Adaptive Neighbor Pairing for Smoothed Particle Hydrodynamics

Brandon Pelfrey and Donald House

Clemson University

Abstract. We present a technique for accelerating Smoothed Particle Hydrodynamics (SPH) by adaptively constructing and reusing particle pairing information. Based on a small calculation performed on each particle pair, we can determine whether or not pairing information needs to be recomputed at each simulation frame. We present simulations that show that for numbers of particles above 8,000, total simulation computation time is less than one-third of that of the non-adaptive algorithm. In addition, our simulations demonstrate that our algorithm produces simulation results that are visually very similar to the unoptimized version and that visual error does not accumulate as the simulation progresses.

1 Introduction

Smoothed Particle Hydrodynamics (SPH) has become a popular method, in the computer graphics community, for simulating free surface flow. Being a *Lagrangian* technique, SPH tracks the fluid using a set of particles, each of which represents a chunk of material with a constant mass. SPH has the advantage, over grid-based Eulerian methods, of being able to easily track complex surface features without the use of a level set and allows for complex splashing and surface tension effects.

The structure of the SPH algorithm requires that fluid particles exert forces upon their neighbors at every time step. Because of this, the time complexity of a naive implementation of the algorithm is $O(n^2)$ in the number of particles, which is too poor even for a moderately-sized system. Therefore, the representation of fluid forces in SPH typically uses weighting kernels of compact support, so that it is only necessary for a single particle to consider neighbors lying within the radius of its weighting kernel. Then if local particle density is bounded it is possible to use uniform grids to turn the system into an $O(n)$ algorithm for a fixed density. Unfortunately, in this case, the algorithm becomes $O(d^2)$, where d is the particle density, which is directly proportional to the maximum number of particles in a grid cell. Thus, for SPH simulations, finding neighboring particles, a process which we call neighbor pairing, is often the most compute intensive component of the SPH algorithm [2]. In this paper we present a technique that alleviates the need to compute this information for each particle at each frame and instead computes lists of particle pairs which are then adaptively updated when needed.

2 Previous Work

SPH was developed simultaneously by Lucy [6] and Gingold and Monaghan [3] for astrophysical simulation of interacting star systems and was introduced, as a fluid simulation method, to computer graphics by Desbrun and Gascuel [2]. SPH has since been extended in many ways in order to simulate exotic types of flows and to counteract numerical instabilities. Premroze et. al. [9] introduced the Moving Particle Semi-Implicit (MPS) technique to computer graphics as a mesh free method for computing multiple interacting fluid flows. Müller et al. [7] introduced interactive real-time simulation of SPH on commodity hardware utilizing a variety of weighting functions, each for different force calculations. Plastic and elastic flows were modeled in work by Clavet et al. [1] using an impulse-based solver capable of dynamic splashing behaviors in real-time, and introduced a two-part pressure force scheme based on double-density relaxation with two polynomial weighting functions. Fast solvers based on parallel algorithms for the GPU were introduced by Takahiro et. al. [4] and brought large scale systems to commodity workstations.

Since the traditional SPH codes do not directly enforce fluid incompressibility, the fluids simulated with SPH have a tendency to compress, giving them a springy, unnatural look that is not volume preserving. This is because pressure forces are calculated as a function of local particle density, and thus act like tiny springs responding to the distance between particles. A number of techniques have been developed to reduce these compression artifacts. Solenthaler and Pajarola [10] introduced a predictive-corrective scheme for incompressible flows, iteratively arriving at a divergence-free result. Recently, Fabiano et. al. [8] used the SPH paradigm to construct a meshless form of the Helmholtz-Hodge decomposition used in the grid-based semi-Lagrangian method [11], resulting in a sparse linear system directly solvable for exactly incompressible flow.

While SPH was originally devised for astrophysical simulation, the method presented here is principally motivated by observations of the computational molecular dynamics experiments of Loup Verlet [12], where the dynamics of argon atoms were simulated through particle-particle interactions. In that paper, instead of computing neighboring particles lists each frame, they are computed once, reused many times, and then recomputed on a regular interval. The number of frames to reuse information was chosen experimentally and was selected so that momentum was approximately conserved. In this paper we expand on this idea for the particular case of incompressible fluid simulation and provide an adaptive method to choose which neighboring particle lists should be recomputed.

3 Overview of SPH

Smoothed Particle Hydrodynamics is a Lagrangian, particle-based, formulation of the Navier-Stokes equations

$$\rho(\dot{\mathbf{u}} + \mathbf{u} \cdot \nabla \mathbf{u}) = -\nabla p + \nabla \cdot (\mu \nabla \mathbf{u}) + \mathbf{F}, \tag{1}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{2}$$

where $\mathbf{u}$ is the velocity field, ρ is the fluid density, p is pressure, μ is the fluid viscosity, and $\mathbf{F}$ is the sum of external or "body" forces exerted on the fluid, accounting for effects such as gravitational acceleration. Equation 1 is the force equation, relating fluid acceleration to the various forces acting on the fluid, and Equation 2 is a statement of incompressibility; i.e. that the fluid velocity field should be divergence free. As a numerical technique, SPH approximates a field locally by a weighted sum of values from nearby particles. Any particular field quantity A, evaluated at position $\mathbf{x}$ is expressed as

$$A(\mathbf{x}) = \sum_i \frac{m_i}{\rho_i} A_i W_h(\|\mathbf{x} - \mathbf{x}_i\|), \tag{3}$$

with A_i, m_i, and ρ_i being the field value, mass, and density at particle i. W_h is typically a smooth radial basis function with finite support, often approximating a Gaussian function. These weighting functions take the distance between two points as an argument, and the subscript h is used to denote the radius of support, thus the summation need be taken over only those particles falling within radius h of position $\mathbf{x}$. Weighting functions are chosen to maximize accuracy and provide stability in numerical simulations. A discussion of weighting function properties is given by Müller et. al. [7].

The use of weighting functions simplifies the treatment of differential operators, since differentiation in the form of gradients and Laplacians is carried onto the weighting functions via linearity, so that

$$\begin{aligned}\nabla A(\mathbf{x}) &= \nabla\Big(\sum_i \frac{m_i}{\rho_i} A_i W_h(\|\mathbf{x} - \mathbf{x}_i\|)\Big) \\ &= \sum_i \frac{m_i}{\rho_i} A_i \nabla W_h(\|\mathbf{x} - \mathbf{x}_i\|).\end{aligned}$$

This formulation makes the assumption that density ρ_i is locally constant. This simplifies the computation of forces and provides negligible error if the density varies significantly only at a distance beyond the weighting function's radius. During simulation, density is needed for essentially every calculation. Using equation 3 we can formulate density at a particle i as

$$\rho_i \equiv \rho(\mathbf{x}_i) = \sum_j m_j W_h(\|\mathbf{x}_i - \mathbf{x}_j\|), \tag{4}$$

where parameters are often chosen to simplify computation, e.g. $m = 1$. Pressure is typically calculated through the equation of state and is approximated by

$$p_i = k(\rho_i - \rho_0),$$

where k is a stiffness constant related to the speed of sound in the fluid and ρ_0 is the rest density of the fluid. The pressure and viscous forces presented by Müller et. al. [7] are given by

$$\mathbf{f}_i^{pressure} = -\sum_j m_j \frac{p_i + p_j}{2\rho_j} \nabla W_h(\|\mathbf{x}_i - \mathbf{x}_j\|), \tag{5}$$

$$\mathbf{f}_i^{viscous} = -\mu_i \sum_j m_j \frac{\mathbf{v}_j + \mathbf{v}_i}{\rho_j} \nabla^2 W_h(\|\mathbf{x}_i - \mathbf{x}_j\|). \tag{6}$$

With these quantities defined, the typical SPH algorithm proceeds as in Algorithm 1.

Algorithm 1. The standard SPH algorithm

```
while simulating do
    for each particle $i$ do
        Compute Densities per Eqn. (4)
    end
    for each particle $i$ do
        Compute $\mathbf{f}_i^{pressure}$ per Eqn. (5)
        Compute $\mathbf{f}_i^{viscous}$ per Eqn. (6)
    end
    for each particle $i$ do
        $\mathbf{F}_i = \mathbf{f}_i^{pressure} + \mathbf{f}_i^{viscous}$
        Integrate($\mathbf{x}_i, \dot{\mathbf{x}}_i, \mathbf{F}_i$)
    end
end
```

4 Neighbor List Construction

While the above constitutes the bulk of a typical SPH implementation, without a mechanism to manage the time complexity of computing interparticle interactions, the simulation will scale quadratically with the number of particles. Below we explain our method for mitigating this scaling problem.

Calculating the full n^2 set of inter-particle forces is both inefficient and unnecessary. Since particle pairs that are farther than kernel radius h have weighting function values of 0, they can be discarded. Taking advantage of this requires an extra step in the SPH algorithm, where a list of interacting particle pairs is created. Particles are linked with a position in a spatial data structure such as a uniform grid [7,4]. Usually, data structures such as KD trees have been employed for these types of search problems, however, KD trees in particular have a super-linear time complexity for radius-based searches. In the computer graphics literature, there have been numerous implementations based on spatial hashing and uniform grids, which have roughly the same performance [7,4,5]. We present here the method for utilizing a uniform grid.

Letting m be the number of dimensions in the simulation, a particle's position is passed to a function, $\mathbf{I}(\mathbf{x}_i) : \mathbb{R}^m \rightarrow \mathbb{N}_0^m$, which provides the m integer

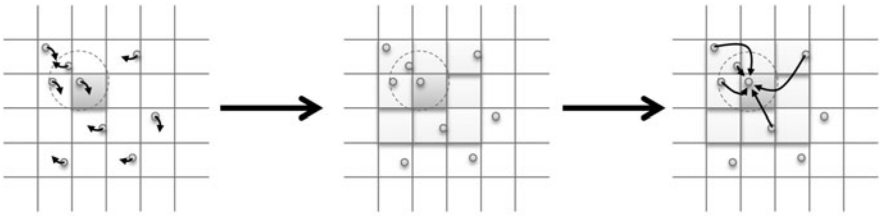

Fig. 1. Using a uniform grid to create particle-interaction lists. (left) All particles are inserted into the grid. (middle) Each particle queries neighboring grid cells and creates a list of neighboring particles. (right) This list is used throughout computation.

coordinates of the grid cell in which the particle lies. If we choose grid cell side lengths to be twice the smoothing radius h we can guarantee that, for any given position, only $2m$ grid cells need to be explored. At each grid cell, a list holds indices to particles whose centers lie inside of that grid cell. The process is shown visually in Figure 1. After this algorithm has been performed, it is no longer necessary to query all of the n^2 pairings. Instead, each particle will have a list of indices to neighboring particles.

5 Adaptive Neighbor List Algorithm

We now present our method for accelerating the algorithm given in the last section, by updating the neighbor lists only when necessary. We note that particle pairs will continue to interact so long as they lie within distance h of each other. Since the SPH algorithm gives us ready access to current particle velocity information, we can make a prediction about the positions of particle pairs in the next frame. For each particle pair, we estimate the inter-particle distance in the next frame by simple Euler integration using position and velocity from the current frame using the update equation

$$\begin{aligned} D_{ij}^{t+\triangle t} &= \|\mathbf{x}_i^{t+\triangle t} - \mathbf{x}_j^{t+\triangle t}\| \\ &\approx \|(\mathbf{x}_i^t + \triangle t\mathbf{u}_i^t) - (\mathbf{x}_j^t + \triangle t\mathbf{u}_i^t)\| \\ &= \|\mathbf{x}_i - \mathbf{x}_j + \triangle t(\mathbf{u}_i - \mathbf{u}_j)\| \end{aligned} \tag{7}$$

Thus, in our algorithm, for each current pair of particles (i, j) if the predicted inter-particle distance $D_{ij}^{t+\triangle t}$ is less than h then the pairing information is kept. If a pair violates this condition then we conclude that the two particles will be out of range of each other in the next frame and flag both particles, indicating that their neighbor lists need to be recalculated. The resulting algorithm is shown in algorithm 2.

Algorithm 2. SPH Algorithm using Adaptive Neighbor Lists

```
$\forall i \; f_i = true$
Clear Particle Neighbor Lists
while simulating do
    Clear Grid Nodes
    Insert Particles into Grid
    for each particle i do
        if $f_i$ or frame % 200 == 0 then
            Form Particle Neighbor List
            $f_i = false$
        end
    end
    for each particle i do
        Compute Density
        for each neighbor particle j in $list_i$ do
            Compute distance $(D_{ij}^{t+\triangle t})^2$ per Eqn. 7
            if $(D_{ij}^{t+\triangle t})^2 > h^2$ then
                $f_i = f_j = true$
            end
        end
    end
    for each particle i do
        for each neighbor particle j in $list_i$ do
            Compute Forces $F_{ij}$
        end
        Integrate Particle States
    end
end
```

We have found our simulations to be more reliable if we also periodically recompute every particle list using the traditional method. Our experiments have shown that recomputing all pairing information after a period of 200 frames, regardless of predictions made by this algorithm, is adequate. We also explored additional conditions for re-computation, such as recomputing if the number of neighbors was small, but found that this did not visibly enhance performance.

Since the estimated positions at the future time step are based on linear approximations to the true integrated paths, the error introduced by this method is bounded by terms on the order of $O(\Delta t^2 ||\partial_t \boldsymbol{f}||)$. For shorter time steps or forces that vary slowly with respect to successive time steps, this method can be expected to give a good estimation to the particle locations in the following time step, and thus will be stable. For this reason, we use spring penalty forces at object faces rather than simply projecting the particles to the surface, since projection would invalidate our linear estimation.

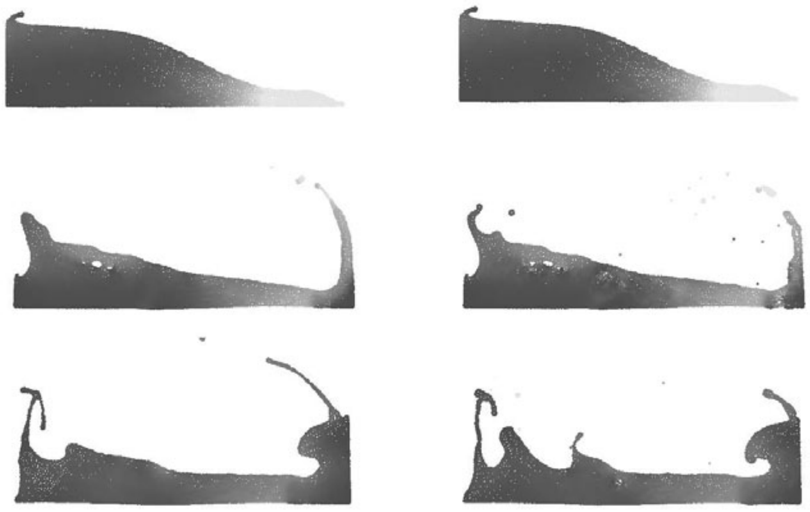

Fig. 2. Three frames taken from identical 2D simulations of a column of water under the force of gravity comparing the visual consistency of standard SPH (left) and our adaptive pairing algorithm (right). Color is used to encode kinetic energy, showing that waves travel similarly through the two simulations.

6 Results and Discussion

We conducted several simulations in two and three dimensions to compare the visual appearance and performance of fluids computed with and without the adaptive neighbor pairing algorithm. Figure 2 shows side-by-side comparisons of three frames from two 2D simulations, one using the standard SPH algorithm, and the other using our approach. Although there are small differences in the fine details, the color coding, which portrays kinetic energy, clearly shows that the overall progression of the two simulations is nearly identical.

Figure 3 gives a more fine-grained demonstration that simulations performed with and without adaptive neighbor pairing maintain excellent visual agreement. This figure shows three frames from a 3D simulation of 4,000 particles, with and without adaptive neighbor pairing. Each particle is individually represented as a ball, whose color coding represents the particle's spatial position at the start of the simulation. It can be readily seen that the positions of individual particles are quite similar, even near the end of the simulation.

Figure 4 compares timing results for a 3D simulation of water being flooded into a cubic tank, with and without our adaptive pairing algorithm. Timing results were obtained with an Intel Core i7 930 with 6 GB DDR3 1600 memory on Ubuntu 10.4 (Linux Kernel 2.6.32). Both simulations use the same uniform grid to accelerate neighbor finding. In this example, the total number of

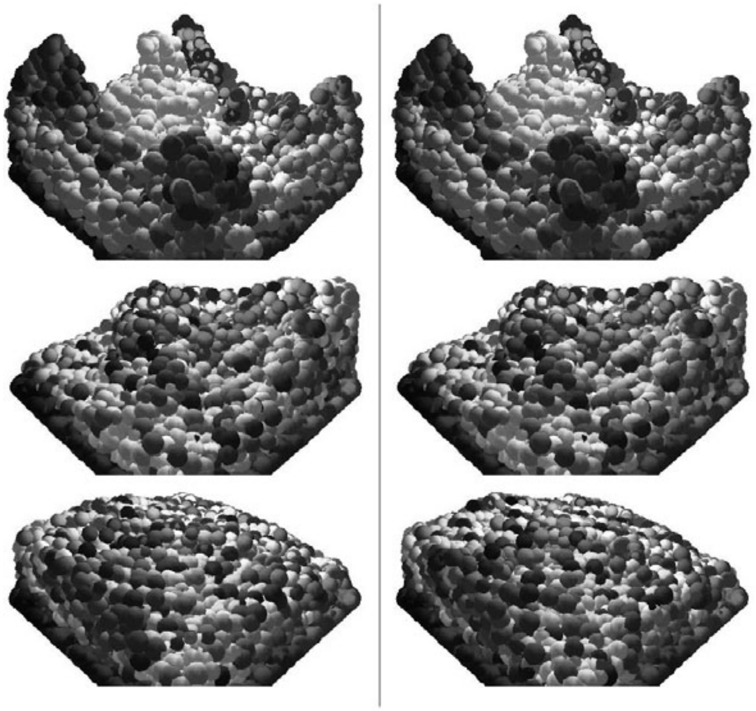

Fig. 3. Three frames from two identical simulations comparing the visual consistency between standard SPH (left), and adaptive pairing (right). Balls representing particles are color coded by their initial position so that differences are obvious. The close similarity between the two animations means that faster simulations can be produced without sacrificing control of the simulation.

particles active in the simulation increases linearly with time, such that the number of particles at the end of the simulation reaches 8,000. These results highlight the fact that for large numbers of particles, adaptive neighbor pairing becomes increasingly attractive, outperforming the standard implementation and decreasing the total time required to simulate a fluid. As can be read from the graph, with a set of only 8,000 particles the average simulation time step is computed in only a third of the time required by the standard SPH algorithm. Thus, we are confident that our method enables SPH simulations to be calculated more efficiently, without sacrificing the control of the fluid animation, at the expense of only minor surface differences.

It is important to note that while this technique is much faster than the standard SPH implementation, it does not lower the asymptotic time complexity of the algorithm since, for stability reasons, we are still required to periodically recompute neighbor lists for all particles.

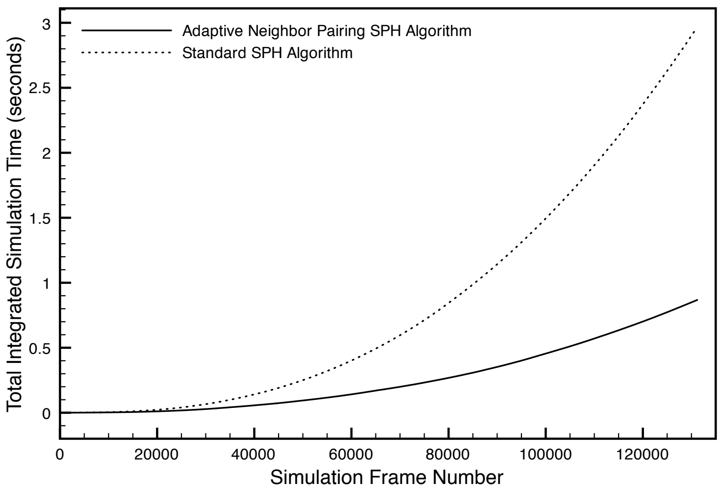

Fig. 4. Integrated simulation time for a simulation where fluid fills a cubic tank. Particles are added to the simulation every 16 frames. The total number of particles at the end of the simulation was 8,000.

We point out that this method can be additionally augmented with information from the scene such as information about collision with rigid bodies or other fluid types. For instance, since fluids interacting with fast rigid bodies or highly deformable models may present a special challenge for this algorithm, it would be possible to flag particles that interact with these objects regardless of other neighboring actions.

Our simulations were intentionally chosen to include high-pressure regions, such as where fluid collides with the floor of a tank. These types of scenarios are typically unstable because of the spring-like formulation of pressure in SPH. In our simulations, we found that the new technique does not introduce any additional instability.

Current research efforts into SPH techniques for GPU see better performance through 3D texture splatting and other techniques that circumvent the need for explicit neighbor-neighbor pairing information. As such, this technique will not be useful for GPU-based techniques.

7 Conclusion

In conclusion, we have presented a new method for speeding up the simulation of SPH fluids, based on maintenance of particle neighbor lists. The algorithm presented can be easily introduced into an existing SPH framework and provides an efficient adaptive method for lowering the computation time of particle-based fluid simulation for computer graphics and animation.

References

1. Clavet, S., Beaudoin, P., Poulin, P.: Particle-based viscoelastic fluid simulation. In: Symposium on Computer Animation 2005, pp. 219–228 (July 2005)
2. Desbrun, M., Gascuel, M.P.: Smoothed particles: A new paradigm for animating highly deformable bodies. In: Computer Animation and Simulation 1996 (Proceedings of EG Workshop on Animation and Simulation), pp. 61–76. Springer, Heidelberg (1996)
3. Gingold, R.A., Monaghan, J.J.: Smoothed particle hydrodynamics. Theory and application to non-spherical stars 181, 375–389 (1977)
4. Harada, T., Tanaka, M., Koshizuka, S., Kawaguchi, Y.: Real-time particle-based simulation on gpus. In: SIGGRAPH 2007: ACM SIGGRAPH 2007 posters, p. 52. ACM, New York (2007)
5. Krog, O., Elster, A.C.: Fast gpu-based fluid simulations using sph. In: Para 2010 - State of the Art in Scientific and Parallel Computing (2010)
6. Lucy, L.B.: A numerical approach to the testing of the fission hypothesis. Astronomical Journal 82, 1013–1024 (1977)
7. Müller, M., Charypar, D., Gross, M.: Particle-based fluid simulation for interactive applications. In: SCA 2003: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, Aire-la-Ville, Switzerland, pp. 154–159. Eurographics Association (2003)
8. Petronetto, F., Paiva, A., Lage, M., Tavares, G., Lopes, H., Lewiner, T.: Meshless helmholtz-hodge decomposition. IEEE Transactions on Visualization and Computer Graphics 99(RapidPosts), 338–349 (2009)
9. Premroze, S., Tasdizen, T., Bigler, J., Lefohn, A., Whitaker, R.T.: Particle-based simulation of fluids. Computer Graphics Forum 22(3), 401–410 (2003)
10. Solenthaler, B., Pajarola, R.: Predictive-corrective incompressible sph. ACM Transactions on Graphics, SIGGRAPH (2009)
11. Stam, J.: Stable fluids. In: SIGGRAPH 1999: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, New York, NY, USA, pp. 121–128. ACM Press/Addison-Wesley Publishing Co. (1999)
12. Verlet, L.: Computer "experiments" on classical fluids. i. thermodynamical properties of lennard-jones molecules. Phys. Rev. 159(1), 98 (1967)