Counting Particles: a simple and fast surface reconstruction method for particle-based fluids

Filomen Incahuanaco Quispe
ICMC-USP
fincahuanaco@usp.br

Afonso Paiva
ICMC-USP
apneto@icm.usp.br

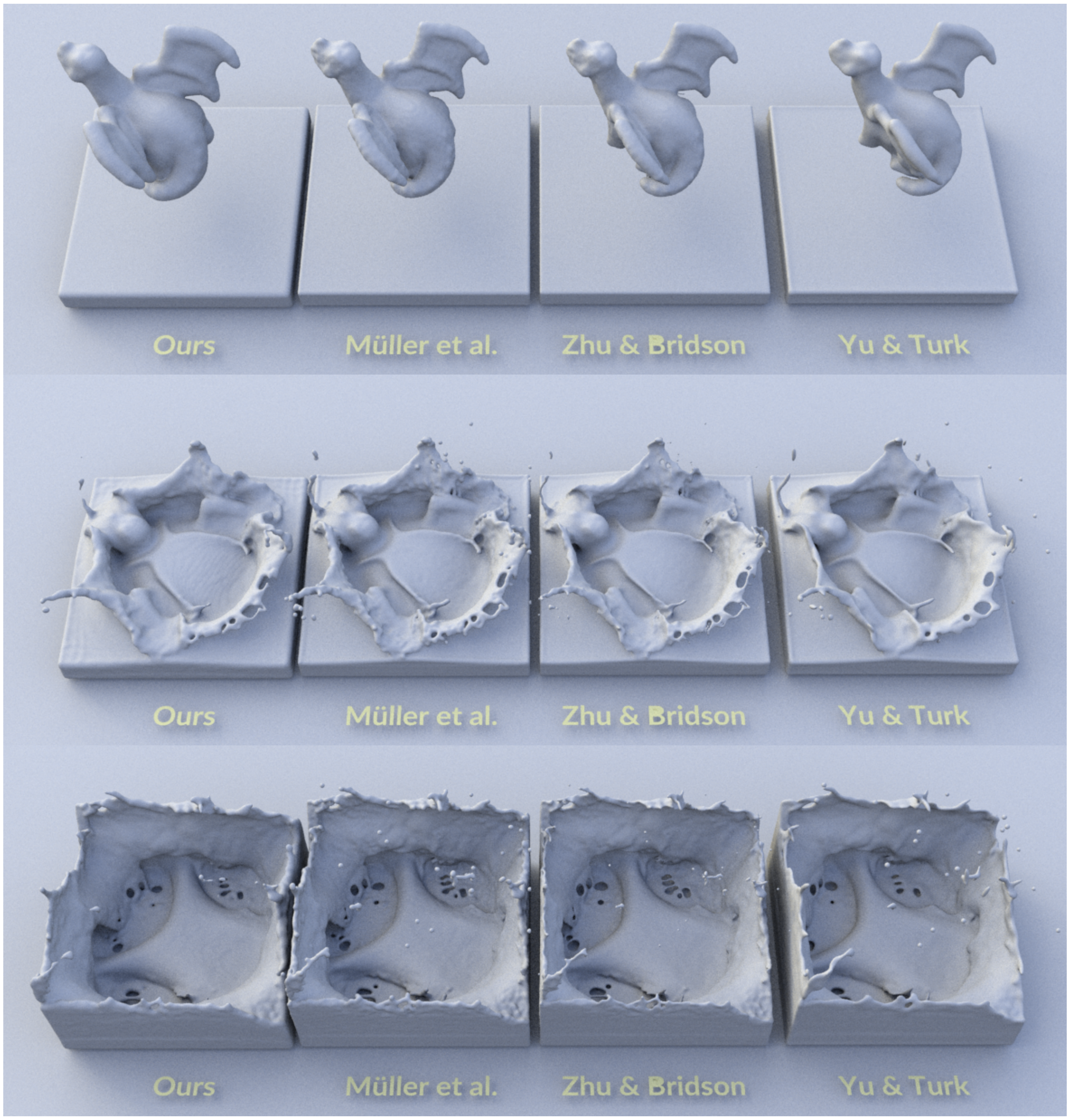


Fig. 1: Surface reconstruction of a liquid toy dragon with 1.2M particles in a grid with 256^3 cells: from left to right, our method, Müller et at. [1] , Zhu and Bridson [2], Yu and Turk [3]. Our method is 2.2× faster than the best competitor method.

***Abstract*—We present a novel and efficient surface reconstruction for particle-based fluid methods. Although particle-based methods are practical for computing level-sets that represent liquid interfaces, these methods are computationally expensive when the number of particles increases considerably due to the intense usage of particle approximations. This paper introduces a simple level-set approximation using a discrete indicator function (DIF) defined by counting particles inside grid cells. Our method is fast, easy to code, and can be adapted straightforwardly in particle-based solvers, even implemented in GPU. Moreover, we show the effectiveness of our approach through a set of experiments against prior surface reconstruction methods.**

I. Introduction

In particle-based fluid methods, such as Smoothed Particle Hydrodynamics (SPH) [4] and Position-Based Dynamics (PBD) [5], 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) [6], 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 [7]. Let $\mathcal{G}$ be a regular grid of resolution $n_x \times n_y \times n_z$ grid 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 [8], [9].

This paper presents a novel and practical surface reconstruction for particle-based fluid methods. We approximate efficiently a DIF by simply counting 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 method in action.

In summary, the contributions of our method are:

- a novel DIF that uses only the number of particles inside the grid cells;
- our framework speeds up considerably the surface reconstruction compared with prior methods;
- our method is simple and easy to code, even in GPU.

II. 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 signed distance field computed from a weighted sum of kernel evaluations from the particles' distances. These methods can use isotropic kernels [1], [2], adaptive size kernels [10] or anisotropic SPH kernels [3]. Despite the existence of parallel implementations of these methods [11]–[13], 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. [14] improved the undesired blobby appearance of the level-sets using a smoothing process by solving a constrained optimization problem. Sandim et al. [15] 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 [16]. 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 [17]–[21]. 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 [22] 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).

III. The method

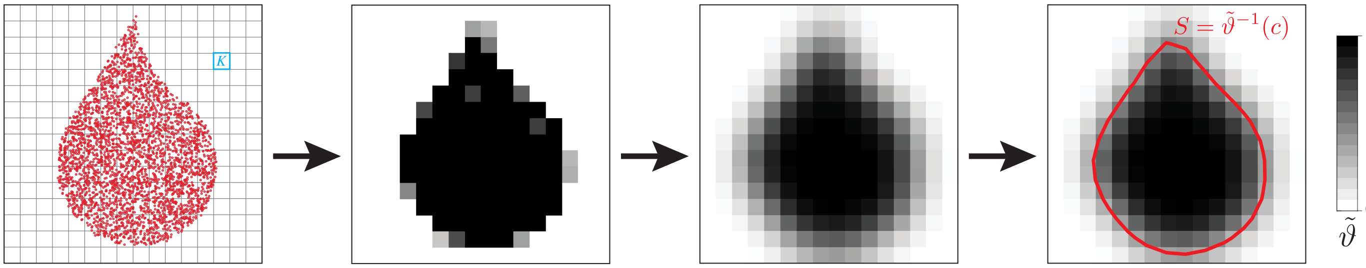
In this section, we explain the pipeline of the proposed surface reconstruction method. Given an input particle system $\mathcal{P}^t$ at time-step t, our method performs three main steps, as illustrated by Figure 2.

DIF evaluation. In this step, 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 μ_0 (at time-step $t = 0$) given by:

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

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. Assuming that the cell volume V_K is entirely occupied by the volume of μ_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:

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



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

Fig. 2: Surface reconstruction pipeline.

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

$$\boxed{\tilde{\vartheta}(K) = \frac{\min(N_K^t, \mu_0)}{\mu_0}}\,. \tag{2}$$

Smoothing. To reduce the DIF discontinuities around the liquid interface, we smooth the field (2) by applying 3D blur filters from image processing [23], e.g., box and Gaussian filters. Firstly, we apply a box filter of size $3 \times 3 \times 3$ to eliminate small holes (i.e., empty cells surrounded by full cells). Then, we apply Gaussian filter of size $5 \times 5 \times 5$ with standard deviation $\sigma = 1.2$ in a single pass to enhance the DIF.

Isosurfacing. In our method, given a 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. The liquid surface is represented by the level-set $S = \tilde{\vartheta}^{-1}(c)$ with isovalue $c \in (0,1)$. Thus, once computed the field $\tilde{\vartheta}$, we have fractional volumes of fluid $\tilde{\vartheta}_i$ located at the centers $\mathbf{p}_i$. Then, we extract the polygonal mesh of the level-set S executing the MC algorithm in the *dual grid*. The cells of the dual grid are obtained by connecting the centers of the adjacent cells of $\mathcal{G}$ (see Figure 3). The MC's lookup table determines the local topology of S inside each dual cell by indexing the configurations of $\mathrm{sign}(\tilde{\vartheta}_i - c)$ at the eight corners of the cell. Considering a linear approximation of $\tilde{\vartheta}$, given a dual edge $e_{ij} = (\mathbf{p}_i, \mathbf{p}_j)$ where $(\tilde{\vartheta}_i - c) \cdot (\tilde{\vartheta}_j - c) < 0$, the intersection point (vertex) $\mathbf{p}$ of S with e_{ij} is computed as follows:

$$\mathbf{p} = (1-\alpha)\,\mathbf{p}_i + \alpha\,\mathbf{p}_j \quad \text{with} \quad \alpha = \frac{c - \tilde{\vartheta}_i}{\tilde{\vartheta}_j - \tilde{\vartheta}_i}\,.$$

After processing each dual cell, the entire surface S is extracted. In our experiments, we use the isovalue $c = 0.25$.

IV. 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 [24]. All results have been achieved using a computer with AMD Ryzen 9 3950X and 32GB RAM and NVIDIA GeForce RTX 2070 with 8GB VRAM.

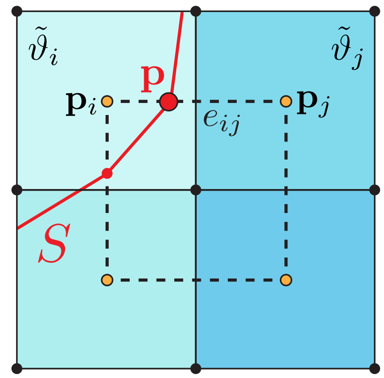


Fig. 3: Isosurfacing of a DIF in 2D: a dual grid is obtained from the grid $\mathcal{G}$ (solid black lines). For each dual cell (dashed black line), MC examines the configuration of $\mathrm{sign}(\tilde{\vartheta}_i - c)$ at the corners $\mathbf{p}_i$ (orange dots) to define the local topology of the surface S (solid red line) and determines the intersection points $\mathbf{p}$ (red dots) between S and the dual edges e_{ij}.

Figures 1, 4, and 5 show comparisons of our surface reconstruction method applied in different simulations in comparison with previous methods proposed by Müller et al. [1], Zhu and Bridson [2], and Yu and Turk [3]. Furthermore, the implementations in C++ of these methods can be found in the Github[1] repository from Kim's book [25].

Table I 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 and the column **res** is the resolution of $\mathcal{G}$. The average **time** of each method across all animation frames was measured using a single-core CPU. The *speedup* (in parenthesis) shows how fast our approach is compared to prior methods. Note that our approach is $2.1\times$ faster in the worst case and $13.9\times$ faster in the best case, demonstrating our method's efficiency.

V. Discussion

Scalability and profiling. Figure 6 shows the computational timing of our approach implemented on GPU and the performance profiling of each stage of the reconstruction pipeline (as shown in Figure 2) with different resolutions of $\mathcal{G}$. As can

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

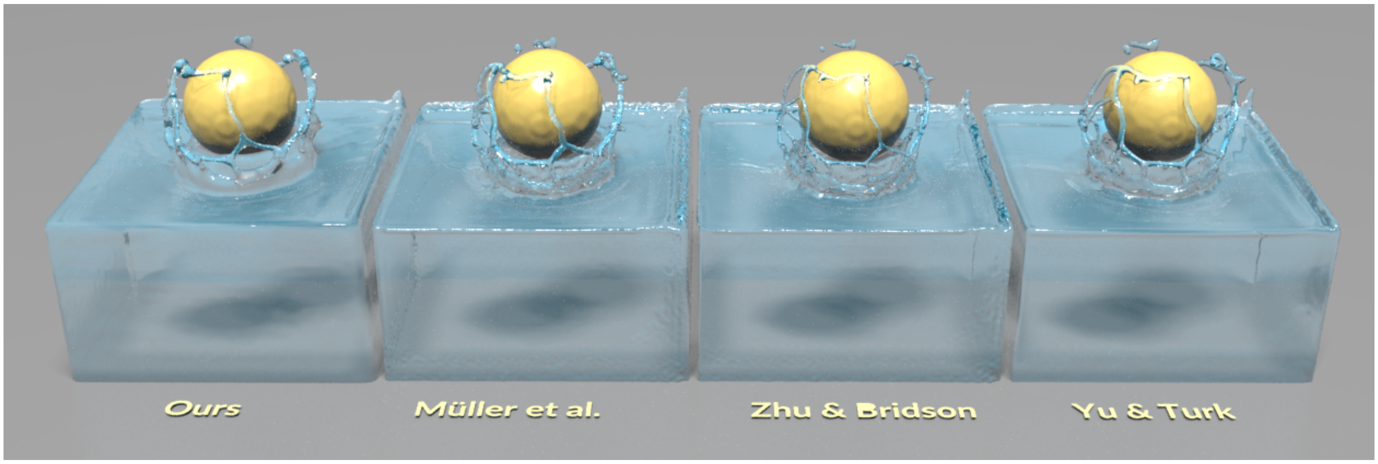


Fig. 4: Surface reconstruction of a floating ball with 0.93M SPH particles in a grid with 256^3 cells: from left to right, our method, Müller et at. [1] , Zhu and Bridson [2], Yu and Turk [3].

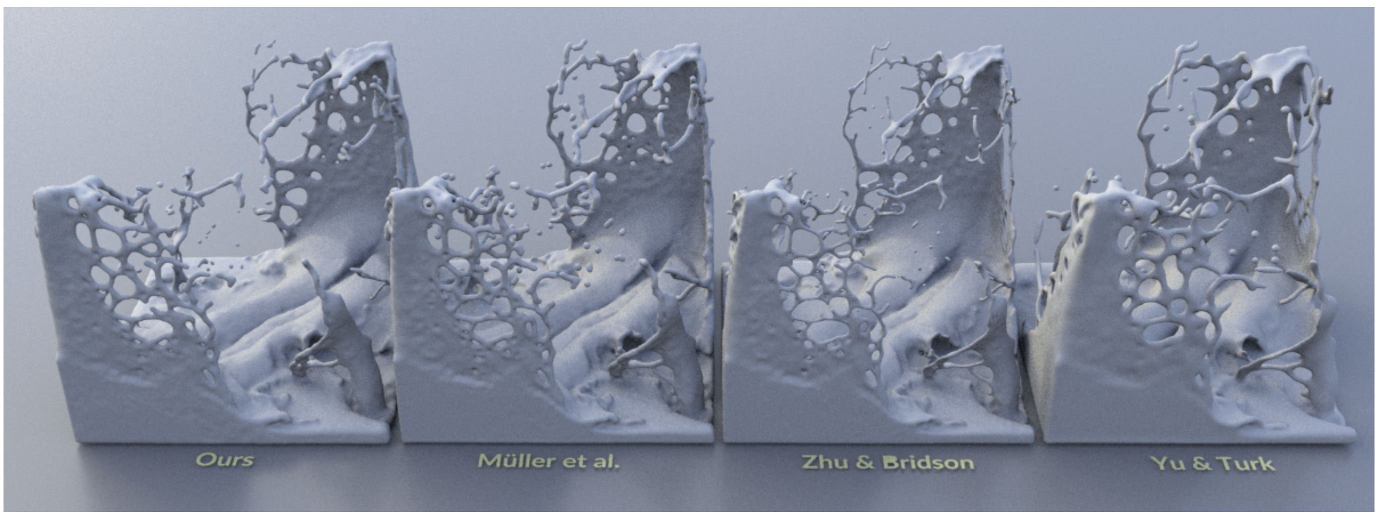


Fig. 5: Surface reconstruction of a double dam-break simulation with 1M SPH particles in a grid with 256^3 cells: from left to right, our method, Müller et at. [1] , Zhu and Bridson [2], Yu and Turk [3].

TABLE I: Average statistics and computational times (in seconds) per frame.

Experiment	$\|\mathcal{P}\|$	res	time (*speedup*) Müller et al. [1]	Zhu and Bridson [2]	Yu and Turk [3]	Ours
Toy dragon (Fig. 1)	1.20M	256^3	22.42 (*3.0*)	15.94 (*2.2*)	85.03 (*11.6*)	7.30 (–)
Floating ball (Fig. 4)	0.93M	256^3	18.85 (*2.6*)	16.23 (*2.3*)	100.05 (*13.9*)	7.21 (–)
Double dam-break (Fig. 5)	1.00M	256^3	19.48 (*2.9*)	14.05 (*2.1*)	76.27 (*11.3*)	6.74 (–)

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

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 [26]. 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 [27] for DIF smoothing and more sophisticated polygonization algorithms suited for DIFs [8], [9] as well.

VI. Conclusion

We introduced a simple and fast surface reconstruction method for liquid interfaces suited for particle-based fluid solvers on both CPU and GPU architectures. The proposed

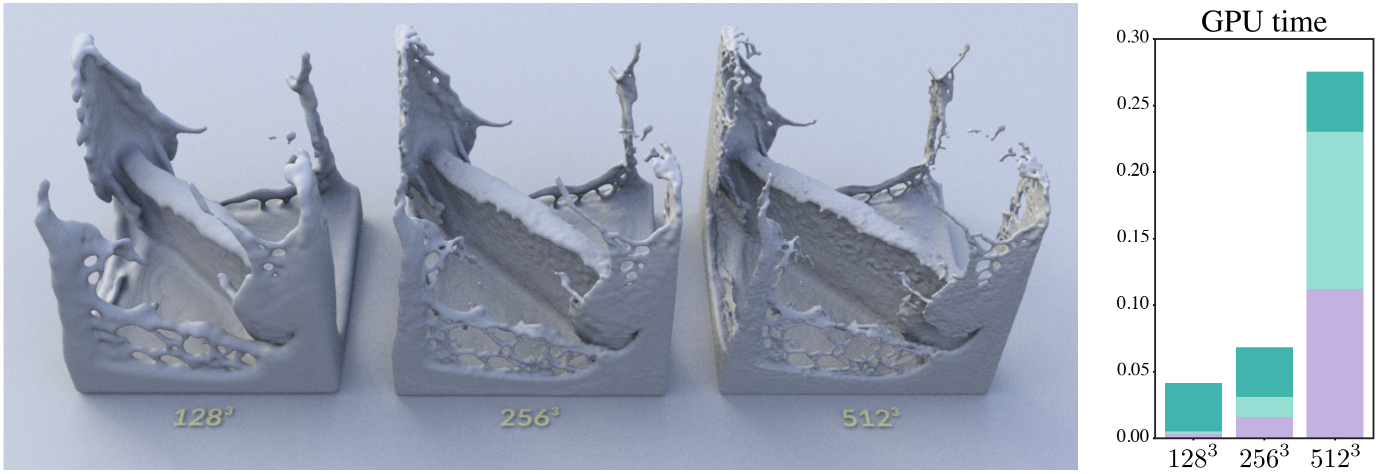


Fig. 6: Analysis of the grid resolution on GPU: surface reconstruction of the liquid splashing in the double dam-break (Figure 5) using our method with different grid resolutions (*left*) and the average computational timing (in seconds) of each pipeline stage (*right*): DIF evaluation (■), smoothing (■), and isosurfacing (■).

method relies on a novel smoothed DIF defined by counting particles inside grid cells, providing a high-quality surface. Our method provides a significant speed-up for surface reconstruction compared to the prior methods, as attested by the set of experiments and comparisons carried out in the paper.

Acknowledgements

We want to thank the anonymous reviewers 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] M. Müller, D. Charypar, and M. Gross, "Particle-based fluid simulation for interactive applications," in *Symposium on Computer Animation (SCA '03)*, 2003, pp. 154–159.

[2] Y. Zhu and R. Bridson, "Animating sand as a fluid," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 965–972, 2005.

[3] J. Yu and G. Turk, "Reconstructing surfaces of particle-based fluids using anisotropic kernels," *ACM Trans. Graph.*, vol. 32, no. 1, pp. 5:1–5:12, 2013.

[4] M. Ihmsen, J. Orthmann, B. Solenthaler, A. Kolb, and M. Teschner, "SPH fluids in computer graphics," in *Eurographics 2014 - State of the Art Reports*, 2014, pp. 21–42.

[5] M. Macklin and M. Müller, "Position based fluids," *ACM Trans. Graph.*, vol. 32, no. 4, 2013.

[6] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," in *SIGGRAPH '87*, 1987, pp. 163–169.

[7] C. W. Hirt and B. D. Nichols, "Volume of fluid (VOF) method for the dynamics of free boundaries," *J. Comput. Phys.*, vol. 39, no. 1, pp. 201–225, 1981.

[8] J. Manson, J. Smith, and S. Schaefer, "Contouring discrete indicator functions," *Comput. Graph. Forum*, vol. 30, no. 2, pp. 385–393, 2011.

[9] F. Evrard, F. Denner, and B. van Wachem, "Surface reconstruction from discrete indicator functions," *IEEE Trans. Vis. Comput. Graph.*, vol. 25, no. 3, pp. 1629–1635, 2019.

[10] B. Adams, M. Pauly, R. Keiser, and L. J. Guibas, "Adaptively sampled particle fluids," *ACM Trans. Graph.*, vol. 26, no. 3, 2007.

[11] G. Akinci, M. Ihmsen, N. Akinci, and M. Teschner, "Parallel surface reconstruction for particle-based fluids," *Comput. Graph. Forum*, vol. 31, no. 6, pp. 1797–1809, 2012.

[12] W. Yang and C. Gao, "A completely parallel surface reconstruction method for particle-based fluids," *Vis. Comput.*, vol. 36, pp. 2313–2325, 2020.

[13] Q. Chen, S. Zhang, and Y. Zheng, "Parallel realistic visualization of particle-based fluid," *Comput. Animat. Virt. W.*, vol. 32, no. 3–4, p. e2019, 2021.

[14] H. Bhattacharya, Y. Gao, and A. W. Bargteil, "A level-set method for skinning animated particle data," *IEEE Trans. Vis. Comput. Graph.*, vol. 21, no. 3, pp. 315–327, 2015.

[15] M. Sandim, D. Cedrim, L. G. Nonato, P. Pagliosa, and A. Paiva, "Boundary detection in particle-based fluids," *Comput. Graph. Forum*, vol. 35, no. 2, pp. 215–224, 2016.

[16] M. Kazhdan and H. Hoppe, "Screened Poisson surface reconstruction," *ACM Trans. Graph.*, vol. 32, no. 3, pp. 29:1–29:13, 2013.

[17] W. J. van der Laan, S. Green, and M. Sainz, "Screen space fluid rendering with curvature flow," in *Symposium on Interactive 3D Graphics and Games (I3D '09)*, 2009, pp. 91–98.

[18] S. Green, "Screen space fluid rendering for games," http://developer.download.nvidia.com/presentations/2010/gdc/Direct3D_Effects.pdf, 2010, game Developers Conference.

[19] T. Imai, Y. Kanamori, and J. Mitani, "Real-time screen-space liquid rendering with complex refractions," *Comput. Animat. Virtual Worlds*, vol. 27, no. 3–4, pp. 425–434, 2016.

[20] L. S. R. Neto and A. L. Apolinário Jr., "Real-time screen space cartoon water rendering with the iterative separated bilateral filter," *Journal on Interactive Systems*, vol. 8, no. 1, 2017.

[21] N. Truong and C. Yuksel, "A narrow-range filter for screen-space fluid rendering," *ACM Comput. Graph. Interact. Tech.*, vol. 1, no. 1, 2018.

[22] F. Oliveira and A. Paiva, "Narrow-band screen-space fluid rendering," *Comput. Graph. Forum*, 2022, to appear.

[23] J. Toriwaki and H. Yoshida, *Fundamentals of Three-Dimensional Digital Image Processing*. Springer, 2009.

[24] J. M. Domínguez, G. Fourtakas, C. Altomare, R. B. Canelas, A. Tafuni, O. García-Feal, I. Martínez-Estévez, A. Mokos, R. Vacondio, A. J. C. Crespo, B. D. Rogers, P. K. Stansby, and M. Gómez-Gesteira, "DualSPHysics: from fluid dynamics to multiphysics problems," *Comp. Part. Mech.*, 2021.

[25] D. Kim, *Fluid Engine Development*. CRC Press, 2016.

[26] K. Wu, N. Truong, C. Yuksel, and R. Hoetzlein, "Fast fluid simulations with sparse volumes on the gpu," *Comput. Graph. Forum*, vol. 37, no. 2, pp. 157–167, 2018.

[27] C.-F. Westin, R. Kikinis, and H. Knutsson, "Adaptive image filtering," in *Handbook of medical imaging*. Academic press, 2000, pp. 19–31.