SURFACE EXTRACTION METHOD FOR PARTICLE-BASED SIMULATION USING IMPLICIT FUNCTION FITTING

Ken Taketani[†] Makoto Fujisawa[†] Masahiko Mikawa[†]

[†] University of Tsukuba

ABSTRACT

We propose a new surface extraction method for particle-based fluid simulations such as smoothed particle hydrodynamics (SPH). Our approach adapts implicit curve fitting commonly used for point-based rendering. Particle-based fluids generally use tens of thousands of particles, too few to extract a smooth surface using implicit curve fitting. We extract particles visible from a given viewpoint, and divide them into sub-groups based on their normal vectors. Implicit curve fitting is then performed against each group. This method allows representation of both smooth and sharp features, even with a limited number of elements. The most procedures are executed on a two-dimensional screen space.

1. INTRODUCTION

Particle-based fluid simulations have been widely used in computer graphics because of their simplicity and flexibility. However, extracting high-quality fluid surfaces from the results of simulation is a nontrivial task. Müller et al. [1] proposed an interactive particle-based simulation method that represents simulation results using metaballs with a kernel function, and renders using a marching cube method. However, this method still has problems in that blobby configurations appear on sharp edges, such as wave crests.

Yu et al. [2] generated smooth surfaces using an anisotropic kernel, and Adams et al. [3] successfully applied adaptive-sized particles by tracking particle-to-surface distances at each time step. However, these methods cannot fundamentally solve the problems we address, because they use density fields to extract surfaces.

Point-based rendering methods to reconstruct surfaces from point-clouds acquired from 3D scanners have also been developed. These methods use implicit curve fitting to reconstruct surfaces with both smooth and sharp features. Lancaster et al. [4] generated implicit surfaces from point cloud data using moving least squares. Carr et al. [5] developed the radial basis functions (RBF) to define shapes. Also, MPU (Multi-level Partition of Unity) implicit surfaces [6] and SLIM (Sparse Low-degree IMplicit) surfaces [7] were developed for non-conforming implicit surface representations. In these surface representations, each node has a support sphere, for which a low-degree implicit polynomial function is calculated. Surface positions and derivatives are calculated by the weighted sum of a set of function values if the point is included in several overlapping spheres.

This paper presents a novel surface extraction method that uses implicit function fitting for particle-based fluid simulations such as SPH. Particle-based fluids generally use tens of thousands of particles, too few to extract a smooth surface using implicit curve fitting. We first use visible particles from a given viewpoint to extract surface particles. The visible particles are then divided into sub-groups based on their normal vector, and the method calculates implicit surfaces for each group. In the grouping process, we use Delaunay triangulation and ghost particles to make a holeless surface, even if the number of particles is low. Finally, the method calculates implicit values for each point by integrating the groups. Our method can represent both smooth and sharp features, even for a limited number of elements. Most processes are implemented on a two-dimensional screen space, making the computational cost very low compared to a three-dimensional methods.

2. SYSTEM OVERVIEW

We assume that input is a set of particles $\mathbf{x}_1, \ldots \mathbf{x}_N \in \mathbb{R}^3$ simulated with SPH and a projection matrix $\mathbf{P} \in \mathbb{R}^{4\times4}$, and other user-defined parameters. Our method follows the procedures:

1. Project the particles on a screen space and detect the visible particle from a depth map.
2. Divide the visible particle into sub-groups according to their normal vector. (Figure1(a) (c))
3. Fit the implicit surface for each group. (Figure1(d)
4. Interpolate between the groups to create a depth map of the fluid surface. (Figure1(e))
5. Generate two-dimensional meshes from the depth map, and reversely project them into a three-dimensional world coordinate system.

Details of each process are described in the following sections.

2.1. DETECT VISIBLE PARTICLES

To calculate the implicit function that fits the fluid surface, we first extract visible particles from the viewpoint. All par-

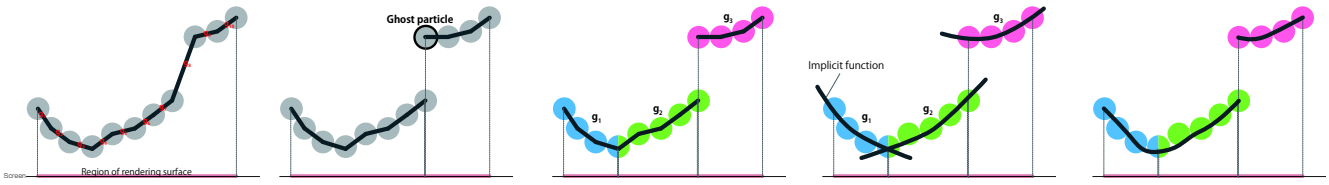

Figure 1: Overview of our method

ticles taken from the simulation are projected on a screen space, and its depth value from the viewpoint is calculated. The screen space defines a regular grid of cell size h with $n_x = \frac{W}{h} + 1$ nodes horizontally and $n_y = \frac{H}{h} + 1$ vertically. The depth map $\mathbf{D} \in \mathbb{R}^{n_x \times n_y}$ stores depth values $d_{i,j}$ at each cell.

First the depth values $d_{i,j}$ are initialized to ∞. Then, the position and radius of each particle are translated from the world coordinate system to the screen space. At this time, depth values from the viewpoint are calculated, and the smallest value and its index are stored in the depth map. The resulting depth map includes the indices of the visible particles.

2.2. GROUPING FOR SURFACE PARTICLES

It is difficult to represent complex shapes by one implicit function. Visible particles are therefore divided into subregions based on their normal vector.

Each group has a central position, a radius, and a normal vector, calculated from the average of the member particles. Particles with similar normal vectors form a group. The proposed method searches for particles with similar normals and positions, and groups them.

We can thus define groups representing the surface using a simple function, as shown in Fig. 2. Sharp features are retained using the interpolation methods described in section 2.4.

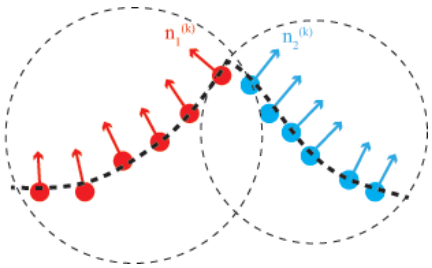

Figure 2: Grouping based on normal vectors

Grouping is done by the following procedure:

1. To create the first groups, which each have just three particles, we use Delaunay triangulation based on particle position $\mathbf{x}'_1, ..., \mathbf{x}'_N \in \mathbb{R}^2$ in the screen space. These triangles give information regarding nearest neighbor particles, and also define a region where the depth map for the surface will be created (Fig. 3(a)).

2. If the difference of the depth value of the three particles in a group exceeds a user-defined threshold d_{max}, the group is deleted to prevent a depth gap in the group. This process can generate holes on the region used for surface mesh creation, which are patched using ghost particles (Fig. 3(b)). Ghost particles have the same position as deleted particles, but with depth values equal to the largest value in the group.

3. Two neighborhood groups can be integrated if their normals are similar. This process is iterated until there are no new integrations.

$$\sum_{i=1}^{N} w(\mathbf{r}_i) \left(f(\mathbf{r}_i) - z_i\right)^2, \tag{1}$$

2.3. FITTING AN IMPLICIT FUNCTION TO EACH GROUP

After all particles are assigned to a group, we calculate an implicit function fit to the particle positions $\mathbf{r}_i = (x_i, y_i, z_i)$ $(i = 1, ..., M)$ in each group using the least squares method. In this paper, we adopt a quadratic implicit polynomial consisting of 10 coefficients as the fitting function. We then minimize the following equation to determine the coefficients:

$$\sum_{i=1}^{N} w(\mathbf{r}_i) \left(f(\mathbf{r}_i) - z_i\right)^2, \tag{2}$$

where $w(\mathbf{r})$ is a weighting function. (See [7] for details.)

2.4. CREATING THE DEPTH MAP

2.4.1. Region of rendering the surface

In order to decide a region of fluid surface on the depth map, we render the triangles that were created in grouping process. However, the polygonal shape appear on its contour and it will be unnatural appearance. To solve this problem, we use a bilateral filter to smooth the contour. In general, the

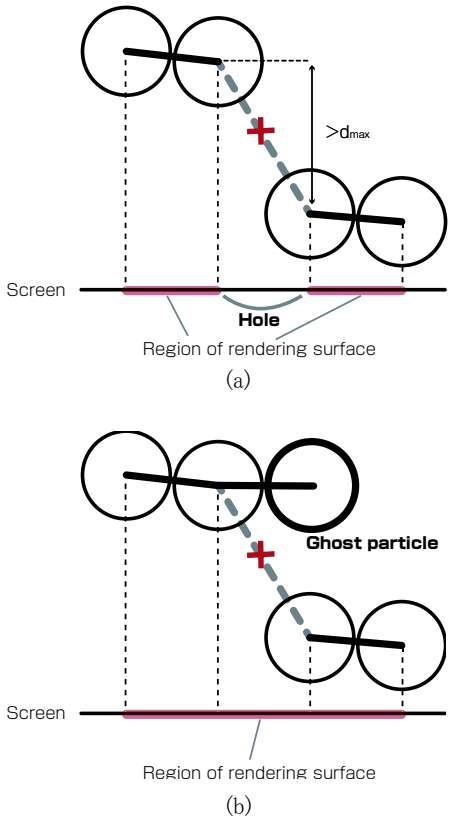

Figure 3: Ghost particles to create a holeless surface

bilateral filter is used two weight functions for the distance between pixels and for the luminance difference. Instead of using the luminance difference, this paper uses the difference of normal vector. By using the normal vectors of each pixel obtained from the normal of the group, the contour can have both the smooth and sharp features. The equations for bilateral filter is followings.

$$z_{i,j} = \frac{\sum_n \sum_m z_{i+m,j+n} w(m,n)}{\sum_n \sum_m w(m,n)}, \tag{3}$$

where $w(x,y) = w_s w_n$ is a combined weight function for the filter and

$$w_s = \frac{1}{2\pi\sigma_s} \exp(-\frac{x^2+y^2}{2\sigma_s^2}), \tag{4}$$

$$w_n = \frac{1}{2\pi\sigma_s} \exp(-\frac{(f_n(i,j) - f_n(i+m,j+n))^2}{2\sigma_n^2}), \tag{5}$$

where f_n is a function to return the normal vector on each pixels.

2.4.2. Interpolation between the groups

We modify the depth map $\mathbb{D}$ for reconstructing smooth surfaces by interpolating the implicit function between overlapped groups. Interpolation parameters are changed depending on the similarity of the group normal to retain sharp features. Figure 4 shows a conceptual diagram of this.

We first detect all groups in a ray from the viewpoint using billboards, and calculate the intersection between the ray and the implicit function. If there is just one group in the

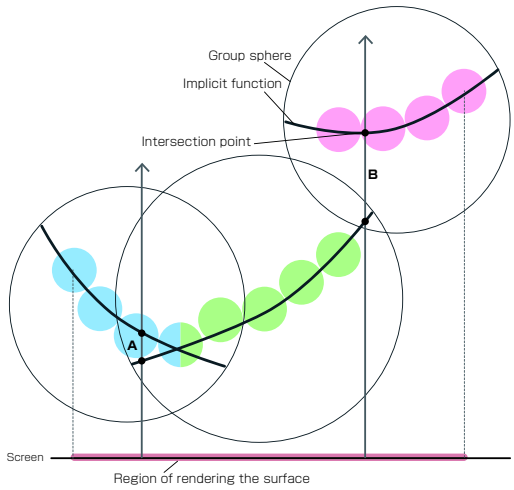

Figure 4: Depth map integration

ray from the viewpoint on the surface rendering region, the distance to its intersection is stored in the depth map as the depth value. If there are multiple groups and the difference of the depth value between these groups doesn't exceeds a user-defined threshold d_{max} (like Figure4 A) , an interpolated distance to the intersection points from the viewpoint is stored into the depth map. Then, the modified depth value $d'_{i,j}$ is calculated as

$$d'_{i,j} = \frac{\sum_k \beta_k c_k}{\sum_j \beta_k}, \tag{6}$$

where c_k is intersection points from the viewpoint on the ray, β_k is a weight function calculated by the distance between the group center and a cell on the screen space, as follows:

$$\beta_k(d_k, b) = \frac{1}{\sqrt{2\pi b^2}} \exp\left(-\frac{d_k^2}{2b^2}\right). \tag{7}$$

Here, b is a variable that is monotonically increasing with group similarity. This paper uses the dot product of the normals. If the similarity of the groups is high, β_k drastically increases.

2.5. GENERATE MESHES

We generate two-dimensional surface meshes from the depth map using screen space meshes [8]. First, the world coordinate $[w, y, z]^T$ is computed with the inverse of the projection matrix $\mathbf{Q} \in \mathbb{R}^{4*4}$. After the mesh transformation, we generate vertex normals in the world coordinate

system using the weighted sum of the normals of adjacent triangles. Finally the triangles and normals are sent to the graphics pipeline.

3. RESULTS AND DISCUSSION

Figure 5 shows the rendered results of a breaking dam using (a) the traditional screen space method [8] and (b) our method. The rendering speed for (a) is 29.6 fps, and for (b) is 2.5 fps on a standard PC with a Core i7-4770 3.40 GHz CPU. As the figure shows, (b) is rendered while better retaining smoothness than does (a). Figure 6 shows that our method allows representation of both smooth and sharp feature, even if the number of particles is low.

Our method has some limitations. If there are few particles in a neighborhood, such as in a splash, our method cannot be adopted because fitting requires more than three particles. In this case, we directly render the particles.

4. CONCLUSION AND FUTURE WORK

We presented a novel surface extraction method for particle-based fluid simulations using implicit function fitting, and applied the method to represent a fluid surface with smooth and sharp features. To adapt implicit surface fitting to particle-based simulations, we grouped particles using Delaunay triangulation and used ghost particles to patch any resulting holes. Most processes were implemented in a two-dimensional screen space, so the computational cost to create the surface is very low compared to other three-dimensional surface extension methods.

Future work will include methods for GPU-based parallel processing and a method for better rendering splashes.

ACKNOWLEDGMENT

This work was supported by JSPS KAKENHI Grant Number 25730069.

5. REFERENCES

[1] M. Müller, D. Charypar and M. Gross, “Particle-based fluid simulation for interactive applications”, In Proceeding of the ACM SIGGRAPH Symposium on Computer Animation, pp.154-159, 2003.

[2] J. Yu and G. Turk, “Reconstructing surface of particle-based fluids using anisotropic kernels”, In Proceeding of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp.217-225, 2010.

[3] B. Adams, M. Pauly, R. Keiser and J. L. Guibas, “Adaptively sampled particle fluid”, ACM Transactions of Graphics (Proc. SIGGRAPH2007), 26(3), 48, 2007.

[4] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin and C. T. Silva, “Computing and rendering point set surfaces”, IEEE Transactions on Visualization and Computer Graphics, 9(1), pp.3-15, 2003.

[5] J. Carr, R. Beatson, J.Cherrie, T. Mitchell, W. Fright, B. McCallum and T. Evans, “Reconstruction and representation of 3D objects with radial basis functions”, In Proceeding of SIGGRAPH 2001, pp.67-76, 2001.

[6] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk and H.-P. Seidel, “Multi-level partition of unity implicits”, ACM Transactions on Graphics (Proc. SIGGRAPH2003), 22(3), pp. 463-470, 2003.

[7] T. Kanai, Y. Ohtake, H. Kawata and K. Kase, “GPU-based rendering of sparse low-degree implicit surfaces”, In GRAPHITE ’06: Proceedings of the 4th International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia, pp.165-171, 2006.

[8] M. Müller, S. Schirm, and S. Duthaler, “Screen space meshes”, In Proceeding of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp.9-15, 2007.

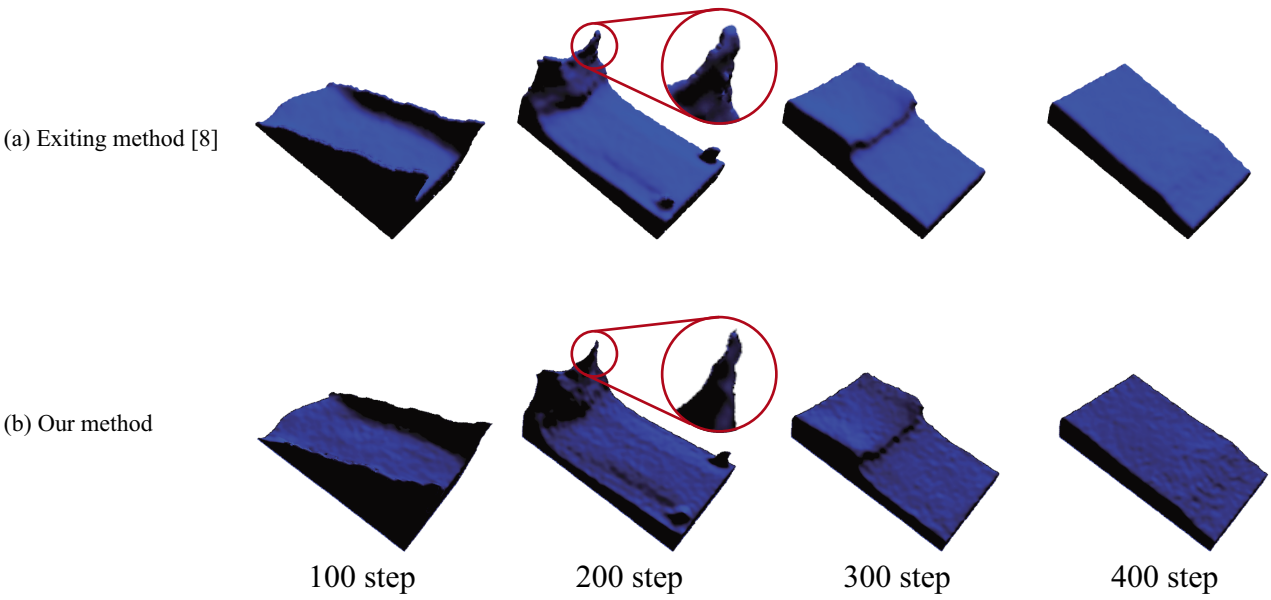

Figure 5: Rendered results of a simulated breaking dam using 35937 particles

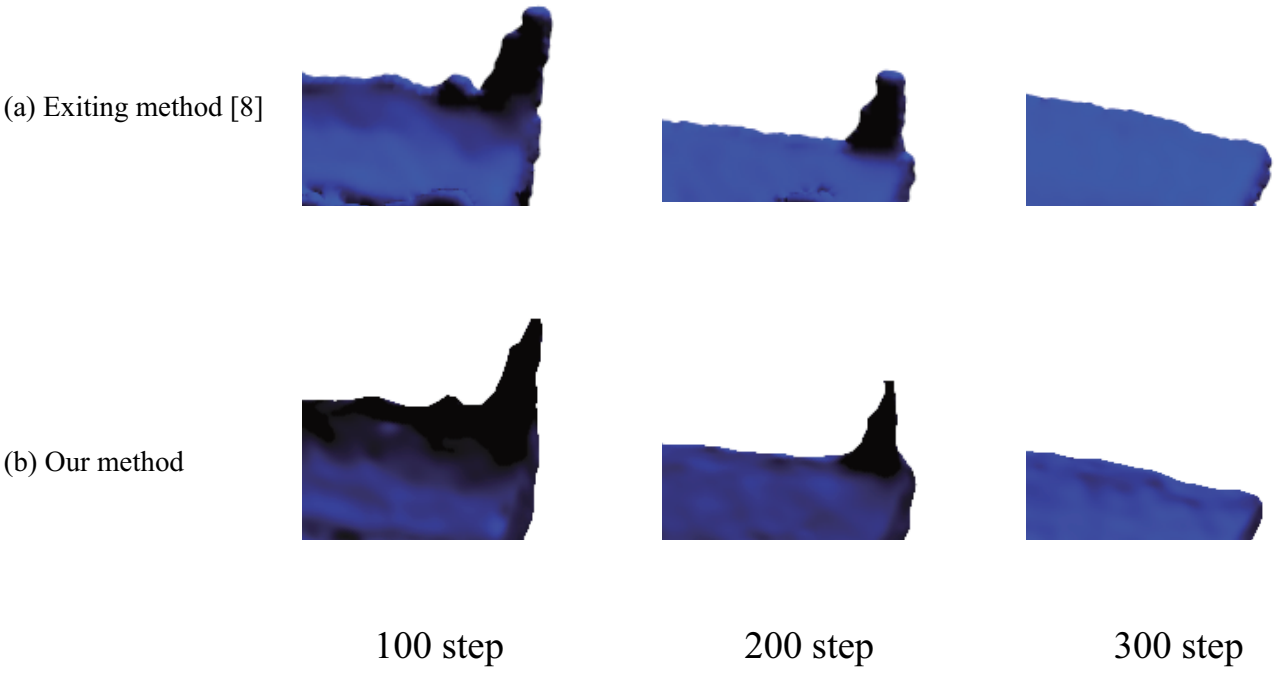

Figure 6: Rendered results of a simulated breaking dam using 5000 particles