GPU Accelerated Surface Reconstruction for Particle-based Fluids

Wei Wu
College of Information Science and Engineering, Ocean University of China, Qingdao 266100,China
oucws2011@163.com

Hongping Li*
College of Information Science and Engineering, Ocean University of China, Qingdao 266100, China
lhp@ouc.edu.cn

Tianyun Su
The First Institute of Oceanography, SOA, Qingdao 266061，China
sutiany@fio.org.cn

Haixing Liu
The First Institute of Oceanography, SOA, Qingdao 266061，China
liuhx@gmail.com

ABSTRACT
Animating fluids with particle-based method has attracted great attention in computer graphics. To render discrete particles, implicit surfaces need to be extracted. But this process is often seen as a bottleneck due to the expensive computation and high memory usage especially for constructing a detailed and artifact free surface. In this paper, a GPU accelerated fluids surface reconstruction method using 2-level grid structure is employed with a scheme of arranging fine surface vertices, which could preserve the spatial locality to facilitate the coalesced memory access on GPU. Meanwhile a parallel cuckoo hashing method is taken to help reduce the memory consumption. A parallel version of the optimized surface reconstruction was performed based on CUDA architecture. In the experiment of comparison to traditional approaches, the results indicated that our surface reconstruction method was more efficient at the same level of quality of the reconstructed surfaces.

Keywords
Smoothed Particle Hydrodynamics; Fluids Simulation; Surface Reconstruction.

1. INTRODUCTION
Smoothed particle hydrodynamics (SPH) has attracted great attention for fluids animation in computer graphics. It works by dividing the fluid into a set of discrete particles. The physical quantity of any particle can be obtained by summing the relevant properties of all of the neighbor particles within the influence radius. However it doesn't look very realistic when the particles are simply rendered individually. To render these discrete particles into a body of fluid, one good way is to reconstruct the fluid surface using Marching Cubes (MC) algorithm [1] to generate the surface triangle meshes. However reconstructing high-quality surface with high computational efficiency and optimized memory consumption is still challengeable.

There exist various techniques to improve the performance of surface reconstruction in recent years, among which, a parallel narrow band method is presented by Akinci et al. [2]. While many approaches process all the grid nodes, they improve the efficiency by only considering grid nodes in a narrow-band region around the surface. Also their algorithm is designed for parallel architectures either on CPU or GPU, which further improve the performance. In this paper, our method is a novel extension to the method of Akinci et al. [2], therefore their steps are briefly outlined here. According to their method, the whole simulation domain is divided into a single level uniform grid with a cell size of r/2 (r is the equilibrium distance between particles). Any grid vertex that located in the proximity of a surface particle is defined as surface grid vertex which needs to be identified. This can be done by defining an AABB which spans 4r length in each direction around each surface particle. Any grid vertex within this AABB are marked and collected into a new array. Then the scalar values computation is only performed on these determined surface grid vertices. Once complete, a grid cell is created for each surface grid vertex. Then the cell is polygonized using MC method with the scalar values of its eight points as parameters. The process of polygonization in each cell can be done simultaneously thus easily adaptable for parallelization. Finally a continuous triangle mesh is extracted to represent the fluid surface.

In this paper, we present a more efficient GPU-accelerated surface reconstruction method for particle-based fluid. The inspiration is mainly originated from the following two observations to the method of Akinci et al. [2].

(1) Scalar values computation is the most exhausting process, whose efficiency is largely influenced by the arrangement mode of surface grid vertices that collected in array.

(2) To polygonize a cell, the array positions of its eight corners need to be identified easily so as to lookup the corresponding scalar values. This is done with the help of a whole grid to indicate where each surface grid vertex locates in the collected array. However, this structure causes unnecessary memory allocation for the unused cells.

2. ALGORITHM
In this section, details of our optimized surface reconstruction algorithm are described. In a preprocessing step, the surface particles are determined by employing the smoothed color field method [3]. Then the optimized surface reconstruction method can be divided into three parts. Firstly the surface vertices are rearranged with the designed 2-level grid structure which improves the spatial locality to facilitate the coalesced memory access on GPU. A parallel cuckoo hashing method is also used to replace the uniform grid, which helps reduce the memory consumption. Secondly, scalar values computation is performed on the determined surface vertices. Finally each surface cell is polygonized with MC method.

2.1 Extracting Surface Vertices
We take a 2-level grid structure to extract surface vertices. In contrast to dividing the uniform grid with a cell size of r/2 directly, we initialize the grid with a cell size of 2r as the coarse level. For each surface particle, the neighboring contiguous 3^3

coarse cells are defined as coarse surface cells. Then each coarse surface cell is further divided into finer cells with a cell size of r/2 as the second level. Obviously each coarse cell contains 64 fine grid vertices. A 2-dimentional subdivision illustration is shown in Fig. 1.

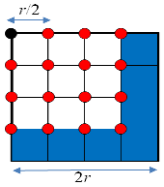

Figure 1: A 2-dimentional subdivision illustration. The thick black line represents the outline of the coarse surface cell whose initial grid vertex is colored in black. Both black and red colored grid vertices are the fine grid vertices in the second level.

In specific implementation, a parallel process starts with each thread corresponding to a coarse surface cell and writing the index values of all 64 fine grid vertices within this coarse surface cell into a continuous memory. In this way, only recording the array position of each coarse surface vertex is enough to help determine the array position of each fine surface vertex. This is done with the help of a hash table. Key-value pairs are built with the index of each coarse surface vertex as key and the corresponding array position as values. Then the parallel cuckoo hashing method is used to determine where a key-value pair is pushed into the hash table.

2.2 Scalar Values Computation

For each determined fine surface vertex, we collect all its neighboring particles within the influence of 4r and take the improved signed distance field approach [4] to compute the scalar value. This process will achieve an efficiency improvement due to the arrangement mode of surface vertices that already built in Section 2.1. This can be explained as the required data for the continuous threads are more likely located within less segments in global memory, which results less data transfer times thus facilitates the coalesced memory access on GPU.

2.3 Triangulation

After completing the scalar values computation, we create a cell for each fine grid vertex. All the scalar values of its eight corners need to be queried from the array for triangulation. If all the corners are located within the same coarse cell, their scalar values are determined based on the local relationship. Or else for the corners that belong to different coarse cells (the blue shading in Fig. 1), their scalar values are determined with the help of the hash table built in Section 2.1. Once complete, the MC look-up tables are queried and the MC method is applied for each fine cell.

Table 1: Performance comparison between different surface reconstruction methods in the Buddha scene.

method	t_{es}[sec]	t_{sf}[sec]	t_{tri}[sec]	t_{tol}[sec]	MEM[GB]
NB	0.33	5.98	0.09	6.4	3.16
Ours	0.05	1.85	0.14	2.0	0.53

3. EXPERIMENTS

To demonstrate the utility of our approach, we build a Buddha scene where a viscous fluid is poured onto a Buddha statue and collected into a boundary box. The falling fluids strike the top of the statue and gradually form a splash towards its side. The narrow band method (NB) and our method were taken to reconstruct surface for a performance comparison.

Figure 2. The Buddha scene with 0.2M particles

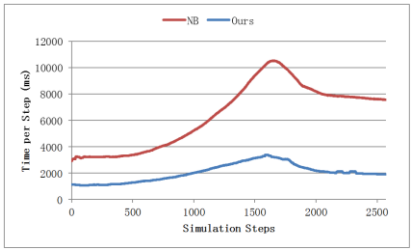

Figure 3. Surface reconstruction time over simulation steps

4. RESULTS AND DISCUSSION

The running time per frame over simulation steps with the NG method and our method is shown in Fig. 3. The results of average running time and average memory consumption per frame are presented in Table 1. t_{es} denotes the time of extracting surface vertices, t_{sf} denotes the time of scalar field computation, t_{tri} denotes the triangulation time, t_{tol} denotes the total running time and MEM represents the memory usage on the graphics card. As shown in Table 1, the average surface reconstruction time with our method is 2044ms. When compared with the NB method, our method achieves a speed-up of 3 times in the running time and 6 times in the memory consumption. The mainly improvement in the running time is at the stage of scalar field computation, which is facilitated with a scheme of rearranging fine surface vertices. The speedup in the storage is due to the applied spatial hashing method, which make the memory consumption truly scale with the fluid surface instead of the whole simulation domain.

5 REFERENCES

[1] Lorensen W E, Cline H E. Marching cubes: A high resolution 3D surface construction algorithm[C]//ACM siggraph computer graphics. ACM, 1987, 21(4): 163-169.

[2] Akinci G, Ihmsen M, Akinci N, et al. Parallel Surface Reconstruction for Particle-Based Fluids[C]//Computer Graphics Forum. Blackwell Publishing Ltd, 2012, 31(6): 1797-1809.

[3] Müller M, Charypar D, Gross M. Particle-based fluid simulation for interactive applications[C]//Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation. Eurographics Association, 2003: 154-159.

[4] Solenthaler B, Schläfli J, Pajarola R. A unified particle model for fluid–solid interactions[J]. Computer Animation and Virtual Worlds, 2007, 18(1): 69-82.