Workshop on Virtual Reality Interaction and Physical Simulation VRIPHYS (2011)
J. Bender, K. Erleben, and E. Galin (Editors)

Time Adaptive Approximate SPH

Prashant Goswami† and Renato Pajarola ‡

Visualization and MultiMedia Lab
Department of Informatics
University of Zürich

(a) (b) (c)

Figure 1: *Examples showing the optimized approximate SPH fluid simulation using (a) up to 200K, (b) 201K and (c) 1M particles with a speed-up of 1.8, 3.59 and 6.94 times, respectively.*

Abstract

In this paper, we present two different techniques to accelerate and approximate particle-based fluid simulations. The first technique identifies and employs larger time steps than dictated by the CFL condition. The second introduces the concept of approximation in the context of particle advection. For that, the fluid is segregated into active and inactive particles, and a significant amount of computation is saved on the passive particles. Using these two optimization techniques, our approach can achieve up to 7 times speed-up compared to a standard SPH method and it is compatible with other SPH improvement methods. We demonstrate the effectiveness of our method using up to one million particles and also compare it to standard SPH particle simulation visually and statistically.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.7]: Three-Dimensional Graphics and Realism—SPH Simulation, Animation, Level-of-detail

1. Introduction

Physically-based fluid simulations have a wide range of applicability in several important domains such as medicine, science, engineering and entertainment. Particle-based Lagrangian methods, such as the *Smoothed Particle Hydrodynamics* (SPH) approach, are particularly attractive for fluid simulation due to their capability to generate small scale detailed fluid motion effects and to handle complex simulation domain boundaries. For realistic fluid visualization, however, a high particle density is required, in particular to resolve fine-scale surface details. Although interactive frame rates can be achieved for a few thousands of particles, ac-

† Email: goswami@ifi.uzh.ch
‡ Email: pajarola@acm.org

DOI: 10.2312/PE/vriphys/vriphys11/019-028

celerating SPH solvers for larger particle counts remains a challenging task.

A key constraint for SPH based simulations is the time step limitation. A well accepted time step limit for low viscosity fluids is defined by the *Courant-Friedrichs-Lewy* (CFL) condition. However, although the CFL condition guarantees convergence and stability in simulation, it is often a too conservative choice. Though techniques have been proposed to employ larger time steps in the context of incompressible and weakly compressible fluid simulations, no general formulation exists that could be applied for fluids with lower stiffness values as well. We present an adaptive heuristic that employs time steps much larger than induced by CFL in certain situations, thereby speeding up the computation while preserving stability.

Many fluid simulations used in virtual environments such as 3D games do not need to guarantee exact physical correctness as long as they can produce a visually convincing and physically plausible effect at a higher speed. To this end, a commonly considered solution to speed up SPH is to use adaptive particle sizes. This, however, comes with its own share of limitations and invariably requires dealing with particles of different sizes. As a second contribution, we address the problem of approximating the physical correctness from a different viewpoint that eliminates the challenges encountered when using variable particle sizes and ensures stability together with acceleration. The presented method segregates particles into active and passive sets based on their location and activity within the fluid. It is on the passive particles that a sizable computational burden can be saved.

The two main contributions of our paper can thus be summarized as follows:

1. Adaptive global time step optimization for low viscosity fluids.
2. Approximated advection based on particle location and activity.

2. Related Work

Fluid simulation using SPH [Mon92, Mon05] was introduced in [MCG03] to simulate and render a few thousand particles interactively, and several approaches have recently been proposed to improve performance. While [SP09] and [IAGT10] can significantly increase the time steps for incompressible and weakly compressible fluids, our approach is rather targeted to improve fluid simulation in general including compressible fluids.

Though the use of larger global time steps could reduce overall computational time, the commonly accepted criteria to set the time step for low viscosity fluids is the CFL condition. However, this is often a rather restrictive estimate. While [DC99] provides some insight into selecting adaptive global time steps based on velocity, force and divergence per iteration, it shows that these criteria might not always lead to an optimal choice. We propose a more general and in fact simple way for choosing adaptive time steps per iteration that alone can bring about a significant performance boost.

Among other methods that work towards performance optimization are parallel GPU accelerated techniques [HKK07, GSSP10] and adaptive particle sizes [SG11, KAJG*06, APKG07, HHK08]. In [ZSP08, HZJ*09, JXZ*09] adaptive particle sizing is integrated with GPU based acceleration. Although GPU algorithms can achieve significant performance improvements in comparison to processor independent methods, the number of particles that can be used for the simulation can be limited by graphics memory.

However, several important issues remain unresolved when using particles with different sizes. Not only do the large particles close to smaller ones inflict on them a high pressure force causing stability problems ([JXZ*09]), non-uniform neighborhood search and time steps must also be dealt with in the implementation. Furthermore, often these methods make use of non-trivial functions, like distance of particles to surface, to carry out merging or splitting operations. Moreover, the density profiles before and after splitting or merging particles are not equivalent anymore.

We present a new approach to efficiently approximate the SPH fluid solver. Our approximation is based on the observation similar to adaptive techniques, that not all particles in a typical simulation play an equal physically relevant role in the global flow and surface reconstruction. The key of our method lies in identifying and separating particles according to their activity levels thereby indirectly introducing a kind of level-of-detail notion. Thereafter, computational efforts can be geared towards the more active particles. This way we completely circumvent the difficult problems faced by simulations using adaptive particle sizes.

3. SPH Basics

In this section, we briefly introduce the fundamentals of SPH as laid out in [Mon92, Mon05]. In SPH, a scalar quantity A is interpolated at location $\mathbf{r}$ by a weighted sum of contributions from nearby particles as given by Equation 1. Here m_j refers to the mass of particle j, ρ_j is its density, $\mathbf{r}_j$ its position and $W(\mathbf{r},h)$ is the smoothing kernel with global support radius h. The gradient of A is obtained by replacing W by its gradient (Equation 2) and a similar formulation exists for the Laplacian (Equation 3).

$$A_i = \sum_j A_j \frac{m_j}{\rho_j} W(\mathbf{r}_i - \mathbf{r}_j, h) \tag{1}$$

$$\nabla A_i = \sum_j A_j \frac{m_j}{\rho_j} \nabla W(\mathbf{r}_i - \mathbf{r}_j, h) \tag{2}$$

$$\nabla^2 A_i = \sum_j A_j \frac{m_j}{\rho_j} \nabla^2 W(\mathbf{r}_i - \mathbf{r}_j, h) \tag{3}$$

We can define the density by replacing A in Equation 1

by ρ, leading to Equation 4. This density is used to compute pressure which in turn is substituted in Equation 7 to give pressure force. Pressure can be derived from the equation of state (EOS) according to [Bat67] by Equation 5 where k is the stiffness constant. When $\gamma = 1$, Equation 5 corresponds to the pressure formulation in [DC96], given by Equation 6.

$$\rho_i = \sum_j m_j W(\mathbf{r}_i - \mathbf{r}_j, h) \quad (4)$$

$$p_i = k\frac{\rho_0}{\gamma}((\frac{\rho_i}{\rho_0})^\gamma - 1) \quad (5)$$

$$p_i = k(\rho_i - \rho_0) \quad (6)$$

$$\mathbf{f}_i^{pressure} = \sum_j m_i m_j (\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}) \nabla W(\mathbf{r}_i - \mathbf{r}_j, h) \quad (7)$$

In addition to pressure forces, viscosity (Equation 8) and gravity forces are also applied to particles. The total force on a particle is obtained by summing up individual contributions from all these forces (Equation 9). The surface of the fluid can be defined by Equation 10, by specifying the threshold on the magnitude of $\mathbf{n}_i$, where $\mathbf{n}_i$ is the gradient of a color field, and extracting all particles above the threshold [MCG03]. For more in-depth survey on SPH formulae and techniques, we refer the reader to [MCG03, Mon92, Mon05]. In our implementation, we have used the smoothing kernels specified in [MCG03].

$$\mathbf{f}_i^{viscosity} = \mu \sum_j m_j \frac{v_j - v_i}{\rho_j} \nabla^2 W(\mathbf{r}_i - \mathbf{r}_j, h) \quad (8)$$

$$\mathbf{f}_i^{total} = \mathbf{f}_i^{pressure} + \mathbf{f}_i^{viscosity} + \mathbf{f}_i^{gravity} \quad (9)$$

$$\mathbf{n}_i = \sum_j \frac{m_j}{\rho_j} \nabla W(\mathbf{p}_i - \mathbf{p}_j, h) \quad (10)$$

4. Global Time Step Optimization

Our first contribution is to introduce a heuristic to select larger time steps for stable simulation. The basic time step formulation in SPH derived from the CFL condition is,

$$\Delta t_{\text{CFL}} = \alpha \cdot \frac{h}{c} \quad (11)$$

$$c \approx \sqrt{k} \quad (12)$$

where h is the global support radius, c is the velocity of sound propagation in the fluid medium, usually given by Equation 12 where k is the stiffness constant, and the constant α is set to 0.4 as per [Mon92]. This formulation basically ensures that information propagating through the medium at velocity c cannot skip the discretization distance h in a single time step Δt.

As is obvious from the formulation of Equation 11, the CFL condition does not take into account the particle motion itself, their velocity or force to compute a tighter approximation for the time step. In order to consider particle dynamics to determine the time step, Equations 13 and 14 have been proposed in [DC96, BT07]. Here f_{max} refers to maximum force per unit mass on a particle, $\nabla \cdot v$ to the divergence of velocity, and $\beta = 0.25$ and $\gamma = 0.005$ are user chosen constants. The final time step in Equation 15 is thus determined by the minimum of Equations 11, 13 and 14:

$$\Delta t_f = \beta \sqrt{\frac{h}{f_{\text{max}}}} \quad (13)$$

$$\Delta t_v = \frac{\gamma}{\|\nabla \cdot v\|} \quad (14)$$

$$\Delta T = \min(\Delta t_{\text{CFL}}, \Delta t_f, \Delta t_v) \quad (15)$$

Figure 2(a) compares the time steps obtained for each iteration using the above three equations for a fluid with $k = 1000$. Whereas Δt_v mostly underestimates the time step, Δt_f overestimates with respect to the CFL condition which might lead to instability or shock waves. For low viscosity fluids, the term incorporating the speed of sound dominates the force factor and Equation 13 can be ignored. Therefore, Equation 15 will not really provide a tight estimate of current time step based on velocity or force on particles.

[Bri09] offers a slightly more robust treatment of the CFL condition by suggesting the modification in Equation 16. Here v_{max} is the maximum particle velocity value in the simulation, F is the total force acting on a particle and V_{max} is the largest predicted velocity value of a particle obtained. This solution is slightly more robust because it takes into account the effective force during the current time step.

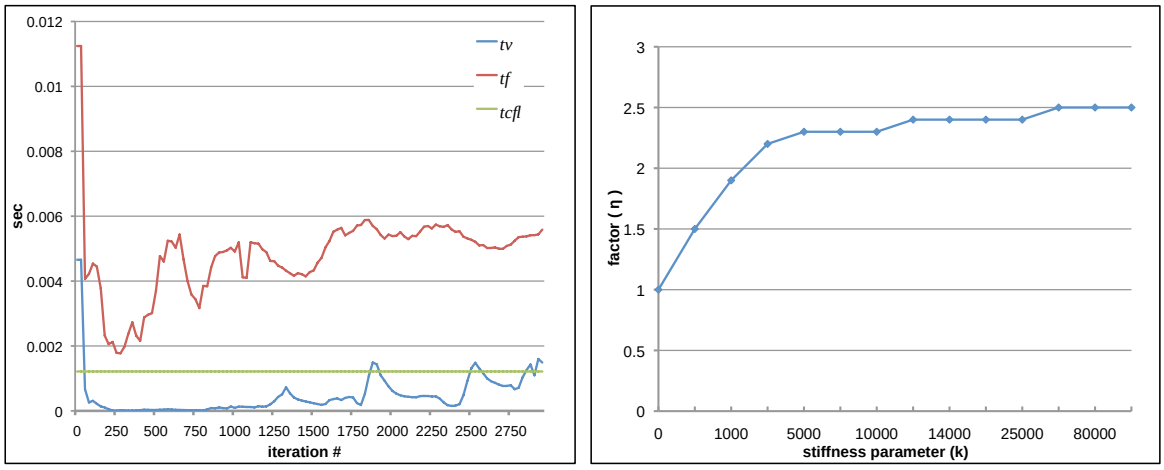
$$V_{max} = v_{max} + \sqrt{h \cdot F} \quad (16)$$

We refine this idea further and overestimate V_{max} (Equation 17) by also including the maximum particle force value, F_{max} in the simulation. The resulting velocity V_{max} is then used to obtain a larger estimate on the global time step.

$$V_{\text{max}} = v_{max} + \sqrt{h \cdot F_{max}} \quad (17)$$

Algorithm 1 outlines our heuristic to choose the time step for each iteration. Here c refers to the relevant speed which is usually taken to be $\sqrt{k}$ and $\alpha = 0.4$ ([Mon92]). The basic motivation is to detect the simulation condition where a larger time step can be allowed (line 4 in Algorithm 1). Under such a circumstance, one can safely use a time step η times larger than dictated by the CFL condition without introducing instability or shock waves. η itself is a scalar factor dependent on the stiffness constant of the fluid and is experimentally approximated by the curve in Figure 2(b).

The proposed heuristic employs a binary choice for the time step: using the conservative CFL condition, Δt_{CFL} time step when the fluid wave is traveling fast, i.e $V_{\text{max}} > \alpha \cdot c$, and $\eta \cdot \Delta t_{\text{CFL}}$ otherwise. For fluids with larger stiffness constants, up to 2.5 times larger time steps can be used, also see Figure 2(b). In PCISPH [SP09, IAGT10], one could choose time steps several times larger than directly derived from CFL condition. However, the presented formulation is par-



(a) (b)

Figure 2: *(a) Time step ΔT selection over time using Equations 11, 13 and 14. (b) Approximate value of scale factor η in relation to stiffness constant k.*

ticularly beneficial to reduce the overall computational time for compressible fluids where the same does not hold true.

Algorithm 1 Time step selection

1: Obtain $v_{\max}$
2: Obtain $F_{\max}$
3: $Compute: V_{\max} = v_{\max} + \sqrt{h \cdot F_{\max}}$
4: **if** $(V_{\max} < \alpha \cdot c)$ **then**
5: $\quad \Delta T = \eta \cdot \Delta t_{\text{CFL}}$
6: **else**
7: $\quad \Delta T = \Delta t_{\text{CFL}}$

Algorithm 2 SPH Algorithm

while animating **do**
  for all particles i **do**
    find neighborhood $N_i(t)$
  for all particles i **do**
    compute density $\rho_i(t)$
    compute pressure $p_i(t)$
  for all particles i **do**
    compute forces $\mathbf{F}^{p,g,v}(t)$
  for all particles i **do**
    compute new velocity $\mathbf{v}_i(t+1)$
    compute new position $\mathbf{p}_i(t+1)$

5. Acceleration by Approximation

In SPH, particles are the information carriers. The movement of particles changes density, which in turn induces pressure forces based on which the particles are moved. The basic steps in SPH computation are given in Algorithm 2. After each iteration, all particles near the surface are used for detailed surface reconstruction and rendering. A large particle count is in particular required for the rendering of high quality detailed fluid surfaces. However, in order to maintain physical correctness and stability, normally all uniformly sized particles of the densely sampled fluid are processed by the computational simulation loop in Algorithm 2. This applies equal computational cost to all particles irrespective of their positions or activity levels within the fluid, hence unduly increasing the overall processing cost.

Our second contribution focuses on optimizing the computational burden based on particle location and movement. This comes from the observation that not all particles in any given iteration equally influence the global flow of the fluid. The movement of some particles might create significant changes in the visual and physical details within a few frames, whereas not much difference might be noticed for others. However, the latter kind still continues to claim computational resources since their updated densities and positions are required for the overall stability of the simulation. Seemingly obvious and plausible optimizations, like skipping neighborhood searches or reusing velocities or forces from the last iteration for particles with low activity are not really scalable, and we have experienced that these can quickly lead to unstable simulations. The challenge therefore, is to determine a way to utilize the inactivity of particles in certain regions without using variable particle sizes or other techniques that are critical to the physical simulation stability.

To cope with these problems while still leveraging the non-uniform activity of particles across the simulation, we adopt another approach. In order to save on the computa-

tional cost of these passive particles, we set them apart in each iteration from the still active ones. These inactive particles have the following two properties:

1. Their movement does not contribute to a noticeable difference in the visual details (especially at the surface) of the fluid during a few time steps.
2. Their movement does not affect significantly the global flow of neighboring particles, i.e they exhibit rather small local movements.

Our approach now is to temporarily restrict the movement of inactive particles and make such particles stationary until they become reactivated. This results in temporarily stationary, non-moving zones of particles within the fluid, which are rather in the interior and not usually at the surface. The idea is somewhat similar to *freezing* or *sleeping* or *deactivating* in rigid body simulations [Sch02] wherein bodies that have come to rest in simulation are fixed at their current place and their state is not simulated anymore. However, in our case inactive zones do not stay static for long and can be reactivated upon new nearby fluid motion.

The activity status of a particle can be decided using Equation 18 where v_i refers to the magnitude of velocity of a particle and n_i to the magnitude of gradient of the color field (Equation 10). V_{cutOff} and N_{cutOff} are corresponding user chosen thresholds. That way no particle is ever treated as passive if it is either moving faster than a certain velocity or if it is near the boundary of the fluid. Here we have chosen $\mathbf{n}_i$ as a metric to select surface particles. One could however use it in combination with the number of neighbors threshold for a better selection. Alternatively, a particle could be defined to be at the surface if the distance to the center of mass of its neighborhood is above a given threshold as given in [SZP07].

$$\mathbf{p}_i.\text{active} = (v_i \geq V_{\text{cutOff}} \ \textbf{OR} \ n_i \geq N_{\text{cutOff}}) \qquad (18)$$

The proposed method, Algorithm 3, starts with finding all currently active particles first in each iteration. In the next step, each active particle with velocity $v_i \geq V_{\text{cutOff}}$ polls its neighborhood and sets the status of all neighboring particles to be active. By doing so, we prepare neighboring inactive particles to become active again by updating their velocity. This step makes sure that no significant fluid activity is lost. This completes $A(i)$, the set of all active particles.

In principle, all remaining particles are passive and will not be advected during the given iteration. However, some of these inactive particles are in close proximity of active ones and hence their densities and pressures are needed, see also Figure 3. We thus define the set $SA(t)$ of semi-active particles for which we also compute neighborhoods, as well as densities and pressures, but force computations are skipped on them. All particles not in $A(t) \cup SA(t)$ constitute the set $P(t)$ of passive particles for which neither neighborhood nor density or force calculations are performed, thereby saving on the computational cost.

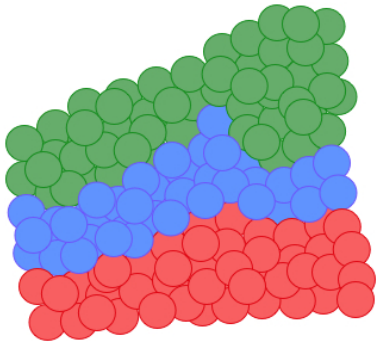


Figure 3: *Semi-active particles (blue) separate active (green) and inactive (red) particles forming an implicit virtual boundary.*

Algorithm 3 Accelerated SPH Algorithm

```
1:  Initialize global time step ΔT = Δt_CFL
2:  while animating do
3:      find all active particles A(t) according to Equation 18
4:      for all particles i in A(t) do
5:          find neighborhood N_i(t)
6:          if v_i(t) ≥ V_cutOff then
7:              mark all particles in N_i(t) active
8:              A(t) ← A(t) ∪ N_i(t)

9:      make all non-active neighbors of A(t) semi-active, SA(t)
10:     mark all remaining particles as passive, P(t)

11:     for all semi-active particles i in SA(t) do
12:         find neighborhood N_i(t)
13:     for all particles i in A(t) and SA(t) do
14:         compute density ρ_i(t)
15:         compute pressure p_i(t)
16:     for all particles i in A(t) do
17:         compute forces F^{p,g,v}(t)
18:     for all particles i in A(t) do
19:         compute new velocity v_i(t+1)
20:         compute new position p_i(t+1)

21:     if required then
22:         adjust V_cutOff

23:     update ΔT using Algorithm 1
```

Marking neighboring particles as active (line 7) or semi-active (line 9) can be combined with neighborhood computation of particles in $A(t)$. This way we can avoid running the same loop twice for particle selection. Furthermore, we skip neighborhood computation for passive particles. It should be noted that the neighborhood search is one of the most expensive parts in the entire SPH routine, see also [APKG07].

Since the choice of active particles has to be made in the beginning of the Algorithm 3 (line 3), we make use of color values from previous frame to decide particle activity in Equation 18. By a similar logic, inactive particles skip color computation (and hence neighborhood computation) till they are reactivated by a neighboring active particle.

In our accelerated approximate SPH method, both semi-active and passive particles do not move. Such particles are static until reactivated by a propagating nearby fluid activity. We achieve this by trimming the standard SPH routines to affect the different particle categories accordingly. Whenever a passive particle is reactivated, it restarts advection with its last active known velocity.

Since in each iteration every active particle adjusts its state with respect to all its neighbors, whether active or passive, simulation never gets unstable. We observe though that for the semi-active and passive particles Newtons' third law is not obeyed since we do not use symmetric and opposite forces to move them. However, this is similar to collision on boundaries, the semi-active particles temporarily form virtual boundaries between advected and static fluid regions. The computational speed-up is obtained at the expense of some momentum loss in each iteration. This is comparable to ignoring smaller higher-order coefficients in a polynomial evaluation and hence is a numerical approximation.

The complete outline of our accelerated SPH fluid solver is given in Algorithm 3. Note that at the end of each iteration, the maximal global time step ΔT is determined using Algorithm 1. Optionally V_{cutOff} can be adjusted if there are too few active particles (line 22).

6. Results

The proposed method is implemented in C++ on Mac OS X, with 2.8GHz Quad-Core Intel Xeon hardware and 4 GB 800MHz DDR2 RAM. All the images are generated offline with POVRAY using the particle positions from the simulation.

The graph in Figure 4 shows the per iteration time step ratio obtained using our heuristic (Algorithm 1) on the typical *falling block of water* simulation example with stiffness value $k = 1000$ and 100K particles. It can be noticed that a time step as large as η times Δt_{CFL} can be used for many iterations thereby speeding up the simulation by a factor approaching η overall, even for compressible fluids with pretty low stiffness values.

In Table 1 we demonstrate the performance gain of our approach over standard SPH. With adaptive time stepping alone, one can achieve speed-up of a factor close to η and sometimes even higher without changing the simulation at all. The performance is then further compared to using adaptive time stepping with approximation on particle movement. We achieve a maximal speed-up of a factor close to 7 for 1M particles and the performance gain improves with particle count and stiffness parameter.

For each demo scene, the initial V_{cutOff} value is specified. Starting with this higher value, V_{cutOff} is incrementally adjusted if the number of active particles are below some threshold. In our experiments, this higher value of V_{cutOff} is initially set to an arbitrary value $\leq 0.025\%$ of $\sqrt{k}$. Each time the number of active particles reach below certain percentage, V_{cutOff} is reduced by 1% subjected to a lower threshold which is set to be around 0.003% of $\sqrt{k}$ in our experiments. However, these values of V_{cutOff} can be easily altered depending on how much damping is tolerable. On the other hand, N_{cutOff} is set once initially and remains constant throughout the simulation. A simple threshold can be set for n_i to select surface particles, also see [MCG03].

Our approach compares well to prior (CPU based) adaptive particle results as reported in [APKG07] which reaches a 4.3 times gain for their largest 630K particle original size armadillo model. Our method reaches a 5 times speedup for the 512K particle falling block setup and a nearly 7 times speedup for the large 1M particle simulation. This is also quite in agreement with the two-scale approach presented in [SG11] where a 6.7 times speed-up is obtained for 2.8M particles. Since in our case acceleration increases with the particle count, we expect speed-ups higher than 7 for more than 1M particles. Note that our approach does not suffer from handling adaptive particles (i.e. problematic density profiles, merging/splitting particles, stability problems or boundary handling) as well as it does not depend on complex functionality (i.e. adaptive distance and search functions) and is thus comparably much easier to implement. Furthermore, in addition to speeding up standard SPH, our approximation approach could also be integrated into PCISPH. For this, one just needs to include density error additionally in Equation 18. This would reduce computational burden to fix the density error once the passive particles are reactivated.

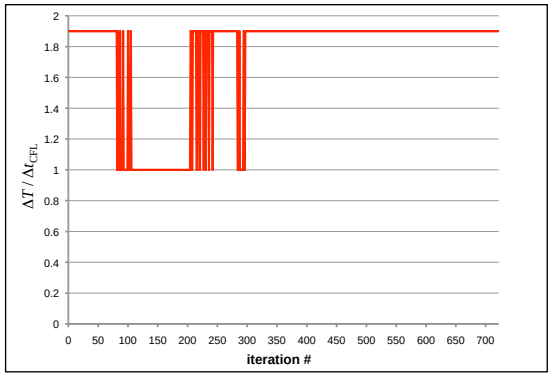


Figure 4: *Per iteration $\Delta T / \Delta t_{\text{CFL}}$ time step ratio for 100K particles using stiffness value 1000.*

Particles	Demo setup	k	η	V_{cutOff}	Standard SPH	Adaptive ΔT		Adaptive ΔT + Approx.	
					Time	Time	Speed-up	Time	Speed-up
200,000	Water pipe with collisions	500	1.5	0.1	24243	15498	1.56	13462	1.8
110,592	Simple water block	1000	1.9	0.09	21272	12086	1.76	6307	3.37
201,348	Water block with collisions	2000	2.2	0.35	58438	26103	2.24	16263	3.59
512,000	Simple water block	1000	1.9	0.1	274630	155158	1.77	54305	5.06
1,000,000	Simple water block	1000	1.9	0.09	927740	509747	1.82	133772	6.94

Table 1: *Comparison of performance between standard, time-adaptive and time-adaptive plus approximated SPH for various particle counts and stiffness values k.* η *is the adaptive time step factor employed in Algorithm 1. All simulation times are given in seconds.*

Figure 5 demonstrates the frames obtained from our particle simulation with four different demo setups and different stiffness constants and particle counts. Figure 6(b) depicts the configuration of semi-active (blue) and passive (red) particles for the displayed simulation in Figure 6(a). The zones of inactive particles move depending on where the particle activity is less. As one can observe from the supplemental video, our optimized and approximated method visually behaves almost indistinguishably from the standard SPH simulation. In Figure 7 we compare the visual difference between standard, time adaptive and time adaptive plus approximated SPH simulations for corresponding frames. It can be noticed that all three appear to be virtually identical, with only very minor differences.

7. Conclusions

In this paper we have presented two techniques to accelerate the standard SPH method. Our global time step selection produces results practically equivalent to the CFL conditioned time step but at a much higher speed, especially in the context of compressible fluids. Furthermore, our particle update optimization introduces an additional performance boost at the expense of advection approximation in the simulation, but still keeps the simulation stable and visually equivalent while circumventing the challenges imposed by adaptive particle size models.

The main limitation of our method is that it can achieve higher speed-ups only if the fluid motion has sizable still regions or eventually settles down. Setting much higher values of V_{cutOff} in such cases can lead to pronounced damping giving the simulation an artificial look. One potential future work along this line could be to assign macro movements to the semi-active and inactive particles in chunks such that all computations for particles within a chunk can be avoided. Further, a theoretical and tighter estimate for choosing η, the speed-up factor in global time step optimization can be investigated.

8. Acknowledgements

We would like to thank Barbara Solenthaler and Maxim Makhinya for their help in image generation and video editing respectively. Also, we are grateful to the reviewers for their valuable comments and suggestions.

References

[APKG07] ADAMS B., PAULY M., KEISER R., GUIBAS L. J.: Adaptively sampled particle fluids. *ACM Transactions on Graphics 26*, 3 (July 2007), 48–54. 2, 5, 6

[Bat67] BATCHELOR G.: *An Introduction to Fluid Dynamics*. Cambridge University Press, 1967. 3

[Bri09] BRIDSON R.: *Fluid Simulation For Computer Graphics*. A.K. Peters, 2009. 3

[BT07] BECKER M., TESCHNER M.: Weakly compressible SPH for free surface flows. In *Proc. of the Eurographics/ ACM SIGGRAPH Symposium on Computer Animation* (2007), pp. 63–72. 3

[DC96] DESBRUN M., CANI M.-P.: Smoothed particles: A new paradigm for animating highly deformable bodies. In *In Eurographics Workshop on Computer Animation and Simulation* (1996), pp. 61–76. 3

[DC99] DESBRUN M., CANI M. P.: *Space-Time Adaptive Simulation of Highly Deformable Substances*. Technical report, Caltech, iMAGIS, 1999. 2

[GSSP10] GOSWAMI P., SCHLEGEL P., SOLENTHALER B., PAJAROLA R.: Interactive SPH simulation and rendering on the GPU. In *Proceedings ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2010), pp. 55–64. 2

[HHK08] HONG W., HOUSE D. H., KEYSER J.: Adaptive particles for incompressible fluid simulation. *The Visual Computer 24*, 7 (July 2008), 535–543. 2

[HKK07] HARADA T., KOSHIZUKA S., KAWAGUCHI Y.: Smoothed particle hydrodynamics on GPUs. In *Proceedings Computer Graphics International* (2007), pp. 63–70. 2

[HZJ*09] HE Y., ZHANGYE W., JIAN H., XI C., CHANGBO W., QUNSHENG P.: Real-time fluid simulation with adaptive SPH. *Computer Animation and Virtual Worlds 20*, 2 (June 2009), 417–426. 2

[IAGT10] IHMSEN M., AKINCI N., GISSLER M., TESHNER M.: Boundary handling and adaptive time-stepping for PCISPH. In *Workshop on Virtual Reality Interaction and Physical Simulation* (2010), pp. 79–88. 2, 3

[JXZ*09] JIAN H., XI C., ZHANGYE W., CHEN C., HE Y., QUNSHENG P.: Real-time fluid simulation with complex boundaries. *The Visual Computer 26*, 4 (April 2009), 243–252. 2

[KAJG*06] KEISER R., ADAMS B., J. GUIBAS L., DUTRÉ P., PAULY M.: *Multiresolution Particle-Based Fluids*. Technical Report 520, ETH Zurich, Switzerland and KU Leuven, Belgium and Stanford University, 2006. 2

[MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fuid simulation for interactive applications. In *Proceedings Eurographics/ACM Symposium on Computer Animation* (2003), pp. 154–159. 2, 3, 6

[Mon92] MONAGHAN J.: Smoothed particle hydrodynamics. *Annual Review of Astronomy and Astrophysics 30* (1992), 543–574. 2, 3

[Mon05] MONAGHAN J.: Smoothed particle hydrodynamics. *Rep. Prog. Phys. 68* (2005), 1703–1759. 2, 3

[Sch02] SCHMIDL H.: *Optimization-Based Animation*. PhD thesis, The University of Miami, 2002. 5

[SG11] SOLENTHALER B., GROSS M.: Two-scale particle simulation. *ACM Transactions on Graphics 30*, 4 (2011), 72:1–72:8. 2, 6

[SP09] SOLENTHALER B., PAJAROLA R.: Predictive-corrective incompressible SPH. *ACM Transactions on Graphics 28*, 3 (2009), 40:1–6. 2, 3

[SZP07] SOLENTHALER B., ZHANG Y., PAJAROLA R.: Efficient refinement of dynamic point data. In *In Proc. of the Eurographics Symposium on Point-Based Graphics* (2007), pp. 65–72. 5

[ZSP08] ZHANG Y., SOLENTHALER B., PAJAROLA R.: Adaptive sampling and rendering of fluids on the GPU. In *Proceedings Eurographics/IEEE VGTC Symposium on Point-Based Graphics* (2008), pp. 137–146. 2

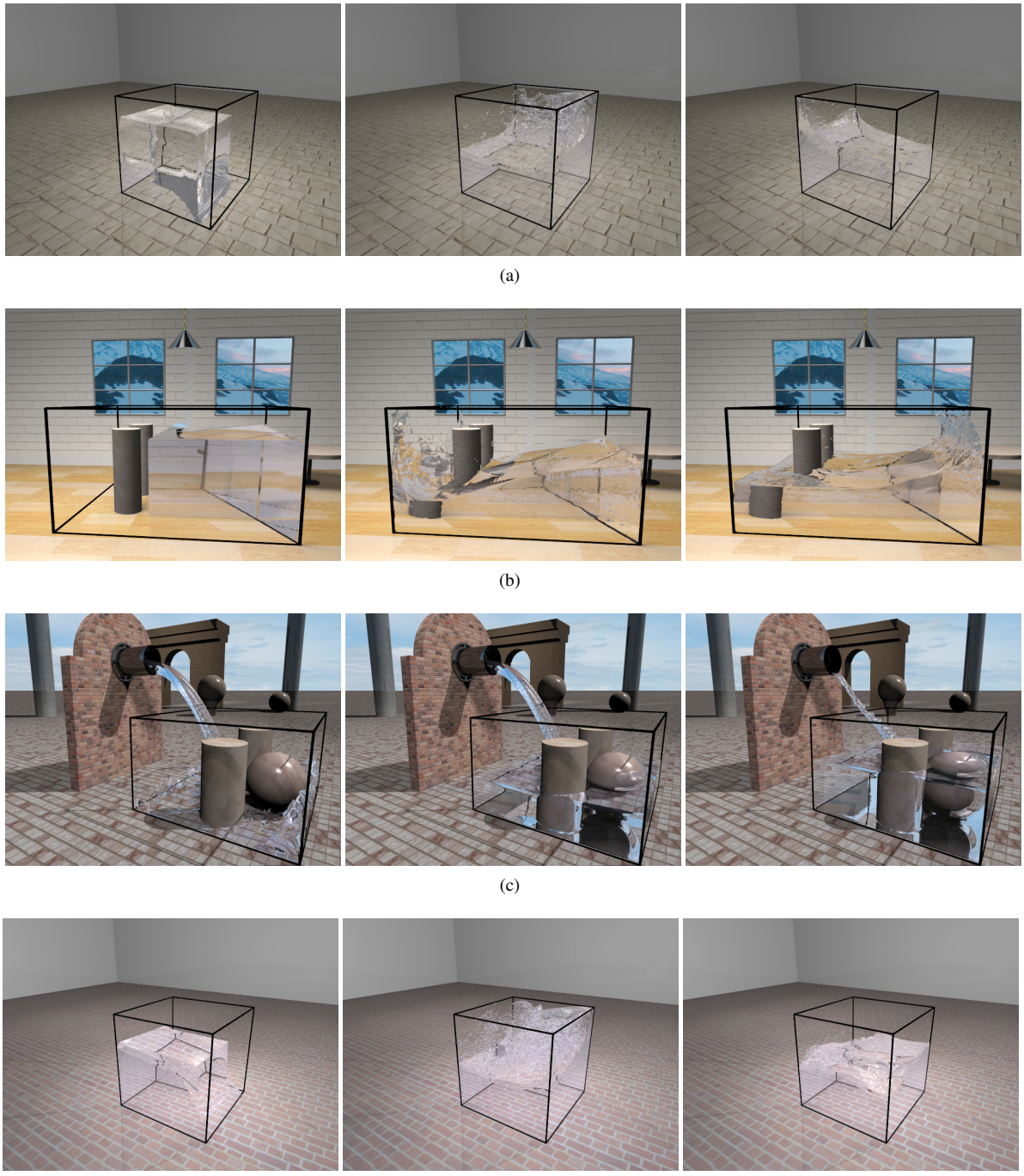
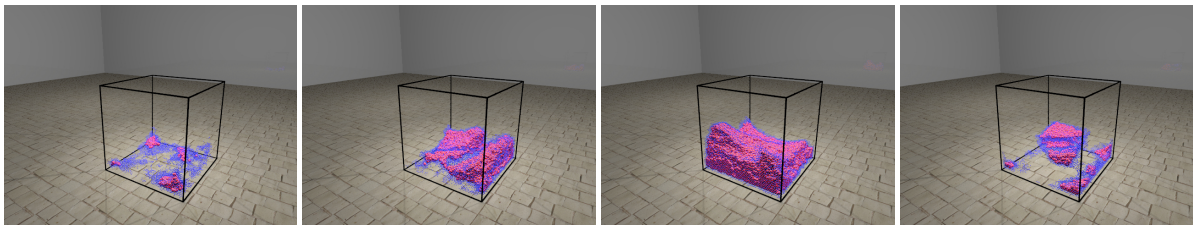


Figure 5: *Fluid simulation with the optimized SPH using (a) 100K particles simple water block, (b) 201K particles water block with collisions, (c) up to 200K particles water pipe with collisions, and (d) 1M particles simple water block. Figures (b) and (c) depict collision with cylindrical as well as spherical objects in addition to domain boundaries.*

(a)

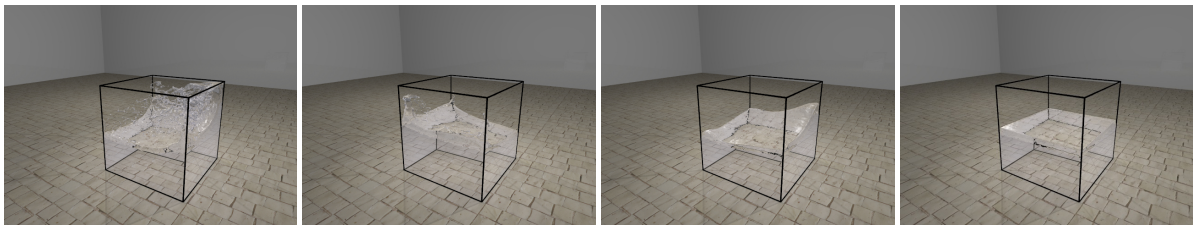


(b)

Figure 6: *(a) Surface, and (b) corresponding semi-active (blue) and passive (red) particles for different time steps.*

(a) Standard SPH

(b) Time Adaptive SPH



(c) Time Adaptive + Approximated SPH

Figure 7: *Visual comparison between (a) standard, (b) time adaptive and (c) time adaptive plus approximated SPH for the same time steps.*