Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

# Surface reconstruction method for particle-based fluids using discrete indicator functions

Filomen Incahuanaco[a,*], Afonso Paiva[a]

[a]*ICMC, Universidade de São Paulo, São Carlos, Brazil*

ARTICLE INFO

*Article history*:
Received June 3, 2023

*Keywords:* surface reconstruction, particle-based fluids, discrete indicator function

ABSTRACT

Particle-based methods are practical for computing level-sets that represent liquid interfaces. However, these methods are computationally expensive to reconstruct the liquid surface when the number of particles increases considerably due to the massive amount of particle approximations. This paper introduces two simple and efficient surface reconstruction methods for particle-based fluids based on discrete indicator functions (DIFs). The first approach provides fast level-set approximation using a DIF defined by counting particles inside grid cells. The second approach generates a high-quality liquid surface using a DIF obtained by the particle distribution inside grid cells. Our DIF-based methods are fast, easy to code, and can be adapted straightforwardly in particle-based fluid solvers, even implemented in GPU. Moreover, we show the effectiveness of our approaches through experiments against prior surface reconstruction methods.



## 1. Introduction

Particle-based fluid methods, such as Smoothed Particle Hydrodynamics (SPH) [1] and Position-Based Dynamics (PBD) [2], particles are usually employed to track the air-liquid interface in Lagrangian fluid flow simulations. Although these methods have been successfully used in interactive applications, rendering complex liquid animations with a high number of particles at reasonable computational times, even in GPU architectures, remains a subject of intense research in computer animation.

The rendering of air-liquid interfaces in particle-based fluids consists of two stages: first, splatting the level-set function defined by the particles to a regular grid. Second, the liquid surface reconstruction is given by the isosurface extracted from a discrete level-set using a polygonization algorithm, such as Marching Cubes (MC) [3], where a polygonal mesh represents the isosurface. However, this entire process is computationally expensive since the surface's smoothness and topological correctness require a high-resolution grid.

On the other hand, in computational fluid dynamics, an indicator function $\mathbb{1}_F$ (i.e., a function that is one on the interior of the fluid body $F$ and zero on the exterior) is usually employed for tracking liquid interfaces [4]. Let $\mathcal{G}$ be a regular grid of resolution $n_x \times n_y \times n_z$ that enclosures the domain. The key idea is to compute a local fraction of volume occupied by the fluid in each grid cell $K \in \mathcal{G}$, as follows:

$$\vartheta(K) = \frac{1}{V_K} \int_K \mathbb{1}_F(\mathbf{x})\, d\mathbf{x}\,, \tag{1}$$

where $V_K$ is the volume of $K$. The scalar field $\vartheta : \mathcal{G} \to [0, 1]$ is known as *discrete indicator function* (DIF). Besides, there are robust surface reconstruction methods from DIFs in the literature [5, 6].

This paper presents a novel and practical surface reconstruction for particle-based fluid methods. We propose two efficient approximations of a DIF using the particles inside grid cells. Therefore, we show the effectiveness of our approach through a set of comparisons against prior surface reconstruction methods. Figure 1 shows our DIF-based methods in action.

We summarize below the main contributions of our approach. Some parts were published in our previous paper [7] and appear in this extended version. New material includes a novel DIF

*Corresponding author: F. Incahuanaco (fincahuanaco@usp.br)

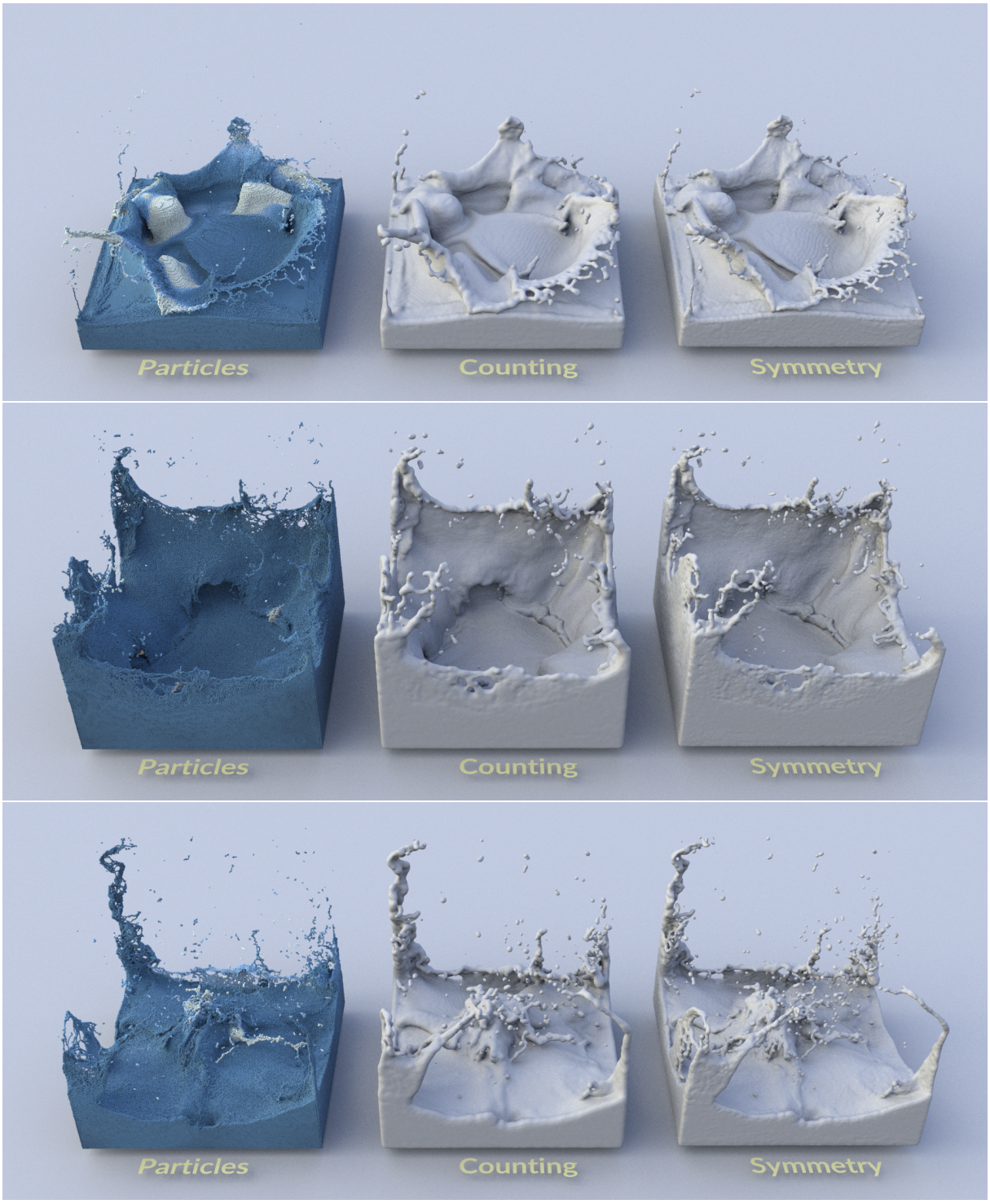



Fig. 1: DIF-based surface reconstruction of a liquid toy dragon with 5M SPH particles in a grid with $384^3$ cells: from left to right, the input particles, our DIF by counting, and our DIF by symmetry.

based on particle distribution and a scheme to recover the small-scale details (e.g., droplets and liquid ligaments) in the surface reconstruction. Thus, the contributions of the paper are:

- A novel surface reconstruction method for meshfree liquid simulations based on DIFs;
- We propose two novel DIFs that use only the particles inside the grid cells;
- A scheme to recover small-scale details in surface reconstruction using union Boolean operation on DIFs;
- Our framework speeds up considerably the surface reconstruction compared with prior methods;
- Our methods are simple and easy to code, even in GPU.

**Paper outline.** The remainder of the paper is organized as follows. Section 2 presents a brief review of related methods in the literature. The proposed reconstruction methods are described in Section 4. Section 5 and Section 6 provide the results and a discussion about our methods giving a glimpse of future work, respectively. Section 7 concludes the paper.

## 2. Related work

To better contextualize our method and highlight its properties we organize the existing methods for particle-based fluid rendering into two main groups: *mesh-based* and *screen-space* methods.

**Mesh-based methods.** The main goal of these methods is to extract a smooth triangle mesh from the particle positions using MC-based algorithms. Typically, the liquid surface is represented implicitly by the zero level-set of a scalar field computed from a weighted sum of kernel evaluations from the particles' distances. These methods can use isotropic kernels [8, 9], adaptive size kernels [10] or anisotropic SPH kernels [11]. Despite the existence of parallel implementations of these methods [12, 13, 14], if the liquid spreads more over the computational domain, the underlying MC grid and its resulting surface mesh become very large, causing excessive memory consumption. Bhattacharya et al. [15] improved the undesired blobby appearance of the level-sets using a smoothing process by solving a constrained optimization problem. Sandim et al. [16] proposed an alternative framework for surface reconstruction. Their framework relies on a level-set definition using the Hermite data (particle positions and normals) from the boundary particles. The liquid surface is obtained fitting the boundary particles using Screened Poisson surface reconstruction [17]. However, this method also suffers the same issues of the kernel-based methods.

**Screen-space methods.** This class of methods performs in 2D image space using a smoothed depth buffer from the visible liquid surface defined by spherical particles, where the resulting surface is represented without mesh generation by using rasterization techniques [18, 19, 20, 21, 22]. The liquid surface's visual quality relies on the depth buffer's image-based filtering process, which may demand large convolution kernels and perform multiple filter iterations. However, beyond the screen-space size and the number of filter passes, these methods' efficiency also depends on the number of particles. Recently, Oliveira and Paiva [23] improved the prior screen-space methods computing volumetric rendering effects in a small subset of particles located at a narrow-band of the air-liquid interface.

Despite the proposed approach belonging to the class of mesh-based methods, our method was strongly influenced by screen-space methods, extending the filtering process to a 3D image (a DIF in our case).

## 3. An overview of surface reconstruction from particles

Usually, in particle-based fluids, the liquid surface $S$ is represented by an isosurface extracted from a scalar field $\phi$:

$$S = \phi^{-1}(c) = \{\mathbf{x} \in \mathbb{R}^3 \mid \phi(\mathbf{x}) = c\},$$

where the scalar $c$ is an isovalue. A typical choice of $c$ is zero or a small positive value to obtain a smooth surface.

Müller et at. [8] proposed an SPH approximation of an indicator function that assumes value one at particle positions $\mathbf{x}_i$ and zero everywhere else in the domain. The scalar field around a grid point $\mathbf{x}$ of $\mathcal{G}$ is defined as

$$\phi(\mathbf{x}) = \sum_i W(\mathbf{x} - \mathbf{x}_i, h)\frac{m_i}{\rho_i}, \tag{2}$$

where $m_i$ is the particle mass and $\rho_i$ is the particle density. The function $W$ is an isotropic *kernel* with smoothing length $h$ given by

$$W(\mathbf{r}, h) = s\, w\left(\frac{\|\mathbf{r}\|_2}{h}\right) \quad \text{with} \quad w(x) = \max\left(0, (1 - x^2)^3\right), \tag{3}$$

where $s$ is a scale parameter. In SPH, the value of $s$ is derived from the unitary property of the kernel, i.e., $\int W(\mathbf{r}, h)d\mathbf{r} = 1$. For the kernel (3), we have $s = 315/64\pi h^3$. The value of $h$ defines the radius of influence of the kernel, typically chosen as the initial particle spacing.

Despite the simplicity of the previous approach, irregular particle distributions may result in visible surface bumps [11]. In order to produce a smoother surface, Zhu and Bridson [9] defined a scalar field based on the weighted average of neighboring particle positions $\bar{\mathbf{x}}$. Considering $s = 1$ in Equation (3), the scalar field is given by:

$$\phi(\mathbf{x}) = \|\mathbf{x} - \bar{\mathbf{x}}\|_2 - r \quad \text{with} \quad \bar{\mathbf{x}} = \frac{\sum_j \mathbf{x}_j\, W(\mathbf{x} - \mathbf{x}_j, 4r)}{\sum_j W(\mathbf{x} - \mathbf{x}_j, 4r)},$$

where $r$ is the particle radius, i.e., half of the particle spacing.

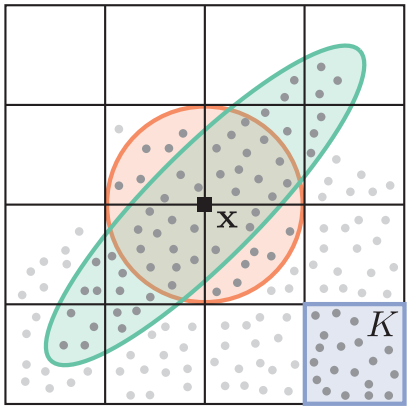
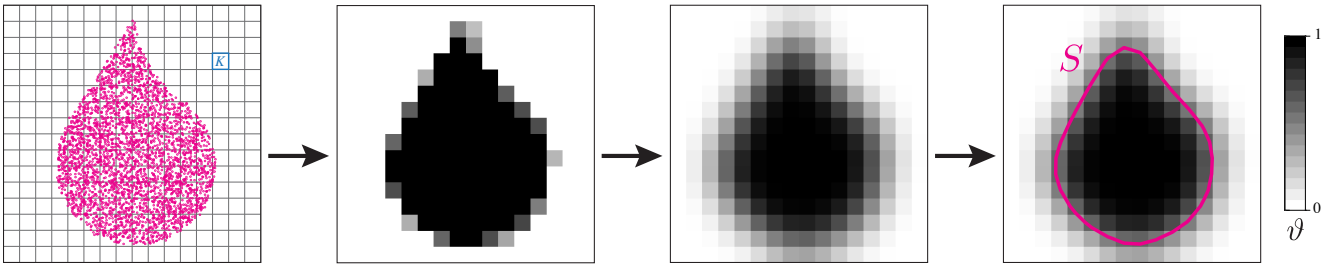



Fig. 2: Discrete scalar field $\phi$ computed from the particles (gray dots). The standard reconstruction methods use the particles inside the region of influence of the kernel to evaluate $\phi$ at a grid point $\mathbf{x}$ using an isotropic (orange) [8, 9] or an anisotropic (green) [11] kernel. In contrast, our DIF-based approach [7] evaluates $\phi$ at a grid cell $K$ using only the particles inside $K$ (blue).




(a) input particles (b) DIF evaluation (c) smoothing (d) isosurfacing

Fig. 3: Surface reconstruction pipeline.

A weakness of these methods mentioned above is that the spherical shape of the isotropic kernel makes it challenging to reconstruct sharp features and flat surfaces. For this reason, Yu and Turk [11] developed an anisotropic kernel that captures the particle distribution near to interface more accurately:

$$W(\mathbf{r}, \mathbf{G}) = s \det(\mathbf{G})\, w(\|\mathbf{G}\mathbf{r}\|_2) ,$$

where $\mathbf{G}$ is an anisotropy matrix determined by the weighted Principal Component Analysis (WPCA). Thus, the scalar field 2 can be rewritten as follows:

$$\phi(\mathbf{x}) = \sum_i W(\mathbf{x} - \bar{\mathbf{x}}_i, \mathbf{G}_i) \frac{m_i}{\rho_i} .$$

The ellipsoidal shape of the kernel in reconstruction provides high-quality surfaces without a blobby appearance. However, the process is computationally expensive due to the calculation of each particle's anisotropic matrix $\mathbf{G}_i$.

Our previous work [7] presented a new strategy for computing the scalar field using a DIF. Instead of evaluating the scalar field at the grid nodes by kernel weighting, we use a simple approximation of Equation (1) for each grid cell $K \in \mathcal{G}$ to yield the scalar field (see Figure 2. In the next section, we will detail our method.

## 4. The method

In this section, we explain the pipeline of the proposed surface reconstruction method. Given an input particle system $\mathcal{P}^t$ at time-step $t$, our method performs three main steps, as illustrated by Figure 3.

### 4.1. DIF evaluation

This section presents two ways to approximate a DIF from particles. The first strategy is based on counting particles inside the grid cells. In contrast, the second relies on the particle distribution inside the cells. In the following, we will present the two strategies.

**DIF by counting.** We approximate the local volume of fluid simply by counting the particles inside each cell $K \in \mathcal{G}$. Let $N_K^t$ be the number of particles of $\mathcal{P}^t$ inside $K$. Firstly, we compute the initial particle average $\mu_0$ (at time-step $t = 0$) given by:

$$\mu_0 = \frac{1}{|\mathcal{F}|} \sum_{K \in \mathcal{F}} N_K^0 , \tag{4}$$

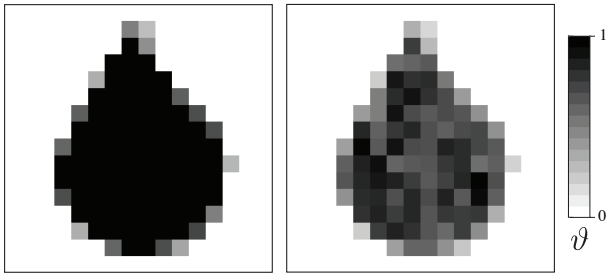



Fig. 4: DIF by counting from the input particles (Figure 3a) using the particle average (*left*) and the maximum number of particles (*right*).

where the operator $|\cdot|$ denotes the set's cardinality and $\mathcal{F} \subseteq \mathcal{G}$ is the subset of *full cells* of $\mathcal{G}$, i.e., formed by cells that contain particles in their interior, otherwise, the cell is classified by *empty*. Assuming that the cell volume $V_K$ is entirely occupied by the volume of $\mu_0$ particles, i.e., $V_K = \mu_0 V_p$, where $V_p$ is the particle volume. Thus, we approximate the Equation (1) as follows:

$$\vartheta(K) = \frac{N_K^t\, V_p}{V_K} = \frac{N_K^t\, V_p}{\mu_0\, V_p} = \frac{N_K^t}{\mu_0} . \tag{5}$$

In order to ensure $\vartheta(K) \in [0, 1]$, the fluid occupancy in a cell $K$ in our approach is given by:

$$\boxed{\vartheta(K) = \frac{\min\{N_K^t, \mu_0\}}{\mu_0}} . \tag{6}$$

Since the DIF decreases from the interior of the liquid to its interface, the particle average $\mu_0$ reduces the variation of the DIF $\vartheta$ in *internal* cells, i.e., full cells that are not adjacent to empty cells. For instance, Figure 4 shows a comparison when we take $\mu_0 = \max_K\{N_k^0\}$ instead of Equation 4. Note that the particle average (4) reduces the DIF's noise.

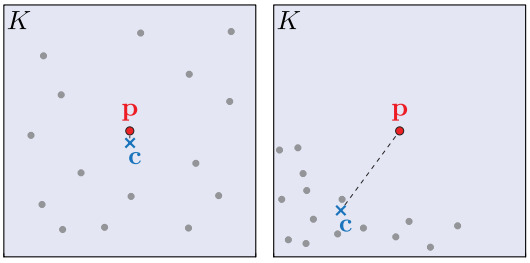



Fig. 5: DIF from particle distributions (gray dots) in a cell $K$: the particle centroid $\mathbf{c}$ (dark blue marker) from a symmetric distribution (*left*) is closer to the cell center $\mathbf{p}$ (red dot) than in an asymmetric distribution (*right*).

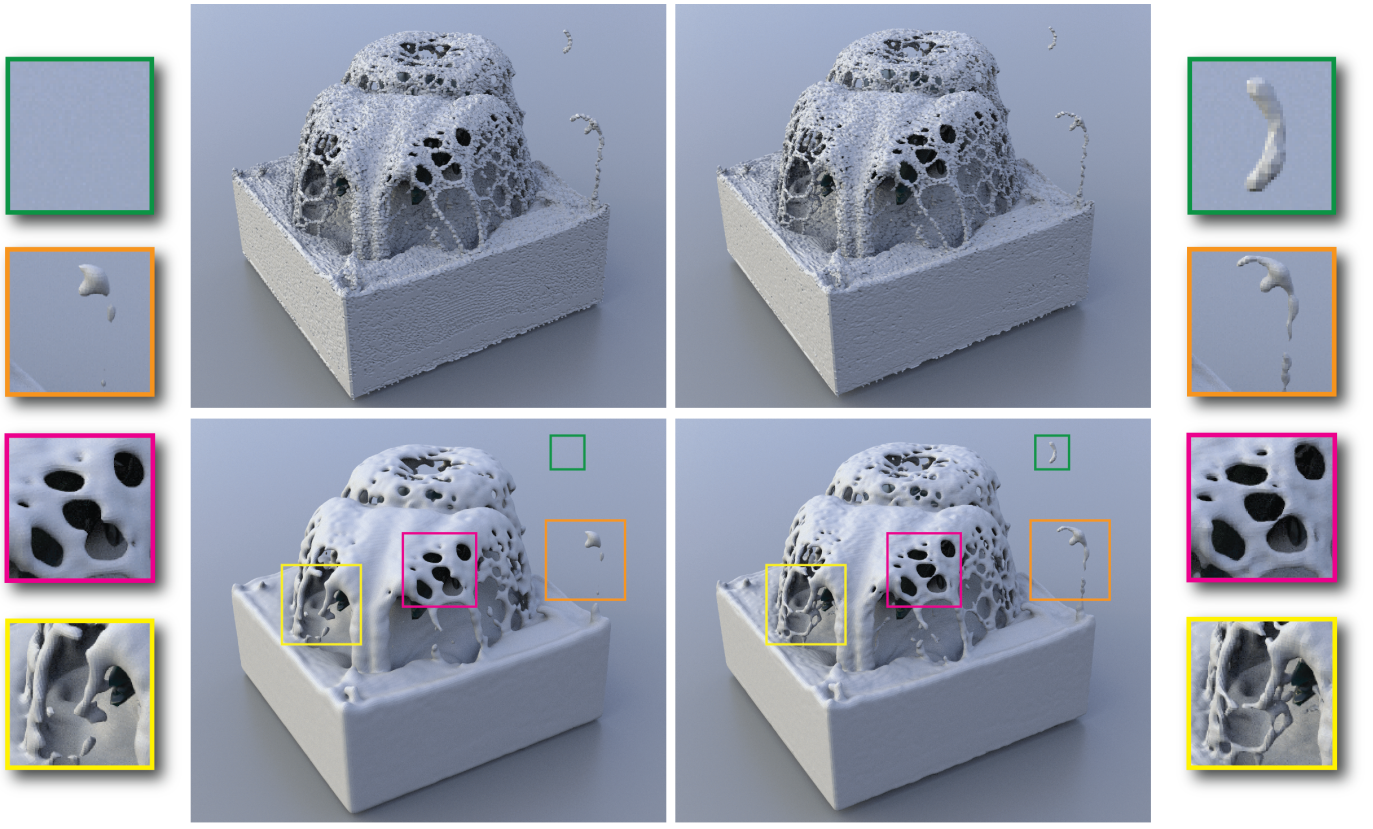


Fig. 6: DIF-based surface reconstruction without smoothing (*top*) and with smoothing (*bottom*) of a spaceship rising from underwater with 4.76M SPH particles in a grid with $384^3$ cells: comparison between DIF by counting (*left*) and symmetry (*right*). As can be seen in zoom-in regions, the DIF by symmetry preservers more details of the liquid surface than DIF by counting.

**DIF by symmetry.** Since the particles are well distributed in internal cells and due to the asymmetric particle distribution in cells close to the liquid interface because there are no particles outside the liquid surface, we can estimate the cell occupancy by considering the symmetry of the particle distribution around the cell center. The *particle centroid* in a cell $K$ is given by $\mathbf{c} = \sum_i \mathbf{x}_i / N_K^t$, where $\mathbf{x}_i \in K$. For symmetric particle distributions in a cell $K$, the particle centroid $\mathbf{c}$ is close to the cell center $\mathbf{p}$ of $K$ (see Figure 5). Therefore, we can approximate the DIF based on the distance between the particle centroid and the cell center normalized by the half length of the cell diagonal, as follows:

$$\boxed{\vartheta(K) = 1 - 2\frac{\|\mathbf{c} - \mathbf{p}\|_2}{\mathrm{diag}(K)}\,.} \tag{7}$$

Note that in symmetric distributions, the DIF (7) assumes values close to 1. Otherwise, $\vartheta(K)$ reaches values close to zero.

The DIF by symmetry produces less surface aliasing than by counting, especially in flat regions, as illustrated by Figure 6.

### 4.2. DIF smoothing

To reduce the aliasing of the liquid surface, we smooth the DIF $\vartheta$ by applying a 3D version of well-known blur filters from image processing [24], e.g., box and Gaussian filters. In addition, robust and efficient GPU implementation of these filters can be found in the VMD library [25].

More specifically, in our DIF-based approach, we apply a Gaussian filter $G_\sigma$ with standard deviation $\sigma$ to enhance the DIF, where the value of $\sigma$ defines the filter mask size $\max\{3, 2 \cdot \lfloor 3\sigma \rfloor - 1\}$. However, due to the high aliasing effect in DIF by counting, we apply an additional box filter with a size mask of $3 \times 3 \times 3$. In our experiments, this strategy provided better results, with less surface dissipation, than applying $G_\sigma$ in a single pass with a large filter size ($\sigma \geq 1.5$). We denote the smoothed DIF by $\tilde{\vartheta}$.

Figure 6 compares the resulting surface from DIF by counting and symmetry. The smoothed DIF by symmetry preserves the surface details because it requires only one smooth filtering pass, avoiding the features' dissipation.

Figure 7 shows the effect of the parameter $\sigma$ in Gaussian filter $G_\sigma$. The surface becomes smoother when we increase the parameter $\sigma$. Nevertheless, thin and small surface details can disappear for high values of $\sigma$.

### 4.3. Isosurfacing

In our method, for each cell $K_i \in \mathcal{G}$, we assume that the smoothed DIF $\tilde{\vartheta}$ is sampled at cell centers $\mathbf{p}_i$ of $K_i$. Thus, once computed the field $\tilde{\vartheta}$, we have fractional volumes of fluid $\tilde{\vartheta}_i$ located at the centers $\mathbf{p}_i$. The liquid surface $S$ is represented by the zero level-set of a discrete scalar field given by:

$$\phi_i = \phi(\mathbf{p}_i) = c - \tilde{\vartheta}(K_i) \quad \text{with} \quad c \in (0, 1)\,. \tag{8}$$

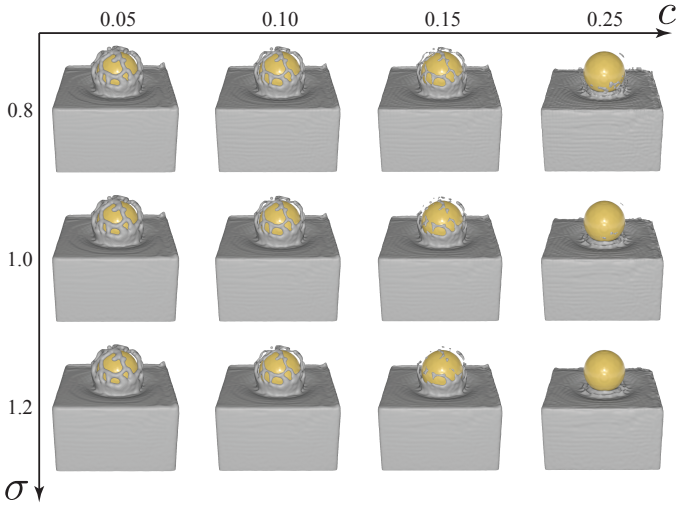



Fig. 7: Effect of varying the parameters: the DIF-based surface reconstruction using different combinations of Gaussian filters $G_\sigma$ and isovalues $c$. The challenge is choosing a set of parameters so the smooth surface preserves details and does not seem like a blob.

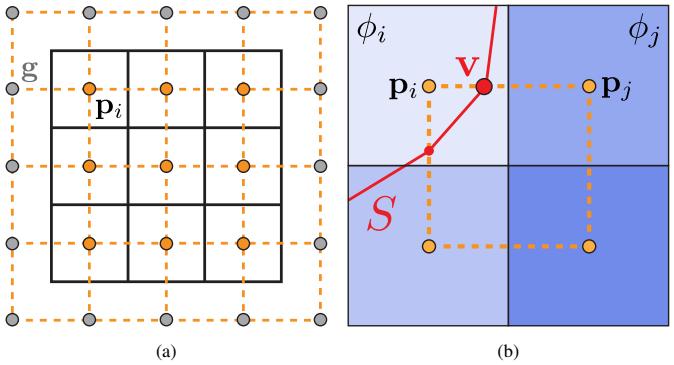



Fig. 8: Isosurfacing of a DIF in 2D: (a) a dual grid (dashed orange line) is obtained from the grid $\mathcal{G}$ (solid black lines). The dual grid points are formed by the cell centers $\mathbf{p}_i$ (orange dots) of $K_i \in \mathcal{G}$ and by ghost nodes $\mathbf{g}$ (gray dots). (b) For each dual cell, MC examines the configuration of sign($\phi_i$) at the corners to define the local topology of the surface $S$ (solid red line) and determines the intersection points (red dots) between $S$ and the dual edges.

The field $\phi_i$ assumes negative values inside $S$ and positive values outside $S$.

We extract the isosurface of $S$ executing the MC algorithm in the *dual grid*. The cells of the regular dual grid are obtained by connecting the centers $\mathbf{p}_i$ of the adjacent cells of $\mathcal{G}$. We complete the dual grid connecting $\mathbf{p}_i$ with *ghost points* $\mathbf{g}$ placed outside of the domain (see Figure 8a). To ensure the surface matches the boundary of $\mathcal{G}$, we use linear extrapolation to set the discrete field at the ghost points. For the edges $\langle \mathbf{p}_i, \mathbf{g} \rangle$ where $\mathbf{p}_i$ belongs to a full cell $K_i$, we have $\phi(\mathbf{g}) = -\phi_i$, otherwise, $\phi(\mathbf{g}) = \phi_i$.

The MC's lookup table determines the local topology of $S$ inside each dual cell by indexing the configurations of sign($\phi_i$) at the eight corners of the cell (see Figure 8b). Considering a linear approximation of the scalar field, given a dual edge $e_{ij} = \langle \mathbf{p}_i, \mathbf{p}_j \rangle$ where $\phi_i \cdot \phi_j \leq 0$, the intersection point (vertex) $\mathbf{v}$ of $S$ with $e_{ij}$ is computed as follows:

$$\mathbf{v} = (1 - \alpha)\,\mathbf{p}_i + \alpha\,\mathbf{p}_j \quad \text{with} \quad \alpha = \frac{\phi_i}{\phi_i - \phi_j}\,.$$

After processing each dual cell, the entire surface $S$ is extracted.

Figure 7 shows that the isovalue choice affects the thickness of the reconstructed surface. The liquid surface becomes thinner when we increase the values of $c$, and conversely. The resolution of $\mathcal{G}$ affects surface quality and computational efficiency of the poligonization. Some authors recommend that a cell size less than or equal to the particle spacing is necessary to capture thin and small-scale details [12].

## 5. Results

We implemented our approach in C++ and a parallel version of our code on GPU using CUDA. The particle-based fluid simulations were produced with SPH using the computational framework provided by DualSPHysics [26]. All results have been achieved using a computer with AMD Ryzen 9 3950X and 32GB RAM and NVIDIA GeForce RTX 2070 with 8GB VRAM.

Figures 9, 10, 11, and 12 show comparisons of our DIF-based methods applied in different simulations in comparison with previous surface reconstruction methods proposed by Müller et al. [8], Zhu and Bridson [9], and Yu and Turk [11]. Furthermore, the implementations in C++ of these methods can be found in the Github[1] repository from Kim's book [27].

Table 1 shows the computational times and some statistics for a set of experiments presented in this section. The column $|\mathcal{P}|$ is the number of particles, the column **res** is the resolution of $\mathcal{G}$, the column $c$ is the isovalue used in Equation (8) to extract the isosurface, and the column $\sigma$ is the filter size used in our experiments. Each method's **time** average and standard deviation *std* (in parenthesis) across all animation frames were measured using a single-core CPU. Note that our DIF by counting and symmetry are 2× and 1.8× faster in the worst cases and 30× and 26× faster in the best cases, respectively, demonstrating our methods' efficiency.

In order to perform a quantitative analysis of the reconstruction methods, given a triangle surface mesh (ground truth), we create a regular particle sampling that fills the mesh volume. The reconstruction error is the *Haussdorf distance* [28] between the ground truth mesh and the reconstructed surface. Figure 13 presents the results of this error analysis in the Stanford Bunny. As can be seen, our DIF by symmetry is the most accurate method.

## 6. Discussion

In this section, we highlight the limitations present in our approach and potential improvements to the current state of the surface reconstruction method and future projects we are working on. Also, we discuss the scalability and the computational time consumption of each stage of our reconstruction pipeline.

[1] https://github.com/doyubkim/fluid-engine-dev

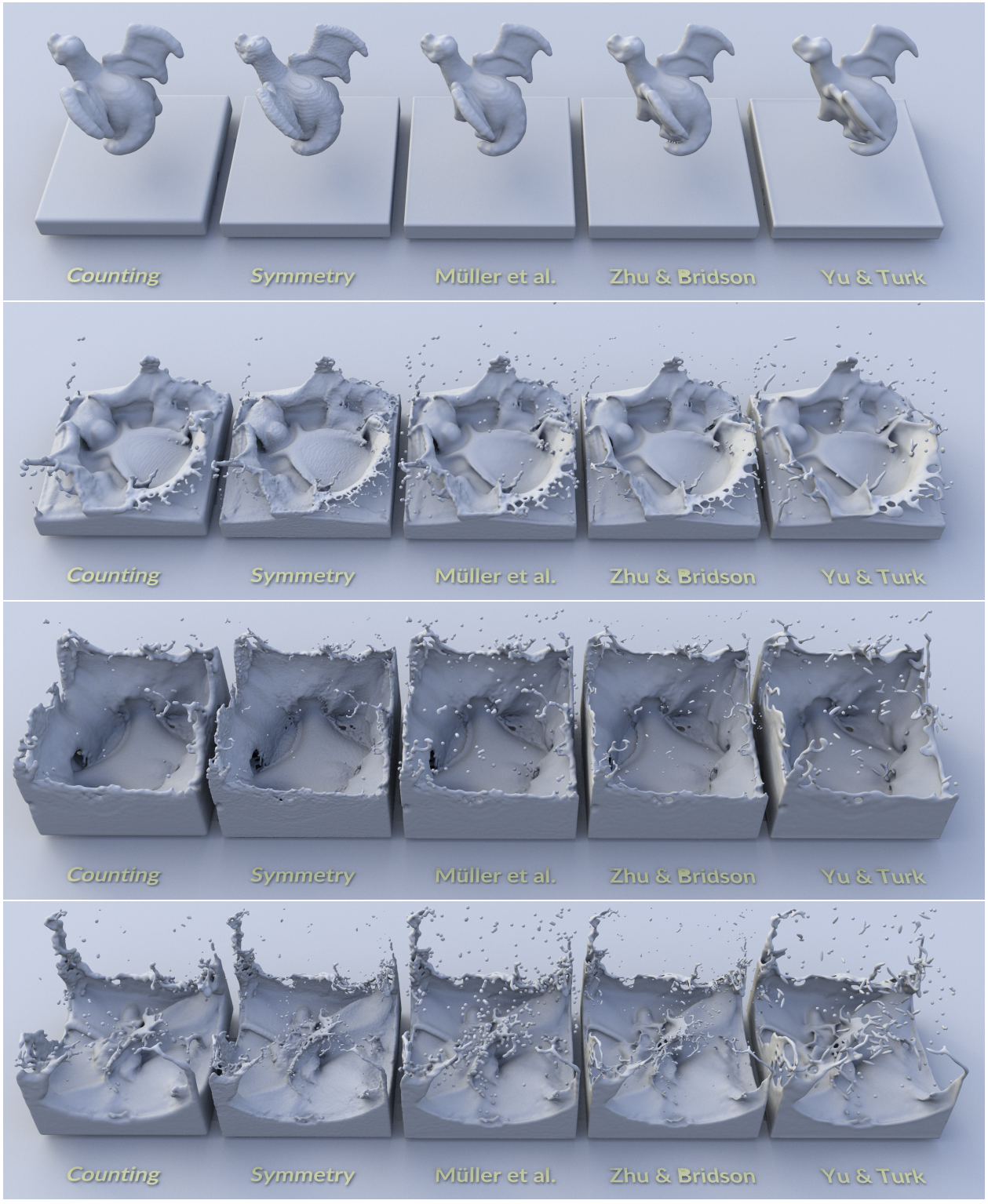



Fig. 9: Surface reconstruction of a liquid toy dragon with 1.2M SPH particles in a grid with $256^3$ cells: from left to right, our DIF by counting, our DIF by symmetry, Müller et at. [8] , Zhu and Bridson [9], Yu and Turk [11]. Our DIF by counting and symmetry are 2.2× and 2.1× faster than the best competitor method (Zhu and Bridson), 11.6× and 11.2× faster than the anisotropic method (Yu and Turk), respectively.

### *6.1. Scalability and profiling*

Figure 14 shows the computational timing of our approach using the DIF by counting, our fastest method, implemented on GPU and the performance profiling of each stage of the reconstruction pipeline (as shown in Figure 3) with different resolutions of $\mathcal{G}$. As can be seen, the computational time related to grid operations to produce the smoothed DIF increases when we refine the grid, becoming a potential bottleneck in high-resolutions. Regarding the rendering, our method preserves nicely small-scale liquid details even in a grid with a resolu-

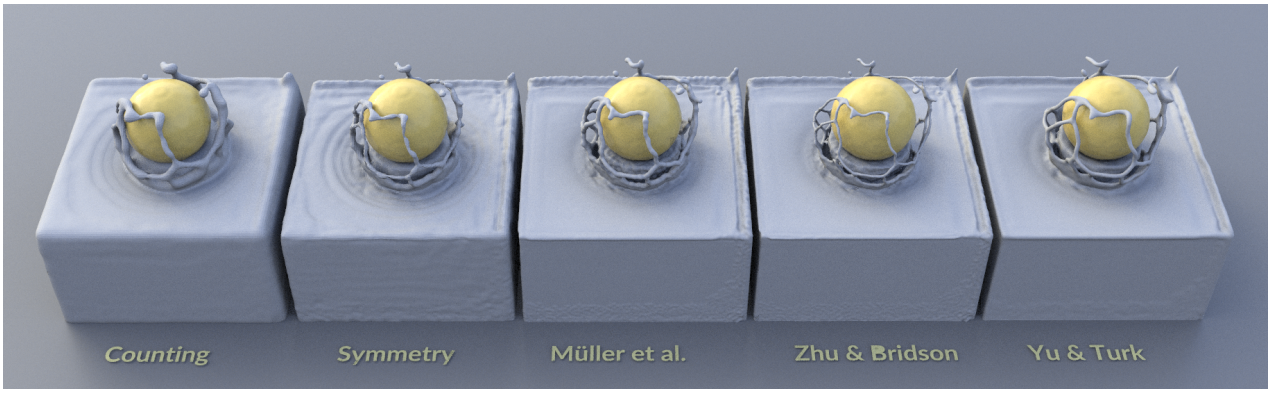



Fig. 10: Surface reconstruction of a floating ball with 0.93M SPH particles in a grid with $256^3$ cells: from left to right, our DIF by counting, our DIF by symmetry, Müller et at. [8] , Zhu and Bridson [9], Yu and Turk [11].

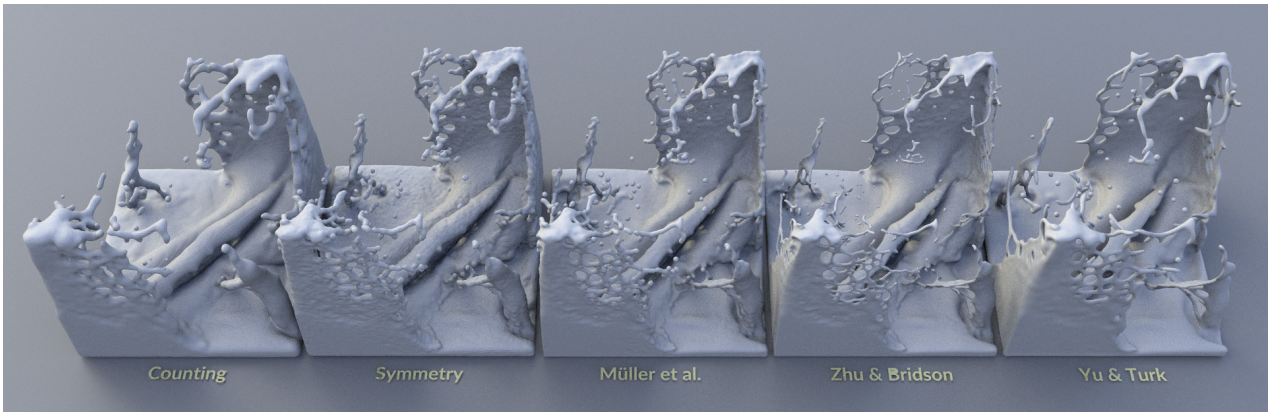



Fig. 11: Surface reconstruction of a double dam-break simulation with 1M SPH particles in a grid with $256^3$ cells: from left to right, our DIF by counting, our DIF by symmetry, Müller et at. [8] , Zhu and Bridson [9], Yu and Turk [11].

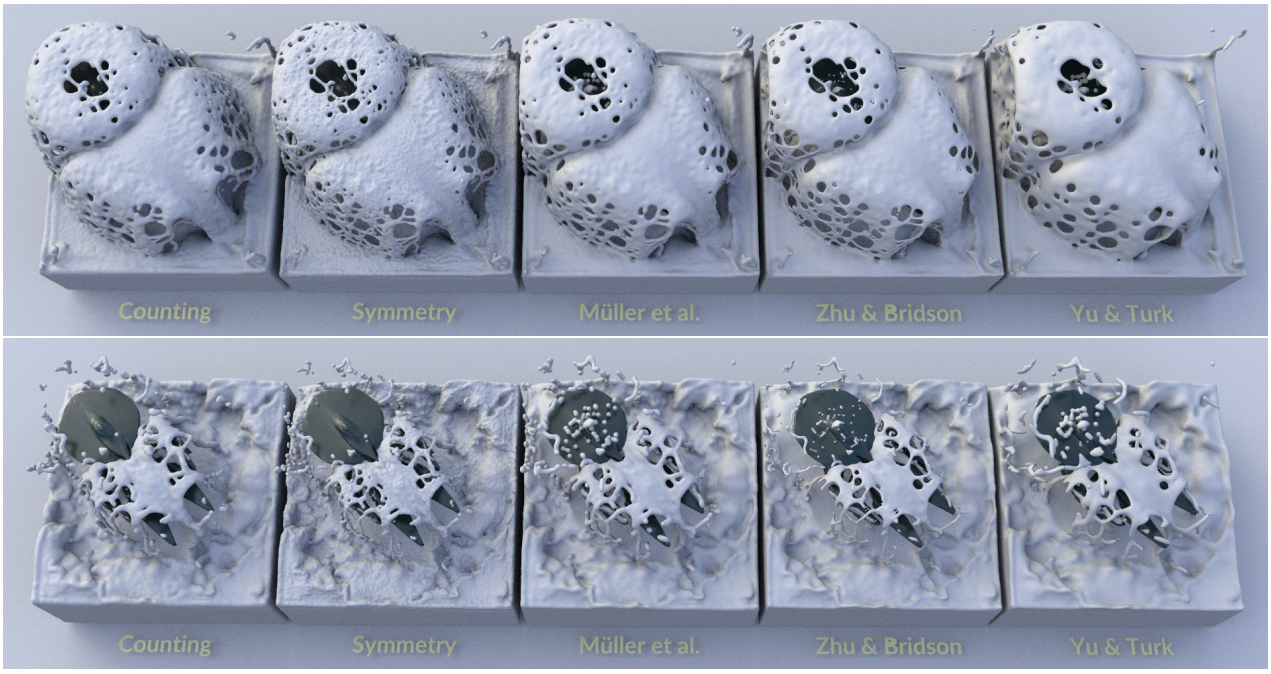
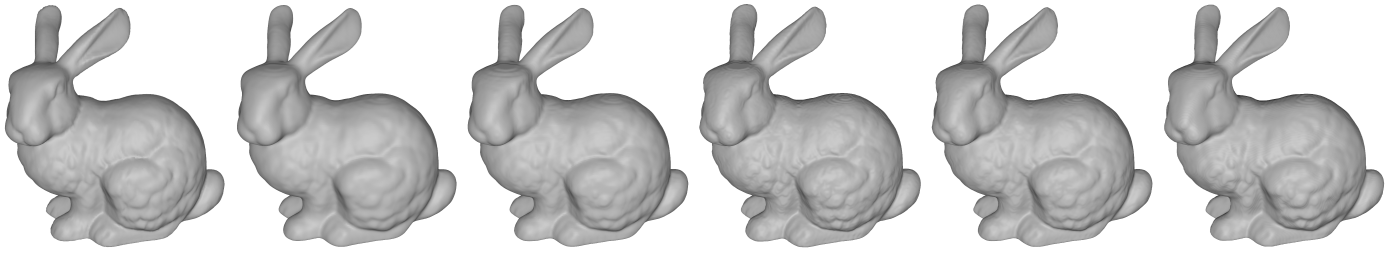



Fig. 12: Surface reconstruction of a spaceship rising from underwater with 4.76M SPH particles in a grid with $384^3$ cells: from left to right, our DIF by counting, our DIF by symmetry, Müller et at. [8] , Zhu and Bridson [9], Yu and Turk [11]. Our DIF by counting and symmetry are 2× and 1.8× faster than the best competitor method (Zhu and Bridson), 30× and 26× faster than the anisotropic method (Yu and Turk), respectively.

Table 1: Average statistics, reconstruction parameters and computational times (in seconds) per frame. The best computational performances and the second best are shown in bold and italic, respectively.

| | | | | | **time** (*std*) | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Experiment** | $\|\mathcal{P}\|$ | **res** | $c$ | $\sigma$ | Müller et al. [8] | Zhu and Bridson [9] | Yu and Turk [11] | DIF by counting | DIF by symmetry |
| Toy dragon (Fig. 9) | 1.20M | $256^3$ | 0.05 | 1.0 | 22.42 (0.59) | 15.94 (0.73) | 85.03 (2.15) | **7.30** (0.66) | *7.59* (0.18) |
| Floating ball (Fig. 10) | 0.93M | $256^3$ | 0.10 | 1.0 | 18.85 (0.50) | 16.23 (0.30) | 100.05 (0.29) | **7.21** (0.63) | *8.63* (0.21) |
| Double dam-break (Fig. 11) | 1.00M | $256^3$ | 0.10 | 1.0 | 19.48 (0.64) | 14.05 (1.79) | 76.27 (3.81) | **6.74** (0.56) | *6.81* (0.15) |
| Spaceship (Fig. 12) | 4.76M | $384^3$ | 0.05 | 0.8 | 129.78 (1.28) | 60.54 (3.17) | 899.84 (5.95) | **29.74** (0.68) | *34.16* (0.63) |



(a) Ground truth (b) *error* = $2.117 \times 10^{-2}$ (c) *error* = $2.105 \times 10^{-2}$ (d) *error* = $2.250 \times 10^{-2}$ (e) *error* = $2.129 \times 10^{-2}$ (f) *error* = $2.110 \times 10^{-2}$

Fig. 13: Reconstruction error analysis in the Stanford Bunny with 1.47M particles in a grid with $256^3$ cells: (a) ground truth, (b) DIF by counting, (c) DIF by symmetry, (d) Müller et al., (e) Zhu and Bridson, and (f) Yu and Turk.

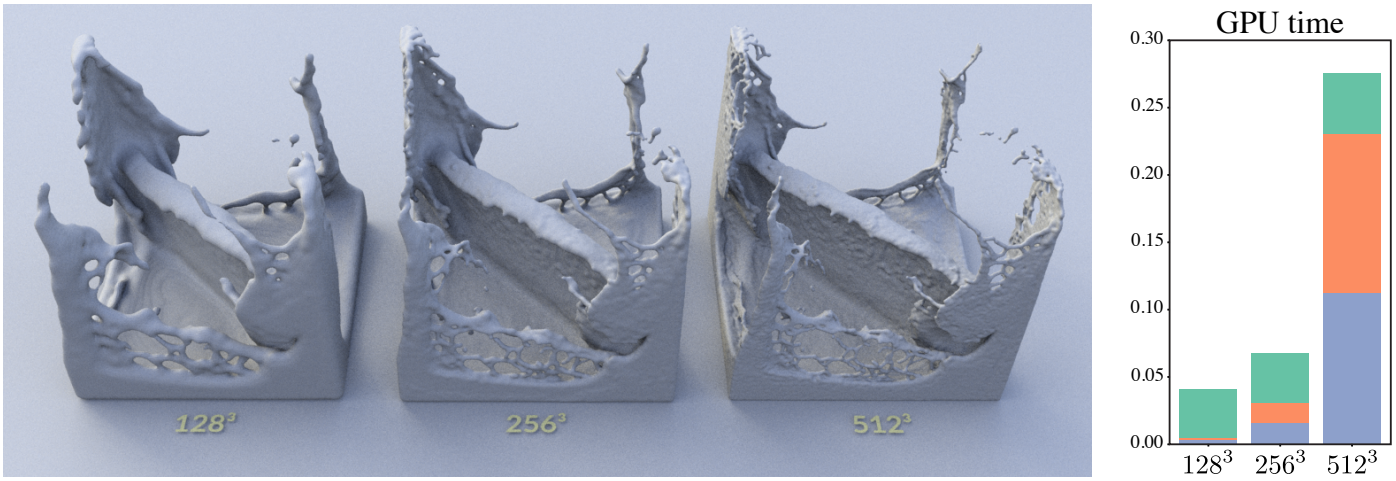



Fig. 14: Analysis of the grid resolution on GPU: surface reconstruction of the liquid splashing in the double dam-break (Fig. 11) using our DIF by counting with different grid resolutions (*left*) and the average computational timing (in seconds) of each pipeline stage (*right*): DIF evaluation (■), smoothing (■), and isosurfacing (■).

tion of $512^3$. Important to note that the GPU version is almost 100× faster than the single-core version for a grid resolution of $256^3$ (see Table 1).

### 6.2. DIF improvement

In order to recover fine and small-scale details in the liquid surface dissipated by the smoothing step, we can perform an additional step in our surface reconstruction pipeline depicted by Figure 3. Since the fine and small-scale details occur in regions where the smoothed DIF $\tilde{\vartheta}$ reaches low values at the grid cells $K$ of $\mathcal{G}$, we perform a thresholding segmentation of the original DIF $\vartheta$ using the isovalue $c$ as follows:

$$\overline{\vartheta}(K) = \begin{cases} \vartheta(K) & \tilde{\vartheta}(K) \leq c \\ 0 & \text{otherwise} \end{cases}.$$

Then, we apply the smoothing step (described in Section 4.2) in the segmented DIF $\overline{\vartheta}$ using a Gaussian blur with a smaller filter size of $\sigma/2$ to preserve the small details. Finally, the resulting surface $S$ is given by the union of the DIFs $\tilde{\vartheta}$ and $\overline{\vartheta}$, i.e., $S = (\tilde{\vartheta} \cup \overline{\vartheta})^{-1}(c)$ with

$$\tilde{\vartheta} \cup \overline{\vartheta}\,(K) = \max\{\tilde{\vartheta}(K), \overline{\vartheta}(K)\}\,.$$

Figure 15 shows the result of this improvement, our DIF-based method with this additional improvement is almost 1.6× faster than the best competitor method [9]. Note that our improvement recovers the surface details even for large isovalues and filter sizes.

This additional stage is ideal when the user wants to avoid fine-tuning parameters ($\sigma$ and $c$), mainly in surface reconstruction involving DIF by counting. However, the main drawback of this scheme is the computational time overhead due to the processing and smoothing of the segmented DIF. For this reason, we suggest this improvement as optional in our pipeline.

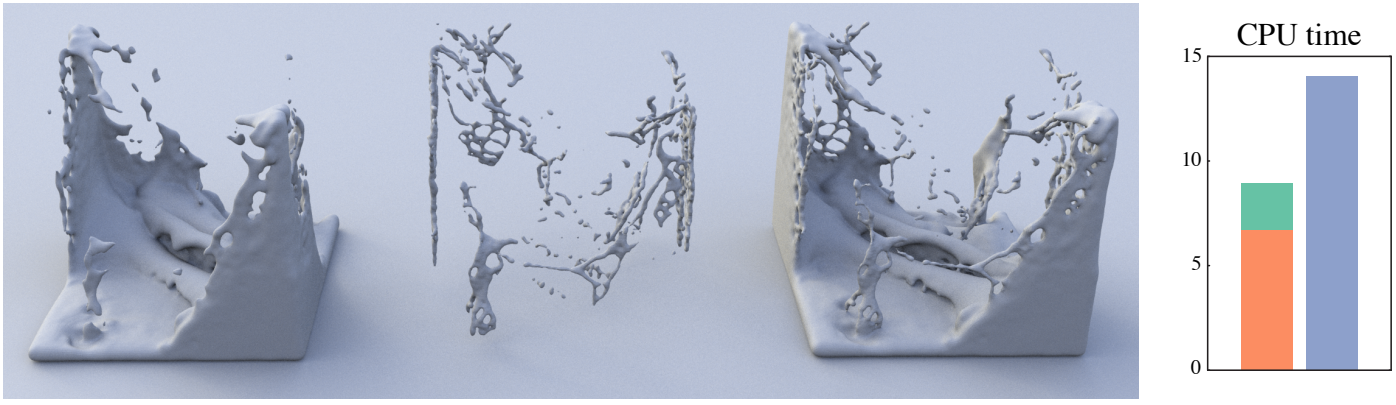



Fig. 15: Recovering surface details by DIF improvement: isosurface of the double dam-break in a grid with $256^3$ cells using DIF by counting with isovalue $c = 0.2$ and filter size $\sigma = 1.2$. From left to right, the surface from the smoothed DIF $\tilde{\vartheta}$, the surface details from the segmented DIF $\bar{\vartheta}$, and the resulting surface from $\tilde{\vartheta} \cup \bar{\vartheta}$. At rightmost, the average computational timings (in seconds) using a single-core CPU: our DIF by counting (■), the additional improvement process (■), and Zhu and Bridson method (■).

### *6.3. Limitations and future work*

The grid representation restricts our GPU implementation to bounded domains. It opens possibilities for replacing our current data structure with sparse grid representations using GVDB Voxels [29]. Aggressive smoothing can remove small surface details like liquid droplets. Thus, another direction of future research is constructing a "detail-aware" blur using adaptive filters [30] for DIF smoothing and more sophisticated polygonization algorithms suited for DIFs [5, 6] as well.

## 7. Conclusion

We introduced two simple and efficient surface reconstruction methods for liquid interfaces suited for particle-based fluid solvers on CPU and GPU architectures based on smoothed DIFs. The first DIF is a fast method defined by counting particles inside grid cells. The second DIF is achieved by analyzing the particle distribution inside grid cells, providing a high-quality surface. Our approach provides a significant speed-up for surface reconstruction compared to the prior methods, as attested by the set of experiments and comparisons in the paper.

## Acknowledgements

We want to thank the anonymous reviewers of SIBGRAPI 2022 for their valuable suggestions. We also thank Samantha Miller from SideFX for their kind donation of the Houdini software. This study was financed in part by the National Council for Scientific and Technological Development – Brazil (CNPq) under grant 309226/2020-1, and the São Paulo Research Foundation (FAPESP) under grant 2019/23215-9. The computational resources provided by the Center for Mathematical Sciences Applied to Industry (CeMEAI), also funded by FAPESP (grant 2013/07375-0).

## References

[1] Ihmsen, M, Orthmann, J, Solenthaler, B, Kolb, A, Teschner, M. SPH fluids in computer graphics. In: Eurographics 2014 - State of the Art Reports. 2014, p. 21–42. doi:10.2312/egst.20141034.

[2] Macklin, M, Müller, M. Position based fluids. ACM Trans Graph 2013;32(4). doi:10.1145/2461912.2461984.

[3] Lorensen, WE, Cline, HE. Marching cubes: A high resolution 3d surface construction algorithm. In: SIGGRAPH '87. 1987, p. 163–169. doi:10.1145/37402.37422.

[4] Hirt, CW, Nichols, BD. Volume of fluid (VOF) method for the dynamics of free boundaries. J Comput Phys 1981;39(1):201–225. doi:https://doi.org/10.1016/0021-9991(81)90145-5.

[5] Manson, J, Smith, J, Schaefer, S. Contouring discrete indicator functions. Comput Graph Forum 2011;30(2):385–393. doi:https://doi.org/10.1111/j.1467-8659.2011.01869.x.

[6] Evrard, F, Denner, F, van Wachem, B. Surface reconstruction from discrete indicator functions. IEEE Trans Vis Comput Graph 2019;25(3):1629–1635. doi:10.1109/tvcg.2018.2809751.

[7] Quispe, FI, Paiva, A. Counting Particles: a simple and fast surface reconstruction method for particle-based fluids. In: 2022 35th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI). 2022, p. 145–149. doi:10.1109/SIBGRAPI55357.2022.9991770.

[8] Müller, M, Charypar, D, Gross, M. Particle-based fluid simulation for interactive applications. In: Symposium on Computer Animation (SCA '03). 2003, p. 154–159.

[9] Zhu, Y, Bridson, R. Animating sand as a fluid. ACM Trans Graph 2005;24(3):965–972. doi:10.1145/1073204.1073298.

[10] Adams, B, Pauly, M, Keiser, R, Guibas, LJ. Adaptively sampled particle fluids. ACM Trans Graph 2007;26(3). doi:10.1145/1276377.1276437.

[11] Yu, J, Turk, G. Reconstructing surfaces of particle-based fluids using anisotropic kernels. ACM Trans Graph 2013;32(1):5:1–5:12. doi:10.1145/2421636.2421641.

[12] Akinci, G, Ihmsen, M, Akinci, N, Teschner, M. Parallel surface reconstruction for particle-based fluids. Comput Graph Forum 2012;31(6):1797–1809. doi:10.1111/j.1467-8659.2012.02096.x.

[13] Yang, W, Gao, C. A completely parallel surface reconstruction method for particle-based fluids. Vis Comput 2020;36:2313–2325. doi:10.1007/s00371-020-01898-2.

[14] Chen, Q, Zhang, S, Zheng, Y. Parallel realistic visualization of particle-based fluid. Comput Animat Virt W 2021;32(3–4):e2019. doi:https://doi.org/10.1002/cav.2019.

[15] Bhattacharya, H, Gao, Y, Bargteil, AW. A level-set method for skinning animated particle data. IEEE Trans Vis Comput Graph 2015;21(3):315–327. doi:10.1109/tvcg.2014.2362546.

[16] Sandim, M, Cedrim, D, Nonato, LG, Pagliosa, P, Paiva, A. Boundary detection in particle-based fluids. Comput Graph Forum 2016;35(2):215–224. doi:10.1111/cgf.12824.

[17] Kazhdan, M, Hoppe, H. Screened Poisson surface reconstruction. ACM Trans Graph 2013;32(3):29:1–29:13. doi:10.1145/2487228.2487237.

[18] van der Laan, WJ, Green, S, Sainz, M. Screen space fluid rendering with curvature flow. In: Symposium on Interactive 3D Graphics and Games (I3D '09). 2009, p. 91–98. doi:10.1145/1507149.1507164.

[19] Green, S. Screen space fluid rendering for games. http://developer.download.nvidia.com/presentations/2010/gdc/Direct3D_Effects.pdf; 2010. Game Developers Conference.
[20] Imai, T, Kanamori, Y, Mitani, J. Real-time screen-space liquid rendering with complex refractions. Comput Animat Virtual Worlds 2016;27(3–4):425–434. doi:10.1002/cav.1707.
[21] Neto, LSR, Apolinário Jr., AL. Real-time screen space cartoon water rendering with the iterative separated bilateral filter. Journal on Interactive Systems 2017;8(1). doi:10.5753/jis.2017.672.
[22] Truong, N, Yuksel, C. A narrow-range filter for screen-space fluid rendering. ACM Comput Graph Interact Tech 2018;1(1). doi:10.1145/3203201.
[23] Oliveira, F, Paiva, A. Narrow-band screen-space fluid rendering. Comput Graph Forum 2022;41(6):82–93. doi:https://doi.org/10.1111/cgf.14510.
[24] Toriwaki, J, Yoshida, H. Fundamentals of Three-Dimensional Digital Image Processing. Springer; 2009.
[25] Humphrey, W, Dalke, A, Schulten, K. VMD – Visual Molecular Dynamics. J Mol Graph 1996;14:33–38. URL: https://www.ks.uiuc.edu/Research/vmd/. doi:10.1016/0263-7855(96)00018-5.
[26] Domínguez, JM, Fourtakas, G, Altomare, C, Canelas, RB, Tafuni, A, García-Feal, O, et al. DualSPHysics: from fluid dynamics to multiphysics problems. Comp Part Mech 2021;doi:10.1007/s40571-021-00404-2.
[27] Kim, D. Fluid Engine Development. CRC Press; 2016.
[28] Taha, AA, Hanbury, A. An efficient algorithm for calculating the exact Hausdorff distance. IEEE Trans Pattern Anal Mach Intell 2015;37(11):2153–2163. URL: https://doi.org/10.1109/tpami.2015.2408351. doi:10.1109/tpami.2015.2408351.
[29] Wu, K, Truong, N, Yuksel, C, Hoetzlein, R. Fast fluid simulations with sparse volumes on the gpu. Comput Graph Forum 2018;37(2):157–167. doi:https://doi.org/10.1111/cgf.13350.
[30] Westin, CF, Kikinis, R, Knutsson, H. Adaptive image filtering. In: Handbook of medical imaging. Academic press; 2000, p. 19–31.