Computer Graphics International manuscript No.
(will be inserted by the editor)

Woosuck Hong · Donald H. House · John Keyser

Adaptive Particles for Incompressible Fluid Simulation

Abstract We propose a particle-based technique for simulating incompressible fluid that includes adaptive refinement of particle sampling. Each particle represents a mass of fluid in its local region. Particles are split into several particles for finer sampling in regions of complex flow. In regions of smooth flow, neghboring particles can be merged. Depth below the surface and Reynolds number are exploited as our criteria for determining whether splitting or merging should take place. For the fluid dynamics calculations, we use the hybrid FLIP method, which is computationally simple and efficient. Since the fluid is incompressible, each particle has a volume proportional to its mass. A kernel function, whose effective range is based on this volume, is used for transferring and updating the particle's physical properties such as mass and velocity. Our adaptive particle-based simulation is demonstrated in several scenarios that show its effectiveness in capturing fine detail of the flow, where needed, while efficiently sampling regions where less detail is required.

Keywords Fluid simulation · Physically based modeling · Natural phenomenon

1 Introduction

It is well known that the complexity of the flow of fluids, especially water, differs depending upon the region of the flow. For example, in large bodies of water, surface flow is generally much more complex than the flow deep below the surface. This concept has been used in Eulerian fluid simulation when developing schemes for grid refinement for detailed behavior and coarsening for efficiency, as in [14]. In a Lagrangian fluid simulation, refinement and coarsening can be done by merging and splitting the particles representing the mass of the fluid. For example, within a Smoothed Particle Hydrodynamics (SPH) simulation, Desbrun and Cani [6] used simplification and refinement of particles to animate soft bodies such as lava.

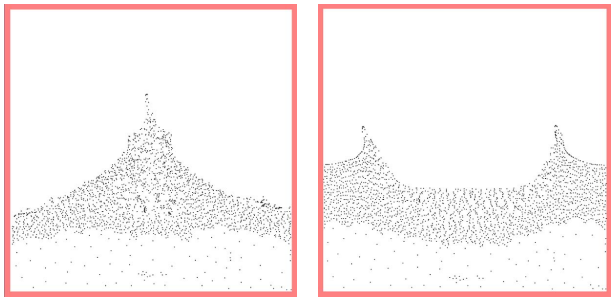


Fig. 1 Fluid simulation with adaptive particle sampling

Particle size adaptation is attractive as a simulation mechanism, since particles provide an efficient means for capturing the small-scale detailed behavior near the surface of a fluid. Even simulations based on an Eulerian grid typically use particles to restore surface detail lost due to smoothing [8]. In Fig. 1 we show examples from an adaptive particle-based fluid simulation, where very fine details of the surface are visible, while in the deep areas, particle merging has been used to avoid oversampling and lend efficiency to the simulation. Although the idea is attractive, it has, to date, only been reported for compressible fluid simulation.

1.1 Our contribution

In this paper, we propose an efficient incompressible Lagrangian fluid simulation method utilizing splitting and merging of particles to adaptively sample the evolving

Woosuck Hong
Dept. of Computer Science, Texas A&M Univ.
E-mail: wshong@cs.tamu.edu

Donald H. House
Dept. of Visualization, Texas A&M Univ.
E-mail: house@viz.tamu.edu

John Keyser
Dept. of Computer Science, Texas A&M Univ.
E-mail: keyser@cs.tamu.edu

fluid motion. The approach uses the hybrid FLIP method [26]. This is an extension to the original work of Desbrun [6], and more recently Adams et al. [2], in which refinement and simplification were performed for compressible flows based on the SPH approach.

The fact that we model incompressible flow makes our method much more suitable than SPH for simulating water. Water is well known to be nearly incompressible, and this fact contributes greatly to its visible behavior. For example, water being poured into a cylindrical glass would maintain an average height proportional to the volume of fluid in the glass. On the other hand, the height of the surface of a compressible fluid would exhibit damped periodic oscillations, giving it an unnatural springy look. To confirm this, one can compare animations in support of the incompressible Particle Level Set Method [8], and those in support of the compressible SPH method [20].

In order to determine the level of particle sampling needed in a region of fluid, we quantify what we call the *deformability* [6] of each region of the fluid and then base particle splitting and merging on this measure. Our method for finding the deformability of a fluid region considers both depth or distance from the surface, and the local Reynolds number [22]. Depth is determined by constructing a signed distance field from the fluid interface as determined by particle positions. The local Reynolds number is estimated on the computational grid, using a ratio of terms from the Navier Stokes equations.

The use of both depth and Reynolds number to obtain a deformability measure allows us to provide the necessary detail to represent both surface shape and complex turbulent flow below the surface, while avoiding over computation in areas where the flow is smooth. Splitting is done in highly deformable areas, and merging in areas of low deformation. With merging, we do not need to assign particles to every grid cell, since the effective radius of a particle can be bigger than the cell width, thus we do not need to worry about occasional gaps in the flow of fluid. Significant memory and computing time savings come from reducing the number of particles in the simulation. We have been able to reduce the number of particles by more than 80% of the number that would have to be used without merging, while producing results that are visually indistinguishable. Likewise, the speed of the simulation is improved significantly.

By using multiple particle sizes, we can also better decouple the size of the auxiliary grid from the size of the particles themselves. The auxiliary grid can be used in the computation of external forces (such as finding curvature and normal vectors when computing surface tension). By supporting multiple particle sizes, we can allow for better computation of these external forces without extra computational cost.

2 Related work

Fluid simulations typically attempt to reproduce the effects of the Navier-Stokes equations that describe fluid flow. A typical representation of the incompressible form of these equations, after some simplification, is:

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

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

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, 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 Eulerian fluid simulation

Eulerian fluid simulations generally solve the Navier-Stokes equation over a grid. Foster and Metaxas [10] were the first in the computer graphics community to achieve full 3D liquid simulation based on the Navier-Stokes equations for incompressible flow. Their simulation method was based on Harlow and Welch's original Marker and Cell (MAC) method [12]. They used a fixed finite-difference grid for computing partial spatial derivatives, an explicit integration scheme for advection and diffusion, a relaxation method for incompressibility, and massless marker particles for surface representation. Stam[24] made significant improvements on the stability of this approach using a semi-Lagrangian scheme to handle velocity advection, implicit integration to handle diffusion, and a Poisson solver for computation of pressure to achieve divergence free flow. Foster and Fedkiw [9] and Enright et al. [8][7] extended Stam's method to handle the fluid-air interface using level set methods augmented by marker particles.

In addition to the above work, several extensions have been attempted to the basic Eulerian method. Greenwood and House [11] simulated fluid with the visual effect of bubbles, Song [23] attempted to reduce the numerical diffusion and dissipation caused by the semi-Lagrangian method by adopting a constrained interpolation profile-based advection scheme. Hong and Kim [13] developed a numerical method to resolve the discontinuities of the interface between two fluids by using the interpolated pressure field and velocity gradient.

2.2 Lagrangian fluid simulation

Particles are the fundamental momentum-carrying simulation element in Lagrangian simulations. In SPH [19], the fluid is composed of a set of particles, with inter-particle forces such as pressure and viscosity computed

at the position of a particle by a smoothing kernel. Desbrun and Cani-Gasculel [5] introduced SPH to computer graphics to simulate highly deformable substances such as lava flow. In [6] refinement of particles by splitting and merging allowed sampling the fluid adaptively. The basic concept of the adaptive sampling scheme for Lagrangian fluid corresponds to Geoffrey's idea [14] who used refinement near the interface for detailed representation and coarsening of fluid cells away from the interface for efficiency in Eulerian fluid. Müller et al. [20] extended the SPH approach for interactive fluid simulation, and introduced a new technique for complex behaviors, such as air-water interaction, boiling water, and trapped air. In a recent paper, Adams et al. [2] have proposed an improved adaptive method for SPH compressible fluid simulation. Though aspects of this paper are similar to ours, we emphasize that our approach handles incompressible fluid, and is in fact the first adaptive particle method to do so.

Premoze [21] introduced Koshizuka's original work [16] on the Moving-Particle Semi-Implicit Method (MPS) to simulate incompressible liquid and multifluid flow. Unlike SPH, MPS handles incompressible fluid. However, the approach is slow, tends towards instability, and is not adaptive. Clavet [4] also extended the formulation of SPH by using a method called double Density Relaxation for enforcing incompressibility and to oppose clustering. Several small-scale simulations of substances such as mud and paint were created realistically with surface tension using springs between particles.

2.3 Hybrid fluid simulation

Zhu and Bridson [26] introduced the hybrid fluid simulation approaches called Particle-in-cell (PIC) [12], and Fluid-Implicit-Particle (FLIP) [3] to computer graphics, in a paper simulating sand flow. Both approaches are fundamentally particle-based, as particles carry the momentum of the fluid, but a MAC grid 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 like MPS. 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-Largrangian method can be avoided, greatly reducing numerical dissipation and loss of flow detail. Kim and Ihm [15] adapted these approaches to water animation, demonstrating realistic turbulent splashing without volume loss.

3 The adaptive fluid simulation algorithm

Our fluid simulation is based on the FLIP method, where particle velocities are transferred to a staggered Marker-and-Cell (MAC) grid, body and diffusion forces are applied, the divergence free property of incompressible fluid is enforced by solving a Poisson equation to obtain a pressure field, and velocities are corrected based on the gradient of the pressure field. The resulting velocity changes are transferred to the fluid mass particles and their positions are integrated through the velocity field. An auxiliary grid is recreated at each time step, serving a computational role only in reconstructing the flow field from the particles. This allows us to avoid costly inter-particle interaction calculations, and provides a more natural field representation for computing pressure.

We have augmented this approach to support merging and splitting of particles. Our overall scheme thus becomes:

- Construct an auxiliary MAC grid around the particles.
- Use the particle velocities to reconstruct a velocity field on the MAC grid.
- Integrate accelerations due to body forces such as gravity and other external forces.
- Integrate accelerations due to diffusion.
- Calculate the pressure field, and project the flow field onto the nearest divergence free field.
- Transfer velocity changes in the flow field back onto the particles.
- Compute a fluid deformability measure based on depth and Reynolds number.
- Split or merge the particles based on the local deformability measure.
- Convect each particle with its velocity through the flow field.

When particle velocity is transferred to the auxiliary grid, the weight of a particle's effect is determined by using a spherical kernel function whose volume is proportional to the mass of fluid represented by the particle.

3.1 Computing deformability

Splitting and merging are adaptively performed based on what we call the fluid's local deformability. In a local region, our deformability measure is based both on depth from the surface, and on the Reynolds number. Since the Reynolds number is the ratio of convection to diffusion, our method requires the fluid to have non-zero viscosity.

3.1.1 Distance from surface

Intuitively speaking, the highly deformable area in a fluid is mainly near the surface, i.e. the interface with another fluid. Areas away from this interface tend to have

smoother flow and are thus less deformable. Further, we wish to be able to reconstruct a smooth surface for rendering, and so we want to be sure to sample finely near the surface. Therefore, we use distance from the surface as our basic criterion for deformability. We compute an implicit representation of the fluid surface directly from the particles, and then use the Fast Sweeping Method [25], which is $O(n)$ in time to construct a signed distance field. This is more easily implemented and faster than the Fast Marching Method [1], which is $O(n \log n)$. Then, the distance from the surface can be computed anywhere inside the fluid using trilinear interpolation.

3.1.2 Reynolds number

In 1883, Osborne Reynolds defined the terms laminar and turbulent flow to describe his experimental results with fluid motion [22]. He proposed the use of a number *Re*, which is the ratio of inertial forces to viscous forces. This number has come to be known as the Reynolds Number. In fluid mechanics, the Reynolds number can be used to identify highly deformable areas of fluid. It is given by

$$Re = \frac{v_s L}{\nu} = \frac{Inertial forces}{Vicous forces}$$

where v_s is the mean fluid velocity, L is the characteristic length, and v is the kinematic fluid viscosity. It is known that the flow rate of turbulence is proportional to the square root of the pressure gradient. That is, high deformation occurs when a high pressure gradient is present, since this will cause high accelerations. We can estimate the square root of the pressure gradient by computing a gradient from the convection term of the Navier-Stokes equation via differencing on the associated grid. We estimate a "deformability factor" roughly proportional to *Re*,

$$Df = \frac{(u \cdot \nabla)u}{\nu \nabla^2 u}$$

by taking a ratio of the convection term and the diffusion term of the Navier Stokes equation 1.

3.2 Determining When to Merge and Split

To determine where it is appropriate to merge and split particles, we divide our simulation domain into a set of layers, based on the distance to the surface. The basic process is illustrated in Fig. 2. The number and placement of these layers is somewhat arbitrary; we want to choose layers that allow detailed simulation (small particles) where needed, and less detail (large particles) where acceptable. Numerous possible schemes could be developed relying on the distance from surface and deformability factor. We describe here a four-layer approach that has proven effective in practice.

Note that in all cases, we must set some minimum and maximum bounds to limit the amount of splitting or merging allowed. Splitting must be limited in order to avoid excessive memory space and noise. Merging must be limited in order to avoid excessive smoothing of particle motion.

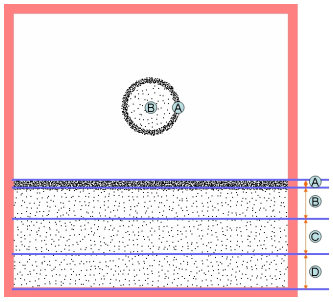


Fig. 2 Layers for Splitting and Merging

Fig. 2 shows the four layers into which we divide the fluid. These layers represent distance from the fluid surface, which is not necessarily the same as depth.

Layer A represents the region closest to the fluid/air interface. We desire the highest resolution in this region, and as a result, we want to use particles of the minimum volume possible. Any particles that enter this region are immediately split to the minimum allowable particle size. This ensures that there is always a highly detailed representation of the surface.

In *Layer B*, all particles are of a nominal, or standard size, which is larger than that in layer *A*. When smaller particles move into layer *B*, merging will be performed, and when larger particles move into this area, splitting will be performed.

Layer C and *Layer D* allow both splitting and merging, based on the deformability measure. The choice of whether to split or merge in these regions is based solely on the deformability factor. Particles are split in higher deformability factor regions and are merged in lower deformability factor regions. Note that instead of layers, one could also easily define an approach where the deformability factor thresholds for merging and splitting are a continuous function of distance from the surface.

3.3 Splitting and Merging

Fig. 3 shows particle splitting on the left, and particle merging on the right. This process must preserve mass, which, for a constant density incompressible fluid, is equivalent to conserving volume, as indicated by the V labels on the figure. One stage of particle splitting splits a single particle into two particles, each of half the volume of the original particle. Particles within some sphere of radius r are merged into a single particle whose volume is

the sum of the original particles. Momentum is conserved using different approaches for merging and splitting.

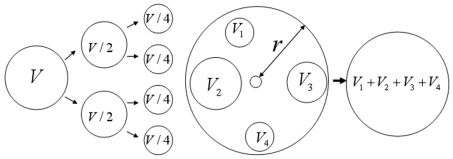


Fig. 3 Mass and volume conservation when splitting and merging particles

3.3.1 Merging

When merging a group of particles, the new particle is placed at the center of gravity of the original particles. The mass, radius, position and velocity of a newly created particle i from neighbor particles are given by

$$m_i = \sum_j m_j, r_i = \sqrt[3]{\sum_j r_j^3},$$
$$x_i = \frac{\sum_j V_j x_j}{\sum_j V_j}, u_i = \frac{\sum_j V_j u_j}{\sum_j V_j}.$$

3.3.2 Splitting

Contrary to merging, splitting of a particle is performed by generating a new set of n particles within the radius of the parent particle. We currently use $n = 2$ but this is arbitrary. Particle mass is distributed evenly among the child particles, and the velocities are simply copied from the original particle, thus conserving momentum. The mass, radius, and velocity of each new particle j, generated by splitting particle i, are given by

$$m_j = \frac{m_i}{n}, r_j = \sqrt[3]{\frac{r_i^3}{n}}, u_j = u_i$$

There are multiple reasonable ways to determine position x_j of the new particles. We choose to place it randomly at a position within the radius of the original particle, using spherical coordinates.

3.3.3 Upper and Lower Bounds

Note that bounds on minimum and maximum size can be set per particle. These maximum and minimum bounds can be set layer by layer (or change continuously) if desired. In practice, we adjust these by layer.

We limit the maximum size of a merged particle so that its radius should not extend beyond the fluid/air interface. A smaller maximum volume is set near the surface, and a larger maximum volume is used in the deep areas of the fluid. The maximum volume is interpreted as an upper limit, beyond which no more merging will occur. We also limit the minimum size of the particles, so that there is not excessive splitting. This minimum volume limit is interpreted as a bound for which no particles will ever be split if it would result in a particle lower than that bound.

Note that these limits restrict the splitting and merging *process*, rather than the sizes of the particles themselves directly. For example, a small particle (below the "minimum" radius) might enter a region, and be unable to merge with any other particle without exceeding the maximum value. In this case, the smaller particle would remain. Likewise, a particle that exceeds the "maximum" might enter a region and be unable to split, since splitting might reduce it below the minimum bound. In our implementation, for example, both of these can occur in layer *B*.

3.4 Kernels and the Auxiliary Grid

When the particle velocities must be reconstructed in the auxiliary grid, a kernel function is used around each particle to determine its contribution to a grid point. Each particle is given a radius that determines the volume of fluid it represents. Each cell point that is within a particle's radius is affected by that particle by an amount determined by distance and the kernel function shape.

We have tested two kernel functions, gaussian and tent. For computational efficiency, we prefer the *tent* kernel function, which is implemented by trilinear interpolation. This simplification does not produce a noticeable effect on the simulation accuracy.

The size of the auxiliary grid can be set independently of the size of the particles themselves. In practice, we usually use a grid cell spacing equal to the "nominal" particle radius of layer *B*. We will call the grid cell width d_g.

To determine velocity values on the grid, we use different schemes depending on the radii of the nearby particles. If the particle radii are generally greater than or equal to d_g (as is generally the case in layers *B* through *D* in our implementation), each sample point on the grid will potentially be affected by multiple particles. In this case, we simply weight the contribution of each particle at the sample point by the value of the kernel function:

$$u_{cell} = \frac{\sum_i W_i(x_{cell}; r_i) u_i}{\sum_i W_i(x_{cell}; r_i)},$$

where the W_j for particle i is defined either radially:

$$W_i(x; r) = K(|x - x_i|; r),$$

or trilinearly:

$$W_i(x; r) = K((x - x_i) \cdot \mathbf{x}; r) K((x - x_i) \cdot \mathbf{y}; r) K((x - x_i) \cdot \mathbf{z}; r).$$

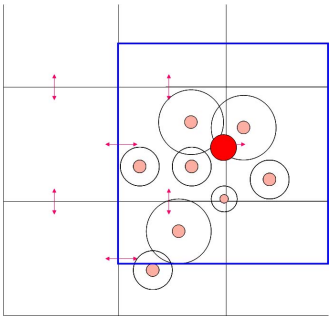


Fig. 4 Sampling of particles with small radii. To determine the radius at the point marked in red, we identify the points that fall within one grid cell width in all dimensions, and use a weighted average of these points.

Note that $K(d;r)$ is the value of the kernel function of radius r at distance d.

When particle radii are near to or smaller than d_g, the sample point might not lie within the volume of any particle, and thus the weighted average described above is not applicable. To avoid such cases, for each sample point we check whether there is a particle with radius less than d_g, within a distance of d_g (in each dimension). If there are such particles, we determine the velocity by taking a volume-weighted average of all such cells, using a d_g radius kernel function for each:

$$u_{cell} = \frac{\sum_i V_i W_i(x_{cell}; d_g) u_i}{\sum_i V_i W_i(x_{cell}; d_g)},$$

where V_i is the volume of particle i. This is illustrated in Fig. 4.

4 Surface Reconstruction from Mass particles

Particles are the fundamental representation in our hybrid method, and must be used to determine the location of the fluid surface. This guarantees that the fluid surface interface will be maintained at the full available level of detail, and that there will be no loss in the volume of the fluid. To do this, a scalar function:

$$\phi(x) = \left(\frac{\sum_i V_i W_i(x; r_i)(1 - \frac{|x - x_i|}{r_i})^2}{\sum_i V_i W_i(x; r_i)} \right)^{\frac{1}{2}},$$

is computed from the particles on an auxiliary grid of the desired surface resolution. The surface interface is simply the zero level set of this function. Once the interface is detected, a signed distance field is easily constructed by fast sweeping, which is exploited for detecting the deformable area in the next step of simulation. The *Marching Cubes Method* [17] is exploited to extract a polygonal isosurface for high quality rendering, and bump artifacts are reduced by smoothing the position and the normal vector of each vertex of the resulting mesh for a few iterations of a relaxation scheme.

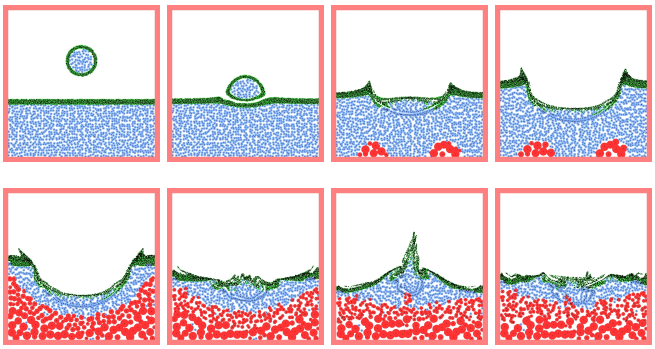


Fig. 5 2D simulation for splitting and merging. Green, blue and red particles are used for split, standard (radius 1) and merged particles. The grid resolution was 30 × 30 × 30.

5 Experimental Results

We have implemented the above process to simulate in both 2 and 3 dimensions. Note that the 2D simulation requires slightly different but straightforward reformulation of some of the listed equations.

Table 1 lists the parameters used in our experiments, including the distance of each layer from the surface, relative to the radius of the "standard" particles in layer B, the Deformability factor values used to determine when splitting and/or merging occur, and the minimum and maximum bounds to use when splitting and merging. The auxiliary grid was set with grid cell size d_g equal to the nominal radius of particles in layer B. For all calculations, constants typical of water were used. These are are kinematic viscosity $\nu = 8.90 \cdot 10^{-7} m^2/s$, with density $\rho = 1000\ kg/m^3$.

Table 1 Parameters used in experiments. For each layer, columns give the distance from surface range (upper and lower bound), the deformability factors above which splitting occurs and below which merging occurs, and the minimum and maximum particle radii (relative to the nominal size) for splitting and merging.

Layer	Lower Bound	Upper Bound	Df Split	Df Merge	Split Min	Merge Max
A	0	1	-	-	0.25	0.25
B	1	4	-	-	1	1
C	4	7	0.3	0.2	3	6
D	7	-	0.1	0.05	5	10

The 2 dimensional water simulation shown in Fig. 5 was sampled with 3,545 particles on an auxiliary 30 × 30 grid. Splitting and merging were performed based on our algorithm for finding the deformable area. The total number of particles was reduced by as much as 61% (to 1,372 particles) by merging during the simulation. The split particles near the interface still support a highly detailed surface representation as shown in the figure.

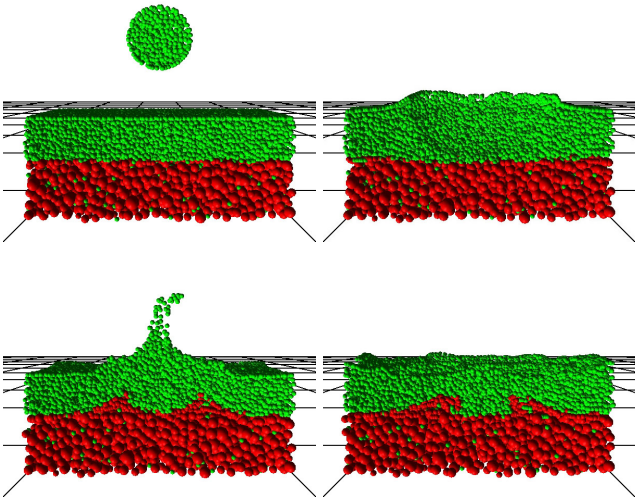


Fig. 6 3D simulation using merging only.

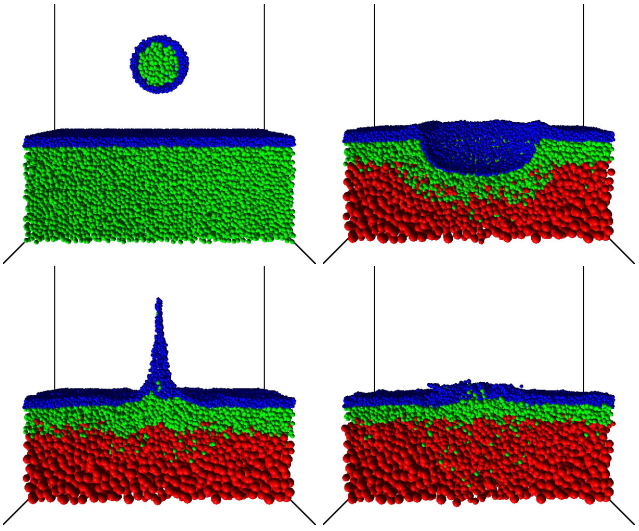


Fig. 7 3D simulation with splitting and merging. Green particles are standard particles which are radius 1. Red particles are the merged particles and blue particles are the split particles with radius less than 1. The grid resolution was 30 × 30 × 30.

The image sequence shown in Fig. 6 demonstrates a three-dimensional water simulation that uses only merging, originally sampled with 87,965 particles and an auxiliary 30 × 30 × 30 grid.

For the animation shown in Fig. 7, we used 113,958 particles in the initial step. Splitting was performed near the surface before the simulation began, in which each particle's volume was split to the minimum size of 0.25 standard particle sizes. Splitting and merging were performed during the simulation. The number of particles was reduced to about 60 percent (69,438 particles) due to merging.

In the 3D simulation shown in Fig. 8, we rendered the surface using the marching cubes technique. This simulation was also performed with splitting and merging. The grid resolution for surface generation was $150 \times 150 \times 150$ and 623,575 particles were generated in the initial step. More than 40 percent of the initial particles were eliminated due to merging.

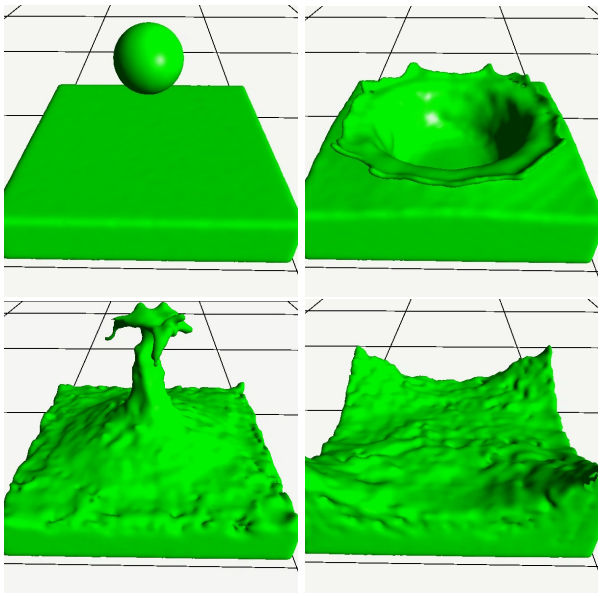


Fig. 8 Animation result with surface-smoothing. The grid resolution for marching cubes was 150 × 150 × 150 and 623,575 particles were generated in the initial step. Merging resulted in more than a 40% reduction in the number of particles.

Table 2 Simulation time table for a 2D simulation similar to Figure 5. NA is the non-adaptive method without our splitting and merging. M1 and M2 are simulations with merging only. AS includes adaptive merging and splitting, giving higher resolution than NA.

	NA	M1	M2	AS
Simulation time(sec.)	0.208	0.167	0.136	0.226

Table 2 presents running times for a sample 2D simulation as in Fig. 5. Timings were the average seconds per frame over a 200 frame sequence, on a Pentium4 1.60 GHz CPU. NA is the timing for a non-adaptive method with no merging or splitting. All particles were of the nominal size, and 1,326 particles were used. M1 presents timing when the particles were merged (but not split) as a preprocess, and no further merging (or splitting) occurred during the simulation. Only 972 particles were used. The M2 simulation starts from the same point as M1, but allows particles to merge (but not split) during the simulation, and the number of particles was eventually reduced to 776. Note that this demonstrates that our particle merging causes noticeable improvements in running time. The AS simulation presents our adaptive sampling approach, including particle splitting. With this approach, we achieve a much finer scale representation of

the fluid surface, at a cost of only about 10% more than the original running time with no splitting or merging.

6 Conclusions

We have presented a novel technique for particle-based fluid simulation that uses mass particle refinement and simplification based on depth and Reynolds number. In order to obtain a detailed representation of the fluid surface, splitting is performed near the fluid surface and splitting or merging away from the surface, depending on the flow complexity. This enables us to save considerable memory space in large scale simulations and improves the speed of the simulation significantly, while maintaining detail where it is needed.

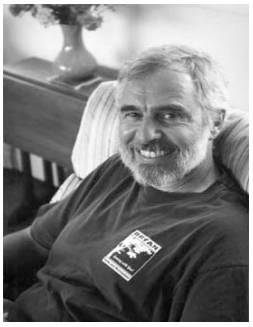
Distance from the surface is used as our basic criterion for determining whether to split or merge particles, and we use a signed distance field for computing this distance. As a complementary criterion, we have proposed a calculation based on the Reynolds number to determine deformability of areas below the surface. Particle splitting and merging are performed adaptively using these criteria.

For our future work, we plan to investigate more faithful capture of the character of the small-scale detail of turbulent water, including splashes and bubbles. To do this, we plan to implement surface tension, using techniques similar to those of Hong [13] and Kim [15]. In addition, we will couple an octree based MAC grid approach (as in Losasso et al. [18]) with our adaptive particle approach, so that appropriate spatial resolution is maintained in all phases of our calculations, further improving the performance of our simulation.

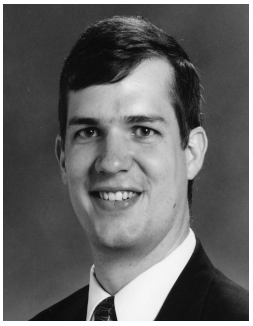
References

1. Adalsteinsson, D., Sethian, J.A.: The fast construction of extension velocities in level set methods. J. Comput. Phys. **148**(1), 2–22 (1999)
2. Adams, B., Pauly, M., Keiser, R., Guibas, L.J.: Adaptively sampled particle fluids. In: SIGGRAPH '07: ACM SIGGRAPH 2007 papers, p. 48 (2007)
3. 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), 314–343 (1986)
4. Clavet, S., Beaudoin, P., Poulin, P.: Particle-based viscoelastic fluid simulation. In: SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation, pp. 219–228 (2005)
5. Desbrun, M., Gascuel, M.P.: Smoothed particles: a new paradigm for animating highly deformable bodies. In: Proceedings of the Eurographics workshop on Computer animation and simulation '96, pp. 61–76. Springer-Verlag New York, Inc., New York, NY, USA (1996)
6. Desbrun, M., Gascuel, M.P.: Space-time adaptive simulation of highly deformable substances. In: INRIA Technical Report (1998)
7. Enright, D., Losasso, F., Fedkiw, R.: A fast and accurate semi-lagrangian particle levle set method. Computers and Structures **83**, 479–490 (2005)
8. Enright, D., Marschner, S., Fedkiw, R.: Animation and rendering of complex water surfaces. In: SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques, pp. 736–744 (2002)
9. Foster, N., Fedkiw, R.: Practical animation of liquids. In: SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pp. 23–30 (2001)
10. Foster, N., Metaxas, D.: Realistic animation of liquids. Graph. Models Image Process. **58**(5), 471–483 (1996)
11. Greenwood, S.T., House, D.H.: Better with bubbles: enhancing the visual realism of simulated fluid. In: SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation, pp. 287–296 (2004)
12. Harlow, F., Welch, J.: Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. J. Fluids. Phys. **181**(8) (1965)
13. Hong, J.M., Kim, C.H.: Discontinuous fluids. In: SIGGRAPH '05: ACM SIGGRAPH 2005 Papers, pp. 915–920 (2005)
14. 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, pp. 805–811 (2006)
15. 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, pp. 335–344. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland (2006)
16. Koshizuka, S., Tamako, H., Oka, Y.: A particle method for incompressible viscous flow with fluid fragmentation. Computational Fluid Dynamics **181**(4), 29–46 (1995)
17. 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, pp. 163–169 (1987)
18. Losasso, F., Gibou, F., Fedkiw, R.: Simulating water and smoke with an octree data structure. In: SIGGRAPH '04: ACM SIGGRAPH 2004 Papers, pp. 457–462 (2004)
19. Lucy, B.L.: A numerical approach to the testing of the fission hypothesis. The Astronomical Journal **82**, 1013–1024 (1977)
20. 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, pp. 154–159. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland (2003)
21. Premoze, S., Tasdizen, T., Bigler, J., Lefohn, A., Whitaker, R.: Particle based simulation of fluids (2003). URL citeseer.ist.psu.edu/article/premoze03particlebased.html
22. Reynolds, O.: An Experimental Investigation of the Circumstances Which Determine Whether the Motion of Water Shall Be Direct or Sinuous, and of the Law of Resistance in Parallel Channels. Philosophical Transactions Series I **174**, 935–982 (1883)
23. Song, O.Y., Shin, H., Ko, H.S.: Stable but nondissipative water. ACM Trans. Graph. **24**(1), 81–97 (2005)
24. Stam, J.: Stable fluids. In: SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques, pp. 121–128 (1999)
25. Zhao, H.: A fast sweeping method for eikonal equations. In: Mathematics of Computation, vol. 74, pp. 603–627 (2004)
26. Zhu, Y., Bridson, R.: Animating sand as a fluid. In: SIGGRAPH '05: ACM SIGGRAPH 2005 Papers, pp. 965–972 (2005)

Woosuck Hong is a Ph.D Student in the Department of Computer Science at Texas A&M University. He received the B.S. and M.S. degrees in Computer Science from Pukyung National University, Korea in 1997 and 2001. His research interests include physically based animation and geometric modeling.



Donald H. House is a Professor in the Department of Visualization at Texas A&M University, where he guided the curricular development of a unique, interdisciplinary Master of Science program in Visualization Sciences. In computer graphics he is best known for the development, with David Breen, of interacting-particle methods for the simulation and visual display of woven cloth. Theirs and the work of a number of others is collected in their volume Cloth Modeling and Animation, published in 2000. Most recently he has been working on novel methods for perceptual optimization, capable of dealing with the complexity of real visualization tasks, and on alternatives to Eulerian methods for fluid simulation. He holds a Ph.D. in Computer and Information Science from the University of Massachusetts, Amherst, an M.S. in Electrical Engineering from Rensselaer, and a B.S. in Mathematics from Union College.



John Keyser is an Associate Professor in the Department of Computer Science at Texas A&M University, where he has been since 2000. He earned his Ph.D. in Computer Science from the University of North Carolina (Chapel Hill) in 2000, and B.S. degrees in Applied Math, Engineering Physics, and Computer Science from Abilene Christian University in 1994. His research focuses on graphics, with particular emphasis on topics related to physically-based simulation, robust geometric computing, real-time techniques, and geometric reconstruction and visualization of biomedical data.