LETTER

An Adaptive Model for Particle Fluid Surface Reconstruction

Fengquan ZHANG[†a)], ***Student Member***, **Xukun SHEN**[†], ***and*** **Xiang LONG**[†], ***Nonmembers***

SUMMARY In this letter, we present an efficient method for high quality surface reconstruction from simulation data of smoothed particles hydrodynamics (SPH). For computational efficiency, instead of computing scalar field in overall particle sets, we only construct scalar field around fluid surfaces. Furthermore, an adaptive scalar field model is proposed, which adaptively adjusts the smoothing length of ellipsoidal kernel by a constraint-correction rule. Then the isosurfaces are extracted from the scalar field data. The proposed method can not only effectively preserve fluid details, such as splashes, droplets and surface wave phenomena, but also save computational costs. The experimental results show that our method can reconstruct the realistic fluid surfaces with different particle sets.

key words: *particle-based fluid, surface reconstruction, adaptive model, scalar field*

1. Introduction

Particle-based method has recently attracted the attention of researchers in fluid simulation. However, the reconstruction of smooth and detailed surfaces for small-scale particle sets is usually a bottleneck due to irregularly placed particles and low computational efficiency. Recently, the high quality visualization of particle fluid surface is discussed in many literatures. Commonly employed methods include metaballs, point splatting, raycasting and marching cubes (MC) [1]–[4]. Premoze et al. [5] introduced a notation of level set method to reconstruct surface from particles, however, breaking waves and droplets cannot be seen. Zhu and Bridson [6] improved the method of [5], they computed the scalar value with an implicit function, which generated not smooth enough surfaces, especially in concave regions. Zhang et al. [7] presented a fluid rendering technology using a metaball method. Unfortunately, the radius of metaball need be chosen carefully, otherwise some bumpy artifacts would appear. Goswami et al. [8] directly used raycsting to render the distance field data on GPU, which allowed the visualization at high frame rates. Without extracting and tracking polygonal meshes, the screen-space method had a good efficiency as shown in [9]. However, the bumpiness would appear in the near-distance perspective.

Our method is inspired by the work of [10], which replaced the spherical kernels using ellipsoidal kernels, and had better accuracy than standard SPH. However, some defects still exist. First, an averaging of particle centers is enforced, resulting in slight volume shrinkage. Then, although it yielded very smooth surfaces, the smoothing step still made ripples less visible in fluid surfaces. Moreover, the computation of scalar field must be implemented on the whole particle sets, increasing computational overhead notably.

Based on above considerations, we propose a new surface reconstruction method that can generate smooth surfaces with fine details, such as splashes, droplets and ripples. In order to reduce the computational overhead of scalar field, our method only generates scalar field around fluid surfaces. The key of our method is applying an adaptive scalar model to reconstruct surfaces, which adaptively adjusts smoothing length based on the neighborhood distribution of particles. For computational accuracy, a constraint-correction rule is used for the smoothing length computation. Finally, the surface is extracted from the isosurface obtained from the scalar field. We also show that our method creates the realistic visualization of fluid surfaces.

2. Surface Reconstruction from Particles

In this section, we will focus on the adaptive model from three aspects, including scalar field definition, determining smoothing length and extracting surface particles.

2.1 Scalar Field Definition

The surface of a fluid is defined as a scalar field $\phi(\boldsymbol{x})$

$$\phi(\boldsymbol{x}) = \sum_j \frac{m_j}{\rho_j} W(\boldsymbol{x} - \boldsymbol{x_j}, h^*_{1,2,3}), \tag{1}$$

where $W(\boldsymbol{x} - \boldsymbol{x_j}, h^*_{1,2,3})$ is the weighted kernel with variable smoothing length $h^*_{1,2,3}$, $h^*_{1,2,3} = (h^i_1, h^i_2, h^i_3)$, m_j is the mass, and ρ_j is the density. The standard SPH kernel commonly uses a constant smoothing length, which can only reflect the distance variation in time and space, but not the directional features. If the traditional method is applied to simulate the anisotropic deformable scene, a lot of details will be lost. Moreover, the spherical smoothing function cannot reflect density distribution in near surfaces. Thus, we adopt an adaptive model that employs a variable smoothing length along each axis of the ellipsoid, which is used to follow the changes of local particle distributions. We select the cubic spline based one-dimensional smoothing function proposed

Manuscript received September 18, 2012.
Manuscript revised December 5, 2012.

[†]The authors are with the State Key Laboratory of Virtual Reality Technology and System, Beihang University, Beijing, China.
a) E-mail: zhangfengquan112@163.com

DOI: 10.1587/transinf.E96.D.1247

by Monaghan [11], and take the form:

$$W(\boldsymbol{k},h)=\alpha_d\times\begin{cases}\frac{2}{3}-k^2+\frac{1}{2}k^3 & 0\le k<1,\\ \frac{1}{6}(2-k)^3 & 1\le k<2,\\ 0 & k\ge 2,\end{cases}\tag{2}$$

where $\alpha_d=\frac{1}{h}$ in one-dimensional and $k=\frac{r}{h}$, k is the distance between two particles normalized by the scalar smoothing length h, and r is the real distance between two particles. Our 3D cubic spline kernel is given by

$$W(\boldsymbol{k},h^i_{1,2,3})=W_1(\boldsymbol{k_1},h^i_1)W_1(\boldsymbol{k_2},h^i_2)W_1(\boldsymbol{k_3},h^i_3),\tag{3}$$

where W_1 is the 1D cubic spline kernel. $\boldsymbol{k_1}$, $\boldsymbol{k_2}$ and $\boldsymbol{k_3}$ are the projections of $(\boldsymbol{x-x_j})/h$ along each of these axes, respectively, h^i_1, h^i_2 and h^i_3 are the three components of semimajor axes of the ellipsoid.

2.2 Determining Smoothing Length

To better obtain the characteristics of particle distributions, each particle has to carry its own smoothing length. The $h^*_{1,2,3}$ is computed by analyzing the main characteristic direction of neighboring particles based on the Principal Component Analysis method.

Concretely speaking, a particle $p\in F$, F is the set of particles with k feature points in the R-neighborhood. $N(p)$ denotes the R-neighborhood particles of p, defined by $N(p)=\{p_i\in F,\ |p_i-p|\le R\}$, $i=1,\ldots,k$. We now analyze the $N(p)$ by PCA. We construct the covariance matrix $C_i[3,3]$ as

$$C_i=(p_1-p,\ldots,p_k-p)\cdot(p_1-p,\ldots,p_k-p)^T,\tag{4}$$

Then we compute the eigenvalues of equation $C_ix_i=\lambda_ix_i$, $i\in 0,1,2$, and let $\lambda_0\ge\lambda_1\ge\lambda_2$. The eigenvector x_0 corresponds to the largest eigenvalue λ_0 that is expected to be aligned with the first main direction in the neighborhood $N(p_0)$. It is the principal axis direction of ellipsoid, and its length is $\sqrt{|\lambda_0|}$, the other two lengths are $\sqrt{|\lambda_1|}$ and $\sqrt{|\lambda_2|}$. So we can initialize the smoothing length of particle p for $h^0_{1,2,3}=(\sqrt{|\lambda_0|},\sqrt{|\lambda_1|},\sqrt{|\lambda_2|})$, which is the three semimajor axes of the ellipsoid. Furthermore, the number of particles in neighborhood should be roughly constant. In order to maintain the amount of neighborhood particles in the scope of smoothing length, we introduce a constraint-correction approach to compute the smoothing length. The method includes the following two steps.

[Step 1: Constraint] We first divide the whole simulation field into regular grid cells, where the grid spacing is shorter than the smoothing length. The smoothing length is controlled by N_{mean}, which is the amount of expected neighborhood, $N_{mean}=n\frac{N_{total}}{N_{grid}}$, where N_{total} is the total particles, N_{grid} is the number of grids containing particles and n is the scale factor decided by users. Each particle has an initial smoothing length $h^0_{1,2,3}=(\sqrt{|\lambda_0|},\sqrt{|\lambda_1|},\sqrt{|\lambda_2|})=(h^0_1,h^0_2,h^0_3)$. Then we compute the number of particles $N^0_{i,j,k}$ near the $Cell_{i,j,k}$ in the ellipsoidal scope of h^0_1, h^0_2 and h^0_3. In general, $h^0_{1,2,3}$ is the approximate length, large (small) for low-density (high-density) regions. Using the length of $h^0_{1,2,3}$ may result in excessively smoothing surface and losing the details of droplets or splashes.

[Step 2: Correction] It is necessary for us to adjust smoothing length to meet the various regional characteristics. After counting the number of particles of $Cell_{i,j,k}$, we adjust the new smoothing length to $h^1_{1,2,3}=\frac{N_{mean}}{N^0_{i,j,k}}(h^0_1,h^0_2,h^0_3)$ according to the average density. We have to examine whether $h^1_{1,2,3}$ satisfies the requirements, since details are closely associated with the number of neighborhood particles. We compute the $N^1_{i,j,k}$ in each cell using $h^1_{1,2,3}$. If the smoothing length has been overestimated, that is, $N^1_{i,j,k}>N_{mean}$, we adjust it into

$$h^*_{1,2,3}=\sigma_i\frac{N_{mean}}{N^1_{1,2,3}}h^1_{1,2,3},\tag{5}$$

where $\sigma_i=1-\frac{|x_i-x_j|/R}{\sum_j|x_i-x_j|/R}$ is a relaxation coefficient that guarantees the numerical stability, R is the length of the neighborhood around $cell_{i,j,k}$. In such situation, the new smoothing length is derived as $h^*_{1,2,3}$ which represents the length of the three semimajor axes of ellipsoid respectively. It reflects the characteristics of local particle distributions. It can not only eliminate redundant neighboring particles and reduce the costs, but also preserve the neighborhood characteristics. If $N^1_{i,j,k}<N_{mean}$, then $h^1_{1,2,3}$ needs be increased appropriately. This case often occurs in the edges or droplet regions. The correction can be executed by the equation as

$$\frac{N^0_{i,j,k}-N^1_{i,j,k}}{h^0_{1,2,3}-h^1_{1,2,3}}=\frac{N_{mean}-N^1_{i,j,k}}{h^*_{1,2,3}-h^1_{1,2,3}},\tag{6}$$

thus $h^*_{1,2,3}=\frac{(N_{mean}-N^1_{i,j,k})(h^0_{1,2,3}-h^1_{1,2,3})}{N^0_{i,j,k}-N^1_{i,j,k}}+h^1_{1,2,3}$, the new estimate is applied for the three axises of ellipsoid. After steps 1 and 2, the adaptive smoothing length $h^*_{1,2,3}$ can be obtained. Finally, the scalar value is computed using Eq. (1).

2.3 Extracting Surface Particles

The computational costs of surface reconstruction are influenced by the number of particles and the resolution of MC

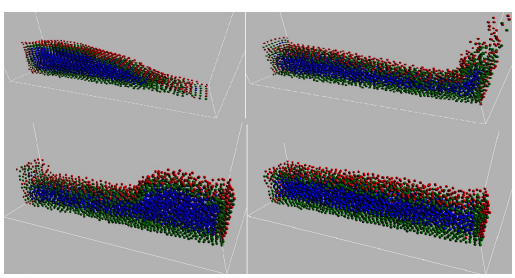

Fig. 1 Surface particle extraction of a dam breaking scene. The outermost red particles are surface particles. For accuracy, we also consider the secondary outer layer green particles as the surface particles. The blue particles are the inner particles that need not to compute in scalar field.

grids. For computational efficiency, we only compute the scalar field around fluid surfaces. A particle p_i is treated as the surface particle when the value of D_i exceeds a certain threshold, as shown in Fig. 1. D_i is defined as

$$D_i = \sigma_1 \cdot |x_i - \overline{x_i}| + \sigma_2 \cdot \frac{F_i}{F_{mean}} + \sigma_3 \cdot \frac{N_i}{N_{mean}} \tag{7}$$

where $\overline{x_i}$ is the center of mass of p_i, F_{mean} is the mean surface tension obtained from the physical simulation step. N_i denotes the number of neighboring particles of p_i, and N_{mean} is the mean value of N_i in each time step. σ_i ($i = 1, 2, 3$) is the weighted parameter used to adjust the scope of surface particles for scalar field computation.

3. Results

In order to show the quality of our method, we simulate several different animations: Cube, Bunny, Pool and Stair. The implementations of simulation and surface reconstruction are executed on a PC equipped with an Intel Core 2 Quad Q9300 @2.5 GHz CPU and a Nvidia GeForce GTX480 GPU with 1.5 G VRAM. Figure 2 shows the bunny falls on a stair scene with 300 K particles, which is simulated using PCISPH solver. The water bunny falls on the stair, then splashes and droplets are emerging. Furthermore, we notice that the smooth surfaces and clear edges of bunny in thin particle regions are also well created. The characteristics of neighborhood particles are well protected by automatically adjusting the smooth length, thus the details can be well-preserved.

Figure 3 is an animation of a cube of water being dropped into a pool with 220 K particles, which is simulated using WCSPH solver. Our adaptive surface reconstruction method creates a smooth surface and preserves sharp features. Moreover, the surfaces still keep smooth as the fluid tends to calm.

Figure 4 shows the water splashes into a pool scene with 400 K particles. It shows a comparison of the method of [10] and our adaptive method. The left is rendered with our method, such as (a) and (c). The right is rendered with the method of [10], such as (b) and (d). (a) and (b) show the effect of fluctuations at the same frame. Because the method of [10] uses a processing of smoothing step, we can see that the effects of ripples disappear and some details in thin particles are also lost. On contrast, our method protects these

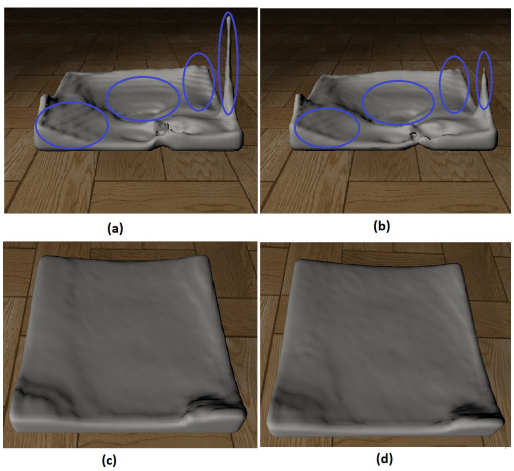

Fig. 4 Comparison of surfaces between our method (left) and [10] (right). The corresponding contrasts are labeled with blue circles. Top row: the effect of fluctuations; Bottom row: the effect of almost calm.

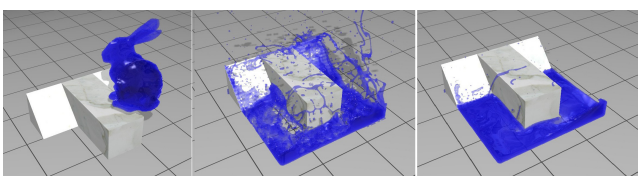

Fig. 2 Surface reconstruction of the bunny falling on a stair scene with 300 k particles. Note that the details, such as edges, corners and splashes of the fluid are well-preserved.

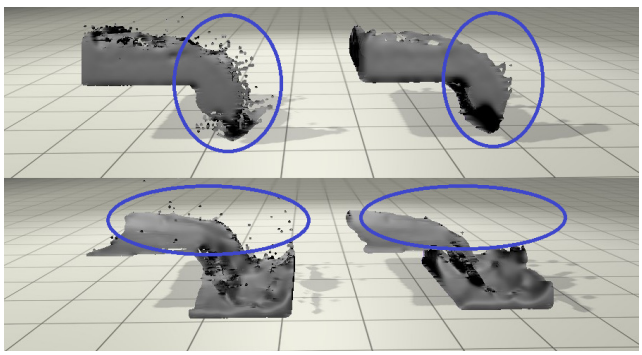

Fig. 5 Comparison of details between our method (left) and [10] (right) and. The corresponding contrasts are labeled by blue circles.

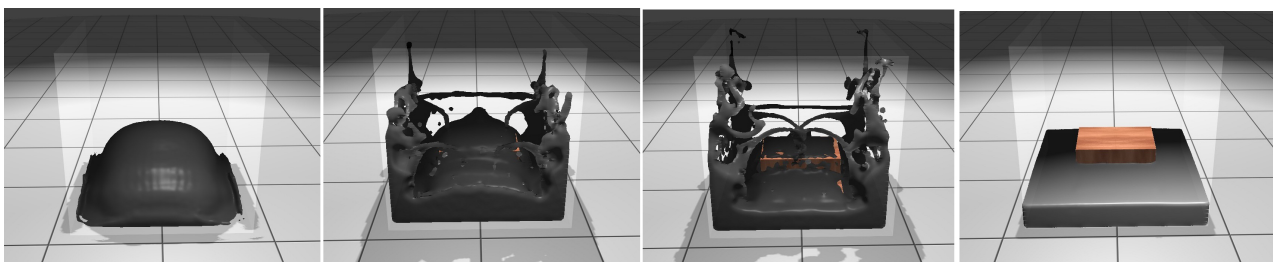

Fig. 3 Surface reconstruction of a cube of water being dropped into a pool with 220 K particles. Note that the smoothness of fluid surfaces and sharp features are well-preserved.

Table 1 Performance of the presented scenes for scalar field computation.

Scene	Solver	Particle numbers	$\frac{Particle_{surf.}}{Particle_{total}}$	Grid resolution	Our method [sec]	The method of [10] [sec]
Cube (Fig. 2)	WCSPH	220 k	0.09	$253 \times 247 \times 283$	0.24	1.62
Bunny (Fig. 3)	PCISPH	300 k	0.15	$312 \times 298 \times 331$	0.32	2.63
Pool (Fig. 4)	WCSPH	400 K	0.13	$330 \times 311 \times 347$	0.57	4.35
Stair (Fig. 5)	PCISPH	600 K	0.12	$355 \times 330 \times 380$	0.81	5.61

details clearly. (c) and (d) show the effect of almost calm at the same frame. We notice that our method produces the same smoothness as [10], and our method is without the process of smoothing. Figure 5 shows an animation of flowing water on a stair scene with 600 K particles. It also shows a comparison of our method and [10]. Noticeably, droplet details are better preserved by our method.

The scalar field computation is implemented on the GPU using CUDA. We have enforced the parallel method to the scalar field computation of [10] and our method. Additionally, the performance is improved by using the optimized MC method [4] and neighborhood searching [8]. Table 1 shows the data of the presented scenes for scalar field computation, including the solver, the ratio of the number of particles, the resolution of MC grids. The last two columns list the averaged time per frame with our method and [10] respectively. The complexity and computational costs of our algorithm are less than the method of [10]. This is due to the extra overhead on the method of [10], i.e., anisotropy matrix computation and repetitive neighborhood search in each time step.

4. Conclusions

In this letter, we have presented an efficient method for high quality surface reconstruction. For computational efficiency, we only compute the scalar field around fluid surfaces. We introduce a new scalar field model based on an ellipsoidal smoothing function with variable smoothing length. For the computational accuracy, a constraint-correction rule is used to adaptively adjust smoothing length. Furthermore, the adaptive model is easy to implement, and it can be applied to standard SPH as well as PCISPH solver for surface reconstruction. In the future, we will extend this method to other visualizations such as viscoelastic and plastoelastic fluids.

This work is supported by National Basic Research Program of China (2010CB734104).

References

[1] Y. Kanamori, Z. Szego, and T. Nishita, "GPU-based fast ray casting for a large number of metaballs," Computer Graphics Forum, vol.27, no.2, pp.351–360, 2008.
[2] B. Adams, T. Lenaerts, and P. Dutre, "Particle splatting: Interactive rendering of particle-based simulation data," Technical Report, Katholieke Universiteit Leuven, 2006.
[3] R.A. Drebin, L. Carpenter, and P. Hanrahan, "Volume rendering," Proc. 15th Annual Conference on Computer Graphics and Interactive Techniques, pp.65–74, 1988.
[4] C. Dyken, G. Ziegler, C. Theobalt, and H.-P. Seidel, "High-speed marching cubes using histogram pyramids," Computer Graphics Forum, vol.27, no.8, pp.2028–2039, 2008.
[5] P. Simop and T. Tolga, "Particle-based simulation of fluids," EUROGRAPHICS 2003, vol.22, no.3, pp.137–146, 2003.
[6] Y. Zhu and R. Bridson, "Animating sand as a fluid," ACM Trans. Graphics, vol.24, pp.965–972, 2005.
[7] Y. Zhang, B. Solenthaler, and R. Pajarola, "Adaptive sampling and rendering of fluids on the GPU," Symposium on Volume and Point-Based Graphics, pp.137–146, 2008.
[8] P. Goswami, P. Schlege, B. Solenthaler, and R. Pajarola, "Interactive SPH simulation and rendering on the GPU," Symposium on Computer Animation, pp.55–64, 2010.
[9] H. Cords and G.S. Oliver, "Interactive screen-space surface rendering of dynamic particle clouds," J. Graphics, GPU and Game Tools, vol.14, no.3, pp.1–19, 2009.
[10] J. Yu and G. Turk, "Reconstructing surfaces of particle-based fluids using anisotropic kernels," Symposium on Computer Animation, pp.217–225, 2010.
[11] J. Monaghan and J. Lattanzio, "A refined particle method for astrophysical problems," Astron. Astrophys, vol.149, pp.135–143, 1985.