the FLIP method and using this grid to compute the surface tension forces. However, doing so would require a very high resolution grid and significant additional computational time in order to capture the small detailed features that are the whole point of incorporating surface tension in the first place.

We prefer to instead apply the surface tension force directly within the FLIP method, as we present here. First, surface particles are identified as those within a small depth ε from the surface in the signed distance field. The normal $\vec{N} = \frac{\nabla\phi}{|\nabla\phi|}$ and curvature $\kappa_\Gamma = \nabla \cdot N$ are computed at each of these points, based on the distance function. This approach is similar to that of Müller et al. [MCG03], though with a different distance field. This force is then transferred to the Octree grid along with particle velocities, and is then used along with other external forces (such as gravity and viscosity) when performing the Eulerian force computations. Because this alternative has significantly lower computational cost and makes use of the Lagrangian nature of the approach, we believe it provides a better alternative for surface tension calculation within a hybrid FLIP implementation.

Figure 6 demonstrates the effect that surface tension has within the simulation.

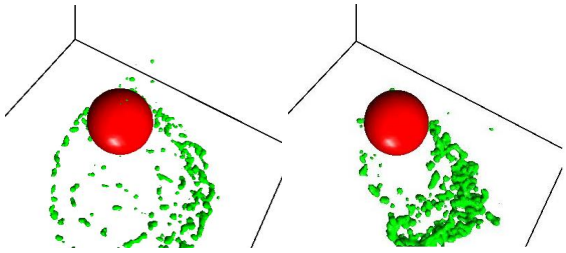

Figure 6: *Demonstration of fluid exhibiting surface tension. It shows two results with surface tension(left) and without surface tension(right). Note that the clustering of the particles is shown in the left picture.*

8. Implementation

We discuss here some further details needed for implementation of this method.

8.1. Constructing the Poisson equation

Solving the Poisson equation [Sta99] is a key step in the fluid simulation. To construct and solve the Poisson equation, we simply use Lossaso's octree based approach [LGF04]. The only difference is that we minimize octree neighbor cell queries by storing all field quantities in the center of each grid cell, speeding some of the computations. In a fixed grid system this could lead to unwanted smoothing of spatial derivatives of the velocity field, since interpolation is required to compute derivatives where they are needed. However, in the spatially adaptive octree formulation, a much finer sampling is supported for the velocity field near the surface, thus reducing the unwanted smoothing in regions of interest. Pressure is computed as follows:

$$V_{cell}\nabla^2 p = \sum_j \left(\frac{p_j - p_i}{\Delta} \cdot n\right) A_{area}\ , \tag{7}$$

where V_{cell} is the volume of the current cell, j, the index of neighbors, n, the outward unit normal of the current cell, Δ, the size of the large cell and A_{size} is the face area of the neighbor cell.

8.2. Divergence

Following the same logic that was used for the pressure gradient, in octree cells the generalized form for the divergence term is

$$V_{cell}\nabla \cdot u = \sum_j (u_j - u_i) \cdot n A_{area} \tag{8}$$

where A_{area} is the area of a cell face, n is the outward unit normal of the current cell and V_{cell} is the volume of the cell in the fluid simulation based on the octree.

8.3. Surface Reconstruction from Mass particles

Our approach for extracting fluid surface is similar to that of Hong et al [HHK08]. Once the interface is detected, the Marching Cubes method [LC87] is used to construct a polygonal isosurface for high quality rendering.

9. Results and Discussion

9.1. Results

We have implemented the algorithm as described, in both two and three dimensions. Results of our implementation can be seen in Figures 1 and 7, as well as in the accompanying video. In Figure 1, the maximum grid resolution is 128^2. 7122 particles were used in the starting configuration. The average number of octree cells was 2419. In Figure 7, the maximum grid resolution is 128^3. 668261 particles were used in the starting configuration. The average number of octree cells was 88033. These two figures show sequences from a 2D and a 3D simulation conducted using our approach, illustrating both the particle sampling and the octree structure. The detail at the surface and the coarsening of the representation below the surface can be clearly seen. Since the simulation is grounded in a particle representation, splashes and spray are handled directly, and actually account for true fluid mass. At the same time, the octree representation provides the fine detail needed around spray particles so that nearby particles continue to form a surface over which surface tension forces can be computed.

9.2. Contrast with Previous Methods

We will briefly contrast our method with the two most similar previous methods.

Note that our approach for merging and splitting is in contrast to that of Adams et al. [APKG07], which can allow some particles of larger size to end up near the surface. While the larger kernel they then use for surface smoothing allows the method to produce smooth results, it also eliminates the small scale surface features that can be vital for producing splashing and similar effects. Since a major goal of our method is to capture fine surface detail (such as from surface tension), we do not adopt their splitting and merging criterion. More fundamentally, that approach is based on SPH, and does not support incompressibility, which is a key feature for realistic water behavior.

Our method is similar to that of Hong et al. [HHK08] in terms of adaptive particle sampling. Compared to that work, our approach differs in providing an adaptive grid structure, the specifics of choosing when to merge/split and how many subparticles to use, and incorporation of surface tension, as well as several implementation-oriented details such as kernel shape.

10. Conclusion

We have proposed a fully adaptive technique for incompressible fluid simulation that uses both adaptive fluid particles and an octree spatial grid. The technique exploits the strengths and avoids the weaknesses of Eulerian and Lagrangian methods, handling advection using a Lagrangian model and the pressure field using an Eulerian model. The method supports both a detailed representation of the fluid surface, and realistic splashing effects. Surface representation can be made fine-grained enough to allow the computation of surface tension forces, adding to the realism of the detail in surface shape and spray particles. Particle splitting and merging are controlled by both depth from the surface and the local flow complexity. Octree depth is determined locally by the size and number of particles. Thus, in deep quiet regions, particles are merged together to form large particles, and correspondingly large cells of the octree grid are created in this region. On the other hand, near the surface, where there is considerable turbulence, particles are split into very small particles and correspondingly very fine octree cells are created.

It is important to note that the benefits of using an octree approach are not based on a time savings. In fact, the overhead for using and updating the octree will cause the adaptive method to run more slowly than an equivalent number of uniform grid cells. However, using an octree enables us to obtain more accurate pressure calculations around the areas of primary interest.

The approach provides a considerable improvement in simulation behavior over other fluid solvers. It is possible to maintain the stability in performance obtainable with Eulerian (semi-Lagrangian) solvers, but without the problems of dissipation and loss of vorticity, and without the problem of volume loss. It is possible to simulate the high detail in surface and splashes that can be obtained with the SPH method, but at the same time to maintain fluid incompressibility. By allowing adaptive particle sizes, the bottleneck of the Lagrangian stage (particle updates) is reduced, at the cost of overhead to maintain the splitting and merging operations. By using an octree grid, the large number of cells that govern performance in the Eulerian stage are reduced, though there is notable additional overhead in the generation and use of the octree itself. The net result is the production of the first fully adaptable (in both the Eulerian and Lagrangian stages) incompressible hybrid fluid simulation.

References

[APKG07] ADAMS B., PAULY M., KEISER R., GUIBAS L. J.: Adaptively sampled particle fluids. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers* (2007), p. 48.

[BR86] BRACKBILL J. U., RUPPEL H. M.: Flip: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. *J. Comput. Phys. 65*, 2 (1986), 314–343.

[CEL06] COLIN F., EGLI R., LIN F. Y.: Computing a null divergence velocity field using smoothed particle hydrodynamics. *J. Comput. Phys. 217*, 2 (2006), 680–692.

[DG98] DESBRUN M., GASCUEL M.-P.: Space-time adaptive simulation of highly deformable substances. In *INRIA Technical Report* (1998).

[HHK08] HONG W., HOUSE D. H., KEYSER J.: Adaptive particles for incompressible fluid simulation. *The Visual Computer 24*, 7-9 (Jul 2008), 535–543.

[HK05] HONG J.-M., KIM C.-H.: Discontinuous fluids. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers* (New York,NY, 2005), ACM Press, pp. 915–920.

[HNB*06] HOUSTON B., NIELSEN M. B., BATTY C., NILSSON O., MUSETH K.: Hierarchical rle level set: A compact and versatile deformable surface representation. *ACM Trans. Graph. 25*, 1 (2006), 151–175.

[HW65] HARLOW F., WELCH J.: Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *J. Fluids. Phys. 181*, 8 (1965).

[IGLF06] IRVING G., GUENDELMAN E., LOSASSO F., FEDKIW R.: Efficient simulation of large bodies of water by coupling two and three dimensional techniques. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers* (New York, NY, 2006), ACM Press, pp. 805–811.

[KCC*06] KIM J., CHA D., CHANG B., KOO B., IHM I.: Practical animation of turbulent splashing water. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, 2006), Eurographics Association, pp. 335–344.

[KTO95] KOSHIZUKA S., TAMAKO H., OKA Y.: A particle method for incompressible viscous flow with fluid fragmentation. *Computational Fluid Dynamics 181*, 4 (1995), 29–46.

[LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques* (New York, NY, 1987), ACM Press, pp. 163–169.

[LGF04] LOSASSO F., GIBOU F., FEDKIW R.: Simulating water and smoke with an octree data structure. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers* (New York,NY, 2004), ACM Press, pp. 457–462.

[MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, 2003), Eurographics Association, pp. 154–159.

[SKK07] SONG, KIM D., KO H.-S.: Derivative particles for simulating detailed movements of fluids. *IEEE Transactions on Visualization and Computer Graphics 13*, 4 (2007), 711–719.

[Sta99] STAM J.: Stable fluids. In *SIGGRAPH '99: ACM SIGGRAPH 1999 Papers* (1999), pp. 121–128.

[SY04] SHI L., YU Y.: Visual smoke simulation with adaptive octree refinement. In *Computer Graphics and Imaging* (2004), pp. 13–19.

[ZB05] ZHU Y., BRIDSON R.: Animating sand as a fluid. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers* (New York, NY, 2005), ACM Press, pp. 965–972.

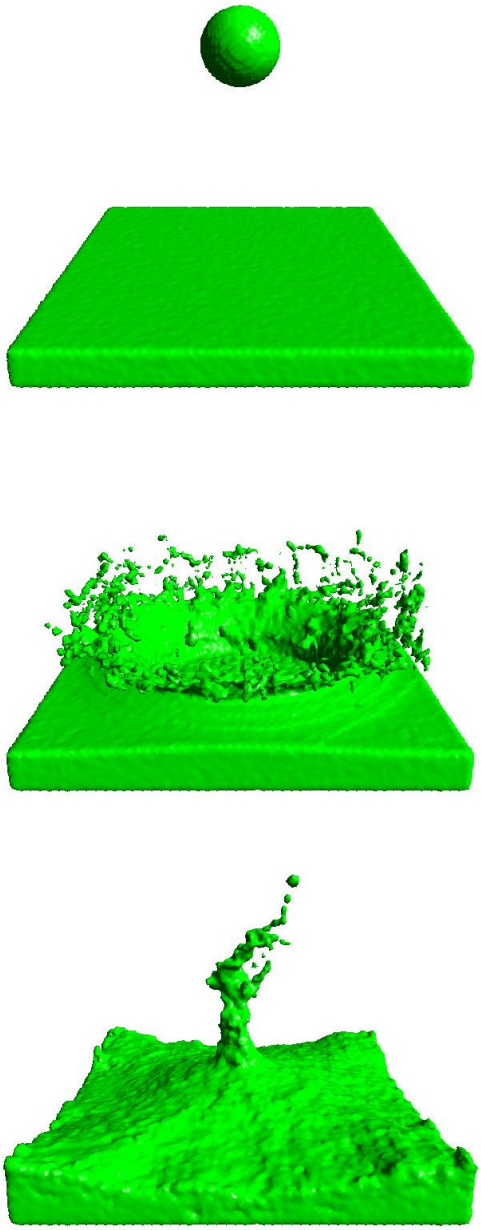

Figure 7: *3D fluid simulation using adaptive particles and an octree grid. The maximum grid resolution is* 128^3. $668,261$ *particles were used in the starting configuration. The average number of octree cells was* $88,033$.