Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2012, Article ID 635631, 9 pages
doi:10.1155/2012/635631

Research Article

Real-Time Simulation of Fluid Scenes by Smoothed Particle Hydrodynamics and Marching Cubes

Weihong Wang,[1,2] Zhongzhou Jiang,[1] Honglin Qiu,[1] and Wei Li[1]

[1] *College of Computer Science, Zhejiang University of Technology, Hangzhou 310023, China*
[2] *State Key Laboratory of Software Development Environment, Beijing 100083, China*

Correspondence should be addressed to Weihong Wang, wwh@zjut.edu.cn

Received 2 August 2012; Accepted 26 September 2012

Academic Editor: Fei Kang

Simulating fluid scenes in 3DGIS is of great value in both theoretical research and practical applications. To achieve this goal, we present an algorithm for simulation of fluid scenes based on smoothed particle hydrodynamics. A 3D spatial grid partition algorithm is proposed to increase the speed for searching neighboring particles. We also propose a real-time interactive algorithm about particle and surface topography. We use Marching Cubes algorithm to extract the surface of free moving fluids from particles data. Experiments show that the algorithms improve the rate of rendering frame in realtime, reduce the computing time, and extract good real effects of fluid surface.

1. Introduction

Simulation of fluid scenes has a wide range of applications in the movie special effects, computer animation, game production, virtual reality, and many other fields [1–4]. At first, most studies of fluid simulation are based on the nonphysical procedural modeling method which generates scene by parameterized surface. Although the simulation is faster, these methods are not based on physical principles, thus lack of authenticity, and can't simulate some detailed effects, such as the effect of wave overturning. At the same time these methods are based on random functions, it is difficult to achieve solid-liquid interaction. So most researchers focused on fluid simulation based on physical methods.

At present, physics-based fluid simulation methods are mainly divided into two types [5]: one is grid-based Euler method, the other is particle-based Lagrange method. Smoothed particle hydrodynamics (SPH) is a new method based on particle-based Lagrange method, which is mainly used in astrophysics at first. J. Monaghan is the first one to apply SPH into simulation of free surface flow. Stam and Fiume [6] brought in SPH method

into fluid simulation to achieve the effect of gas and flame. And then, the researches on fluid simulation mainly adopt particle-based Lagrange method. Harada et al. [7] made the searching adjacent particle feasible by constructing uniform spatial grid. Under this circumstances, fluid particles are relatively dispersed and lots of idle grid units would appear. Therefore, Grahn [8] made the fast searching adjacent particle of arbitrary size scene possible by constructing the space uniform grids with Hash function through GPU. Kolb and Cuntz [9] made use of GPU to achieve the whole process of SPH. However, this method calculates in the grids and interpolates in the particles, which leads to physical discontinuousness and simulation results distorted. David Lopez et al. [10] made use of SPH to build the physical pressure modal on river basin of the Villar Del Rey Dam, and it simulated the fluid scene of discharge floodwater. The way to deal with boundary interaction is too simple. So it simplified the terrain boundary to regular area, but the terrain surface is rugged in 3DGIS.

Although there are many SPH methods to simulate fluid in literature, they can't apply to simulate fluid in real-time and high efficiency in complex scene [11–13], such as simulating debris flow and flood and other process of fluid evolution in the 3DGIS. This paper improves 3D spatial grid partition algorithm to increase speed of neighboring particles searching, and we also propose a real-time interactive algorithm on particle and terrain surface. The results show that this method can calculate in real-time and has a good rendering rate.

2. Smoothed Particle Hydrodynamics

The basic idea of SPH makes fluid as a series of discrete particles, through the role of the smooth, which has certain radius of kernel functions to a particle physics scalar (such as density, pressure, etc.) assigning to the adjacent particles, as shown in Figure 1. Physical scalar A of any particle $X(x, y, z)$ can be calculated by

$$A(x) = \sum_j m_j \frac{A_j}{\rho_j} W(X - X_j, h), \tag{2.1}$$

$$\nabla A(x) = \sum_j m_j \frac{A_j}{\rho_j} \nabla W(X - X_j, h), \tag{2.2}$$

where ρ_j is density of the particle X_j, m_j is quality of the particle X_j, and h is radius of smoothing kernel. If the distance from the particle X to particle X_j, that is, $|X - X_j| < h$, the particles X_j is in smooth domain of the particle X, we can get the weight of particle X_j by smoothing kernel function $W(X)$. Otherwise $|X - X_j| < h$, the weight of the particles X_j is zero. In the solution of fluid equations, we often need to take derivatives of the physical scalar, and this operation only affects smoothing kernel function in SPH method. Therefore gradient of physical scalar value A can be expressed as (2.2).

3. Grid-Based Neighboring Particles Search

The existing neighboring particle search algorithms mostly adopt to traverse all particles directly, with time complexity of $O(n^2)$ and n stands for the number of fluid particles. Chen et al. [14] constructed an index table of two-dimensional array of particle and spatial grid by dividing the three-dimensional space grid with grid number being the primary key and

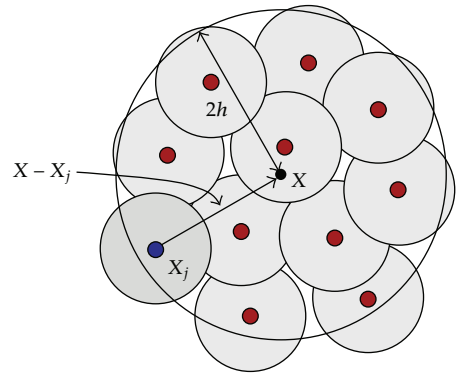

Figure 1: The basic principle of SPH.

the index table being radix sorted (algorithm complexity of radix sort is $O(n)$). The index numbers of the first particle which corresponds to each grid are obtained so that each grid in the particle is identified.

In this paper, the improved neighboring particle search algorithm doesn't need to create the index table of particle and grid but divide grids, which saves memory space. It also doesn't need to sort the particles and the index table, so it reduces the complexity of algorithm. The algorithm steps are as follows.

Step 1. Divide the three-dimensional spatial grid. In this paper, bounding box of the three-dimensional terrain space is divided into cube grid with length of h. h stands for the radius of smoothing kernel. The Standard Template Library std::vector < Particle > m_Grid is used to store information of particles of each spatial grid. Particle is a class which stores density, velocity, spatial coordinates, and other physical information of fluid particle, and a Particle* pointer pointing to next particle.

Step 2. Put the fluid particles into three-dimensional spatial grid. Insert all the particles $X_i(x, y, z)$ $(1 \le i \le n)$ into the corresponding spatial grid number (x_i, y_i, z_i) by (3.1). That is to insert the current particle pointer into the spatial grid:

$$\text{Grid}(x_i, y_i, z_i) = \{\min\_\text{int}(x/h), \min\_\text{int}(y/h), \min\_\text{int}(z/h)\}, \tag{3.1}$$

where x, y, z stands for the particle X_i in x, y, z axis coordinate, respectively, min_int is the positive integer function that get the minimum value, and grid(x_i, y_i, z_i) represents spatial grid number.

Step 3. Calculate the neighboring particles of each particle. In SPH method, particles are only affected by other particles in radius of smoothing kernel, so we will find the scope for particle located in the neighborhood of 27 spatial grids. Firstly, calculate the spatial grid number Grid(x_i, y_i, z_i) of particle X_i by (3.1); then calculate index number of Grid(x_i, y_i, z_i) by

$$\text{Gindex} = y_i * \text{GridWidth} * \text{GridHeight} + z_i * \text{GridWidth} + x_i. \tag{3.2}$$

GridWidth, GridDepth, and GridHeight, respectively, stand for the dimension on direction X, Y, Z of spatial grid, then the contained particles of $m_$Grid(Gindex) are stored in pointer array. Finally, visit all the particles in Grid $\{x_i \pm 1, y_i \pm 1, z_i \pm 1\}$ and calculate current particle of density and physical pressure and so on by using the equations derivated from (2.1) and (2.2).

4. Interaction between the Fluid Particles and the Boundary

The simplest algorithm is exhaustive algorithm on dealing with interaction between fluids and boundary. Assuming that there are M geometric facets and N fluid particles in the scene, its time complexity is $O(M^*N)$. In the 3DGIS, the number of triangular patches of terrain mostly ranges from 10^5 to 10^7 and the number of fluid particles is generally between 10^3 and 10^6. If the exhaustive algorithm is adopted, the computation will be too huge to be accepted. In this regard, [14] proposed a boundary interaction algorithm based on spatial mesh, which inserts information of the terrain boundary and obstacle into the spatial grid, then judges whether the fluid particles are intersected with boundary geometry patches and obstacle in spatial grid when the fluid particles pass through. The algorithm requires geometry patches of boundaries to be small enough; if geometry patches are large, the number of grid number each grid space occupied will increase, which will lead to reduced efficiency of algorithm.

Because the number of triangular grids of terrain is huge in the 3DGIS, if the above algorithms are adopted for boundary interaction, the amount of calculation will be too huge and the efficiency of real-time rendering will be degenerated. Regarding this, this paper proposes a real-time interaction method of the boundary bounce particles, where the time complexity of algorithms is $O(N)$ and N is the number of fluid particles. Interaction between the fluid particles and the terrain is mainly used in interaction between the fluid particles and the terrain surface to prevent fluid particles from penetrating through the terrain surface, shown in Figure 2 The basic idea of algorithm is as follows.

Step 1. Calculate intersection T_i of the ray and triangular grid of the terrain surface. The origin of the rays is $X_i\,(x_i, y_i, z_i)$ and direction is straight down $(-y)$. As DEM data is regular grid data, two-dimension array can be used for storage. Spatial coordinate Vex1 of the triangle patches of terrain surface can be calculated by (4.1). Ver2 is $(m * x\_\text{step}, m\_\text{height}[m][n+1], (n+1) * x\_\text{step})$. In a similar way, coordinates of Ver3 and Ver4 can be calculated. The x and z axis's coordinates of T_i and X_i are equal, if $(z_i - z)/(x_i - x) \geq 1$, then calculate the elevation of T_i by bilinear interpolation of triangle 1. Otherwise, calculate the elevation of T_i by bilinear interpolation of triangle 2, x_i and z_i are the coordinates of V_i, x and z are coordinates of Ver1. x, y, z, respectively, stands for the particle X_i on x, y, z axis coordinate. $x_$step and $z_$step, respectively, stands for space interval on x, z axis coordinate of the terrain elevation. $m_$height stands for the two-dimension array which stores the terrain data:

$$\begin{aligned} m &= \text{int\_min}\left(\frac{x_i}{x\_\text{step}}\right), \\ n &= \text{int\_min}\left(\frac{z_i}{z\_\text{step}}\right), \\ y &= m\_\text{height}[m][n], \\ \text{Ver1}(x, y, z) &= \{m * x\_\text{step}, y, n * z\_\text{step}\}. \end{aligned} \tag{4.1}$$

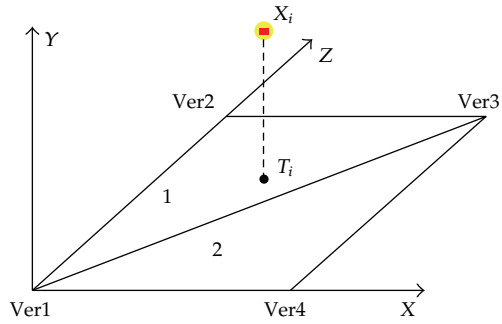

Figure 2: Interaction between the fluid particles and the terrain surface.

Step 2. Judge whether the particle X_i is in the area that rebounded by the terrain. First, calculate the spatial distance $|X_i - T_i|$ from particles X_i to T_i. If $|X_i - T_i| > R$, then the particle X_i is not in rebound area, so go to Step 1, and deal with particle X_{i+1}. Otherwise, $|X_i - T_i| \leq R$, then enter Step 3, R values for h which is the radius of the smoothed particle.

Step 3. Calculate force of rebound boundary. When fluid particle X_i is close to terrain, the particle X_i will be rebounded by the terrain surface. The rebound force f_i can be calculated by (4.2), where u_i is speed of the particle, K_{stiff} is the hardness parameters of the terrain, K_{damp} is deceleration parameter of terrain to the particle, and n is the normal unit vector of the triangular patches that include the intersection T_i. When the fluid particles gradually approach the terrain surface, the rebound force increases gradually until the velocity of the particle decreases to zero. At this time, the particle is still subjected to the effect of rebound which increases the velocity of the particle, thus preventing fluid particles to penetrate through the terrain. The direction of the rebound force is normal direction of the triangular mesh plane. Note that the rendering system only renders the clockwise triangular patches; we adopt clockwise triangles for the calculation of the normal unit vector of triangular patches that the particles locate in:

$$f_i = \begin{cases} K_{\text{stiff}} * (2R - d) - u_i * n * K_{\text{damp}} * n, & R > d,\ d = |X_i - T_i|, \\ 0, & \text{otherwise.} \end{cases} \tag{4.2}$$

5. Visualization of the Fluid Surface

The fluid surface reconstruction is important for fluid simulation. Lighting and texture rendering of the fluid surface will improve the fidelity of fluid scene. Iwasaki et al. [15] proposed Point Splatting method to build the surface rendering, and this method has highly real-time rendering efficiency, but emptiness appears where fluid particles are relatively sparse [16–20]. In this paper, Marching Cubes algorithm is proposed to be applied in constructing fluid surface of free moving, which has the advantages of simple operation and fast drawing. The process is as follows.

Step 1. The 3D space is evenly divided into spatial grids. Traverse all the particles in the fluid region, calculate minimum coordinates (Ver_{min}) and maximum coordinates (Ver_{max}) of

the particles which are bounding box of fluid region, and divide the box into homogeneous spatial grids.

Step 2. Calculate the density of the vertices of each spatial grids. Then use the divided grids as the sampling points of fluid scenes to get fluid density field data. Calculate the density value of each grid of the 8 vertices by using (2.1) and SPH interpolation on particles of each grid.

Step 3. Determine the threshold value of density isosurface of the fluid surface. According to the fluid pressure equal to atmospheric pressure, and the state equation of an ideal atmosphere, threshold value of density isosurface of the fluid surface can be obtained as follows:

$$\rho_{\text{surface}} = \frac{p}{k}, \tag{5.1}$$

where ρ_{surface} stands for the fluid surface density, p for atmospheric pressure, and k for gaseous constant.

Step 4. Compare each vertex density value of spatial grid with threshold value of isosurface. If the vertex density value is greater or equal to the isosurface value, vertex value is 1 and the vertex is in the isosurface; otherwise is 0 and the vertex is outside of isosurface. According to the result of the comparison, structure the grid state table.

Step 5. Calculate the vertex of density isosurface coordinates of each spatial grid. If a vertex of a side of a grid is in the isosurface, while the other vertex is out of the isosurface, then, this side inevitably intersects with the desired isosurface. Firstly, according to the grid state table, get the grid's sides which intersect with the isosurface. Then calculate the intersection of grid's side and isosurface by linear interpolation method.

Step 6. Draw density isosurface. By using the central difference method and the linear interpolation method, calculate vertex normal of each triangle. Finally according to the coordinate values of each triangle vertex and normal vector, draw density isosurface.

6. Experiment Results

In this paper, the experiment platform is Intel Core 2 Duo CPU T6670@2.2GHZ. Main memory is 2 G, and graphic card is NVIDIA GeForce 9300GS with memory of 256 M, and operating system is Windows XP, and the drawing SDK is OSG (OpenSceneGraph).

Figure 3 shows fluid outflowing from a topographic position in 3DGIS, and its dynamic process of evolution on the terrain surface. The fluid particle number is 30000. Figure 4 shows the Figure 3 Scene by Marching Cube algorithm to construct the effect of fluid surface of free moving.

As this algorithm applies Marching Cube algorithm which is the same as [8] to construct the fluid surface and field particle search and complex boundary interaction algorithm which are used in this algorithm are different to [12], thus they are comparable. Table 1 shows the comparison of simulation speed of the algorithm and two other kinds of algorithm. Figure 5 shows comparison of computing time per frame. Table 1 and Figure 5 show that this algorithm reduces the computation time per frame.

Figure 3: Fluid in the dynamic evolution process of particle simulation (particle number 30000).

Figure 4: Effect of reconstruction of fluid surface mesh reconstruction (particle number 30000).

Table 1: Comparison of the three algorithms on simulation rate.

This paper algorithm		[12] Algorithm		[8] Algorithm	
Particle number	FPS	Particle number	FPS	Particle number	FPS
2000	153.4	3000	68	4000	33
4000	94.3				
8000	45.2	8000	26		
16000	26.4	20000	12.7	20000	10
32000	14.5				
64000	8.2			30000	6.7
128000	3.8				

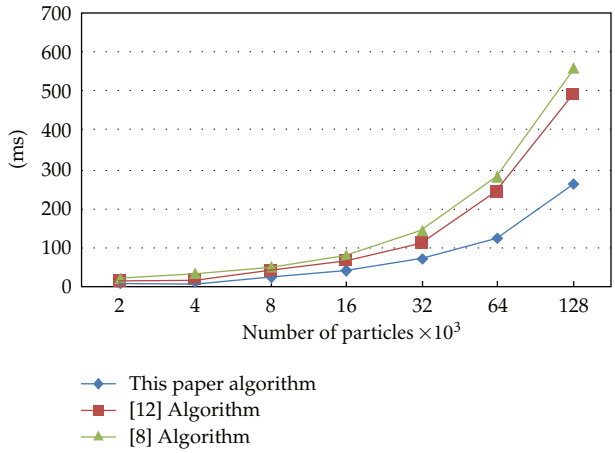

Figure 5: Comparison of the three algorithms on simulation rate.

7. Conclusion

In order to simulate the fluid scene in 3DGIS, this paper proposed an SPH algorithm for fluid scene simulation. The experiment results show that this algorithm can calculate in real-time and has a good real-time rate of rendering frame, and achieve the desired goals. In near future we intend to further improve the algorithms of extracting fluid surface and enhance reconstruction speed and accuracy of the fluid surface mesh.

Acknowledgments

This paper is supported by the National Natural Science Foundation of China (60873033), the Natural Science Foundation of Zhejiang Province (R1090569), and the State Key Laboratory of Software Development Environment Open Fund (SKLSDE-2012KF-05).

References

[1] M. Carlini and S. Castellucci, "Modelling and simulation for energy production parametric dependence in greenhouses," *Mathematical Problems in Engineering*, vol. 2010, Article ID 590943, 28 pages, 2010.

[2] M. Li, C. Cattani, and S. Y. Chen, "Viewing sea level by a one-dimensional random function with long memory," *Mathematical Problems in Engineering*, vol. 2011, Article ID 654284, 13 pages, 2011.

[3] Carlo Cattani, Shengyong Chen, and Gani Aldashev, "Information and Modeling in Complexity," *Mathematical Problems in Engineering*, vol. 2012, Article ID 868413, 4 pages, 2012.

[4] S. Chen, H. Tong, and C. Cattani, "Markov models for image labeling," *Mathematical Problems in Engineering*, vol. 2012, Article ID 814356, 18 pages, 2012.

[5] Y. Liu, X. Liu, H. Zhu, and E. Wu, "Physically based fluid simulation in computer animation," *Journal of Computer-Aided Design and Computer Graphics*, vol. 17, no. 12, pp. 2581–2589, 2005.

[6] J. Stam and E. Fiume, "Depicting fire and other gaseous phenomena using diffusion processes," in *Proceedings of the 22nd Annual ACM Conference on Computer Graphics and Interactive Techniques*, pp. 129–135, Los Angeles, Calif, USA, August 1995.

[7] T. Harada, S. Koshizuka, and Y. Kawaguchi, "Smoothed particle hydrodynamics on GPU," in *Proceedings of The Computer Graphics International*, pp. 63–70, Rio de Janeiro, Brazil, 2007.

[8] A. Grahn, "Interactive simulation of contrast fluid using smoothed particle," *Hydrodynamics*, pp. 1–69, 2008.

[9] A. Kolb and N. Cuntz, "Dynamic particle coupling for GPU based fluid simulation," in *Proceedings of the 18th Symposium on Simulation Technique*, pp. 722–727, Erlangen, Germany, 2005.

[10] D. Lopez, R. Marivela, and F. Aranda, "SPH model calibration using data from pressure of the prototype still basin of Villar Del Rey Dam, Spain," in *Proceedings of the 33rd IAHR Congress, Water Engineering for a Sustainable Environment*, pp. 187–198, 2009.

[11] S. Y. Chen, "Kalman filter for robot vision: a survey," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 11, pp. 4409–4420, 2012.

[12] P. Kipfer and R. Westermann, "Realistic and interactive simulation of rivers," in *Proceedings of the 32nd Annual Conference on Computer Graphics (SIGGRAPH '06)*, pp. 1–8, New York, NY, USA, June 2006.

[13] S. Chen, Y. Wang, and C. Cattani, "Key issues in modeling of complex 3D structures from video sequences," *Mathematical Problems in Engineering*, vol. 2012, Article ID 856523, 17 pages, 2012.

[14] X. Chen, Z. Wang, J. He, H. Yan, and Q. Peng, "An integrated algorithm of real-time fluid simulation on GPU," *Journal of Computer-Aided Design and Computer Graphics*, vol. 22, no. 3, pp. 396–405, 2010.

[15] K. Iwasaki, Y. Dobashi, F. Yoshimoto, and T. Nishita, "Real-time rendering of point based water surfaces," *Advances in Computer Graphics*, vol. 4035, pp. 102–114, 2006.

[16] C. Cattani, "Shannon wavelets for the solution of integro-differential equations," *Mathematical Problems in Engineering*, vol. 2010, Article ID 408418, 22 pages, 2010.

[17] E. G. Bakhoum and C. Toma, "Specific mathematical aspects of dynamics generated by coherence functions," *Mathematical Problems in Engineering*, vol. 2011, Article ID 436198, 10 pages, 2011.

[18] S. Y. Chen and Z. J. Wang, "Acceleration strategies in generalized belief propagation," *IEEE Transactions on Industrial Informatics*, vol. 8, no. 1, pp. 41–48, 2012.

[19] N. M. Kwok, X. Jia et al., "Visual impact enhancement via image histogram smoothing and continuous intensity relocation," *Computers & Electrical Engineering*, vol. 37, no. 5, pp. 681–694, 2011.

[20] S. C. Lim, C. H. Eab, K. H. Mak, M. Li, and S. Y. Chen, "Solving linear coupled fractional differential equations by direct operational method and some applications," *Mathematical Problems in Engineering*, vol. 2012, Article ID 653939, 28 pages, 2012.