The Visual Computer
https://doi.org/10.1007/s00371-020-01898-2

ORIGINAL ARTICLE

A completely parallel surface reconstruction method for particle-based fluids

Wencong Yang[1] · **Chengying Gao**[1]

Abstract
We present a novel surface reconstruction pipeline that significantly improves reconstructing efficiency while preserves high-quality details for particle-based liquid simulation. Our surface reconstruction algorithm is a sort of completely parallel narrow band method. At the beginning of reconstruction, we develop a spatial hashing grid-based strategy to identify surface particles, which is much more precise and simpler than the smoothed color field. Consequently, those precise surface particles ensure accurate extraction of scalar field in the narrow band around surface without any redundancy, which brings great performance improvement for subsequent reconstruction stages. Furthermore, in order to obtain a better computation performance, we carefully analyze the potential race conditions and conditional branches of each reconstruction step between parallel threads and come up with a completely parallel reconstruction method combined with the exclusive prefix sum algorithm. Our method is pretty straightforward to implement. Experimental results demonstrate that our method runs up to dozen times faster than the state-of-the-art of narrow band-based fluid surface reconstruction, especially for large-scale particle-based fluid.

Keywords Smoothed particle hydrodynamics · Fluid simulation · Surface reconstruction · Narrow band

1 Introduction

Particle-based fluid simulation methods have a wide range of applications in computer graphics due to its realistic visual effect and efficiency, including computer games, medical simulators and virtual reality applications. Over the years, smoothed particle hydrodynamics (SPH) have been gaining considerably increased interest for its simplicity and have been successfully applied to simulate a variety of complex fluid phenomena [21]. Nevertheless, it is an unavoidable and intractable challenge when it comes to efficiently reconstructing smooth and artifact-free surfaces from particles, especially for interactive applications.

At present, a popular and widely used reconstruction method is to establish a scalar field, and then the surface meshes are generated from the isosurface of that scalar field with Marching Cubes algorithm [18] or Marching Tiles algorithm [30]. In the whole reconstruction process, the construction of scalar field is the most time-consuming step. The resolution of scalar field grid prominently influences the quality of the reconstructed surface as well as the reconstruction time. Fine scalar grid is required so as to gain detailed surface and computation time grows cubically at the same time, which could be unacceptable for high-performance demand.

Noting that only a small part of the scalar field cells passes through the fluid surface, researchers proposed two different approaches to reduce computational complexity, which are adaptive grid scheme and narrow band scheme. Octree is a common data structure used in adaptive grid scheme [13,15,19,27]. In the framework of octree, surface regions are further subdivided to catch finer details. However, the expensive construction cost of octree is nonnegligible, and it's not easy to implement on GPU for parallelization. Narrow band methods work with uniform grid and only concentrate on a thin layer around surface, which makes the computational complexity and the memory consumption scale with the fluid surface instead of the simulation volume [4,31]. Generally speaking, narrow band approach is more friendly to parallelization. It is worth noting that in the narrow band scheme, accurate surface cell extraction is crucial. If it is not done well, some non-surface cells will be mistaken for surface

✉ Chengying Gao
mcsgcy@mail.sysu.edu.cn

Wencong Yang
yangwc3@mail2.sysu.edu.cn

[1] School of Data Science and Computer Science, Sun Yat-Sen University, Guangzhou, China

Published online: 06 July 2020

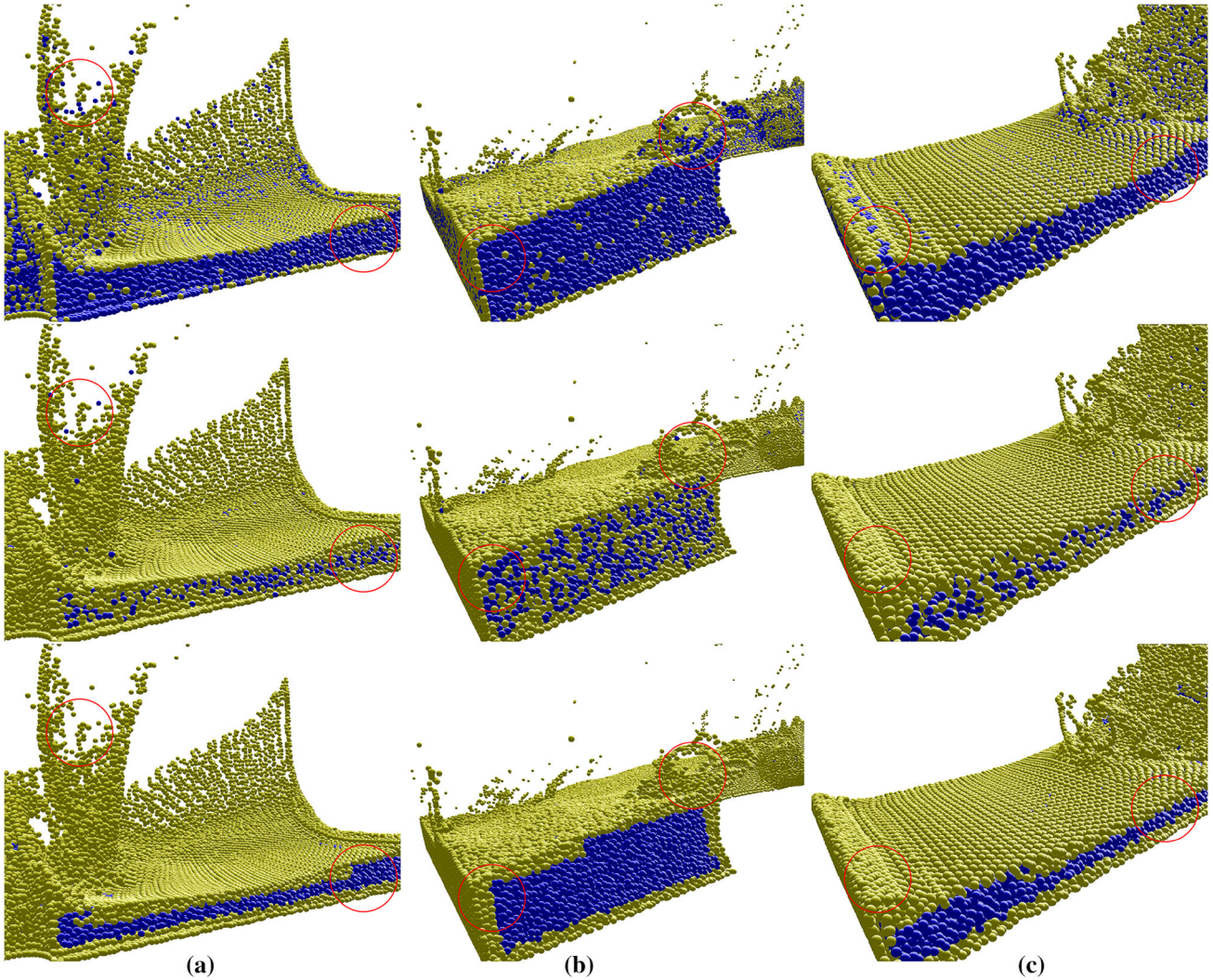


Fig. 1 Experimental comparisons between the smoothed color field and our method. Column **a**–**c** are three different fluid scenes simulated by PCISPH solver. For each column, the first two pictures display the results of the smoothed color field with high threshold and low threshold, respectively. The third picture of each column shows the accurate result of our approach. The yellow particles are surface particles while the blue particles are internal particles

cells, and even worse, some surface regions may be missed, resulting in holes in the reconstructed meshes as shown in Fig. 1.

Our contribution In this paper, a completely parallel surface reconstruction pipeline that could greatly improve computation efficiency without sacrificing the quality of reconstructed meshes is proposed. Every step of the pipeline is specifically designed to implement in parallel architecture. The main contributions of this paper are described as follows:

1. In order to extract narrow band regions around surface more accurately, we develop a spatial hashing grid-based strategy to identify surface particles more precisely. Compared with the previous smoothed color field, our approach is easy to implement with less memory access and less computation. With these surface particles, surface scalar cells are extracted without any omission and redundancy, which maximizes the performance improvement of the narrow band method.
2. We design a robust, efficient and fully parallel surface reconstruction algorithm. Every step of the algorithm is implemented in parallel on GPU without any complex data structure by analyzing the potential race conditions and proposing an approach combined with parallel exclusive prefix sum to predict the memory address for each thread. Our algorithm successfully avoids the thread conflict and could make full use of GPU launching no idle threads at all.

We implemented our method and integrated it into a standard PCISPH solver [25]. Our experimental results show that our algorithm could achieve remarkable efficiency improvement.

2 Related work

In the field of surface reconstruction for fluids, we concentrate on efficiency of scalar field construction techniques incorporating with Marching Cubes. Over the years, researchers aimed at achieving an optimal balance between surface quality and efficiency. For surface quality, the definition of the implicit function of the isosurface is of vital importance. Blinn came up with the earliest methods by introducing blobbies [6]. The disadvantage of this method is that it lacks of ability of generating flat surfaces, especially for fluid particles with sharp features. Afterwards, Müller et al. proposed a novel algorithm that the scalar field is calculated by weighting density information of neighbor particles to ameliorate bumpiness of classical blobbies [21]. However, the bumpy appearance is still obviously visible. Later, a signed distance field computation technique by weighting average values of neighbor particle positions was proposed by Zhu and Bridson [34], where the main downside is that it suffers from artifacts in concave regions. Solenthaler et al. corrected the artifacts on-the-fly by considering the movement of the neighbor particles' center of mass [26]. Similar to Solenthaler et al, Onderik et al. define fluid surface as the modified point-to-center distance-based implicit function [24] which nicely overcomes problems with clustered particles, though it still remains artifacts near isolated particles. Adams et al. also presented a distance-based surface tracking technique [1] to address the issues of the method of Zhu and Bridson. Though this technique is capable of reconstructing smooth surfaces better, the shortcoming comes at high computational complexity, which makes it more appropriate for adaptively sampled particle sets. Bhattacharya et al. cast the reconstruction problem in terms of constrained optimization and solved the optimization using a level-set approach for particle skinning-generating surfaces from animated particle data [5]. At present, the algorithm of anisotropic kernels proposed by Yu and Turk can obtain the smoothest surface at the expense of complex implementation and computation where the isotropic kernels are stretched or shrank along the associated directions of the density distribution in the particle neighborhood [32]. And then Wang et al. extended this anisotropy method to the surface reconstruction of multiphase fluids simulation [28].

Surface reconstruction is a critical step between fluid simulation and rendering, which is closely related to the subsequent rendering quality. However, it usually takes non-negligible time to obtain smooth enough surfaces where up to 90% of the computation time is spent on constructing the scalar field depending on the cell size and the size of influence region. Currently, researchers have carried out research on the effective reconstruction method of fluid surface, and put forward some efficient algorithms [2,4,31,33]. Müller et al. pointed out that identifying the surface cells and then only performing Marching Cubes over these cells could be a great approach to construct surface quickly [21]. However, they just mentioned this scheme by the way without a specific and feasible parallel solution. Later, to solve the efficiency problem of traditional octree, Bridson and Fedkiw presented a sparse block grids structure where the large grid is divided into blocks and only blocks around the narrow band region exist [7]. Houston et al. designed a novel RLE sparse level sets and the regions are encoded by run-length encoding with respect to distance to the narrow band [12]. Nevertheless, it's not suitable to be implemented on GPU. And then Zhou et al. proposed a parallel surface reconstruction algorithm where a parallel octrees approach is employed [33]. Nielsen and Museth introduced a dynamic tubular grid (DT-grid) structure, where the narrow band is not established on regular grid or a tree without the limitation of any computational box [22]. Afterward, in order to handle very high grid resolutions efficiently, Nielsen et al. put forward a new method using out-of-core techniques together with compression strategies [23]. Canezin et al. introduced a better neighborhood computation where the local fluid topology around each particle is tracked using a graph structure for the correction of blending artifacts in the reconstructed fluid surface, and hence the graph connectivity needs to be taken into account carefully [8]. Noting that not so much work has been devoted to parallelized surface reconstruction, Akinci et al. began to work on parallelization of reconstruction focusing on the topic of narrow band method [4]. They presented a parallel surface reconstruction scheme for particle-based fluids that achieves considerable speedup compared to other serial implementations. However, the pipeline is not completely parallelized so as to avoid potential race conditions. Later, Akinci et al. proposed an adaptive surface reconstruction method using 3-level uniform grids [3], and so a careful and thoughtful crack repair step need to be taken into account to obtain correct results. More recently, Wu et al. ameliorated the memory efficiency of the method of Akinci et al. using an efficient parallel cuckoo hashing while there still remains serial part in their implementation [31].

3 Surface particle identification

The narrow band surface reconstruction algorithm only deals with those grid vertices near the fluid surface. Other grid vertices in the internal and external regions of the fluids have no contribution to surface meshes, so they can be directly

discarded to avoid useless computation to greatly speed up the reconstruction process. Therefore, the key to the narrow band-based surface reconstruction algorithm is how to select the grid vertices near the fluid surface efficiently and accurately.

At present, the narrow band-based reconstruction method for particle-based fluids is mainly divided into four steps: surface particles identification, surface vertices identification, scalar field computation and triangulation. The surface particles are used for identifying surface vertices. The so-called surface particles are those fluid particles near the surface area of the fluids, and the surface vertices which are extracted with respect to their distance to surface particles are those scalar field grid vertices in the narrow band regions. Surface particles play a critical role in the narrow band method since they are closely related to the extraction accuracy of the narrow band regions of the fluids. If the surface particles identification is not accurate enough, the narrow band regions will cover those grid vertices which are actually far from the surface area in the internal and external regions of the fluids, causing the performance degradation. Moreover, some genuine surface vertices may be omitted leading to the generation of holes in the surface meshes, which is absolutely unexpected and unacceptable.

Akinci et al. employed the smoothed color field (SCF) to identify surface particles [21]. The definition of the SCF is described as follows

$$cs(\boldsymbol{x}) = \Sigma_j m_j \frac{1}{\rho_j} W(\boldsymbol{x} - \boldsymbol{x_j}, h) \tag{1}$$

where $\boldsymbol{x}$ is the particle position to be detected. m_j, ρ_j and $\boldsymbol{x_j}$ are the mass, density and position of the neighbor particles where the neighborhood is a spherical region with $\boldsymbol{x}$ as the center and h as the radius. W is the smoothing kernel function. Then, by judging whether the gradient length of the smoothed color field exceeds a given threshold l as shown in Eq. (2), we could identify those surface particles.

$$|\nabla cs(\boldsymbol{x})| > l \tag{2}$$

Nevertheless, extracting surface particles in this way is not robust enough, because the threshold needs to be chosen extraordinary carefully due to its high sensitivity to surface particles. Some non-surface particles will be labeled as surface particles if the threshold is too small, while some surface particles will fail to be caught if the threshold is too large. In addition, it suffers from detecting the isolated parts like splashes requiring some extra operations to handle it. Accordingly, we propose a simple, accurate and robust surface particles identification method with no need to calculate the gradient. Currently in the particle-based fluids, for the sake of rapid query of neighbor particles for a give point, a uniform spatial hashing grid structure is usually established and each fluid particle is mapped into one of the cells of the spatial grid according to particle's position. Our method will utilize this spatial grid to detect the surface particles precisely and quickly.

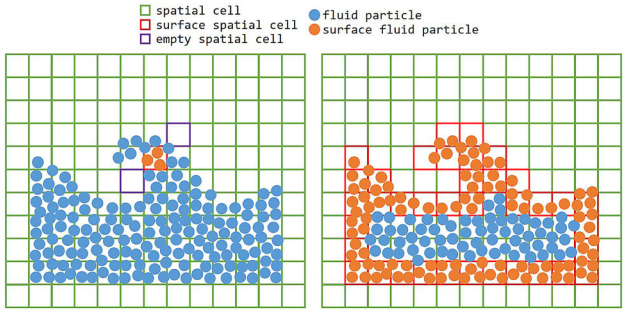


Fig. 2 Surface particles identification in two-dimensional case. The left picture shows one of the spatial cells detected as surface spatial cell, and so the particles in it are surface particles. The right picture displays all the surface particles identified with our approach

Compared with the scalar field grid, spatial hashing grid is coarser. Our approach starts with finding out those spatial grid cells around fluid surface named surface spatial cells. A spatial grid cell is surface spatial cell if and only if it's not empty and at least one adjacent cell is empty. Here, a spatial grid cell is empty means no fluid particle is mapped into that cell. There are eight adjacent cells in two-dimensional case, while there are twenty-six adjacent cells in three-dimensional case. As shown in the left picture of Fig. 2 taking 2D as example, the top right cell and the bottom left cell which are purple are empty without any particle in them, which signifies that the center red cell is supposed to be surface spatial cell. Therefore, the particles colorized as orange in this surface spatial cell are surface particles. By performing this judgment in parallel for all of cells of the spatial hashing grid, surface particles could finally be identified precisely forming a thin layer around the surface, which could be seen in the right picture of Fig. 2.

It could be pretty straightforward to implement our approach where just a simple query about whether there are particles in each spatial cell and its adjacent cells is need for each parallel thread instead of traversing all of the neighbor particles and calculating gradient. The pseudocode of our approach is given in Algorithm 1. Compared to the smoothed color field, our scheme benefits from less computation, less memory access, accurate identification of surface particles, simple implementation without extra consideration for isolated particles and convenience for no need to set a threshold. We implemented our method and the smoothed color field, respectively, and compared their difference for surface particles identification with several different fluid scenes shown in Fig. 1. Obviously, the smoothed color field suffers from leaving out some surface particles and misidentifying some

internal particles as surface particles because of its sensitivity to the gradient threshold causing inaccurate surface particles that would be used for subsequent steps unfortunately. Thus, a lower threshold l should be set to prevent omittance, causing many more internal false-positives as we can see in the second row of Fig. 1. In contrast, our approach could extract fewer but accurate surface particles (as can be seen from Table 1), which lays the foundation for the precise surface vertex identification.

Algorithm 1 Surface particle identification

```
threadId ← getThreadIdGlobal()
if spatialHashingGrid[threadId].isEmpty then
    return False
end if
neighborsIdArray = getAll26Neighbors(threadId)
isSurfaceCell ← False
for nId ∈ neighborsIdArray do
    if spatialHashingGrid[nId].isEmpty then
        isSurfaceCell ← True
        break
    end if
end for
if not isSurfaceCell then return False
end if
particlesIdArray = getAllParticlesOfCell(threadId)
for pId ∈ particlesIdArray do
    isSurfParticleArray[pId] ← True
end for
return True
```

4 Completely parallel pipeline

In order to generate smooth and realistic fluid surfaces, the scalar field grid is not supposed to be too coarse making the reconstruction step becomes even more time-consuming than simulation and rendering. In recent years, accelerating surface reconstruction by GPU becomes an increasingly common and popular approach in many fields of computer graphics [10,11,16,17,29,31]. Nevertheless, it's pretty challenging to design an efficient parallel fluid surface reconstruction algorithm due to the necessary consideration of avoiding race conditions and dealing with data dependence. The key point in solving this challenge is how to combine the narrow band method with the parallel architecture to obtain the maximum performance improvement.

Akinci et al. have proposed a narrow band-based parallel surface reconstruction algorithm and obtained remarkable speedup compared with previous serial algorithms [4]. However, their algorithm is not fully parallel and there exists serial parts owing to avoidance of race conditions. Noting this, we design a completely parallel reconstruction pipeline which could further improve the efficiency. All of the steps

Table 1 General information of the experimental scenes

Scene	#particles	$\frac{\#particles_{surf}}{\#particles_{total}}$		Cell size = r				Cell size = $r/2$			
		scf	Ours	$\frac{\#vertices_{surf}}{\#vertices_{total}}$		#triangles	grid res	$\frac{\#vertices_{surf}}{\#vertices_{total}}$		#triangles	Grid res
				Ak12	ours			Ak12	Ours		
DBD	0.2M	0.72	**0.38**	0.12	**0.10**	8.9k	$176 \times 120 \times 92$	0.095	**0.074**	0.36M	$351 \times 240 \times 184$
Bunny	0.9M	0.67	**0.31**	0.09	**0.08**	0.85M	$231 \times 342 \times 231$	0.058	**0.046**	2.73M	$462 \times 684 \times 462$
Drop	2.4M	0.45	**0.15**	0.14	**0.07**	0.37M	$227 \times 282 \times 227$	0.123	**0.051**	1.51M	$453 \times 564 \times 453$
Hemisphere	3.4M	0.43	**0.11**	0.08	**0.04**	0.38M	$375 \times 301 \times 227$	0.066	**0.028**	1.54M	$749 \times 601 \times 453$

The scf is the abbreviation of the smoothed color field
All the data are given average per frame

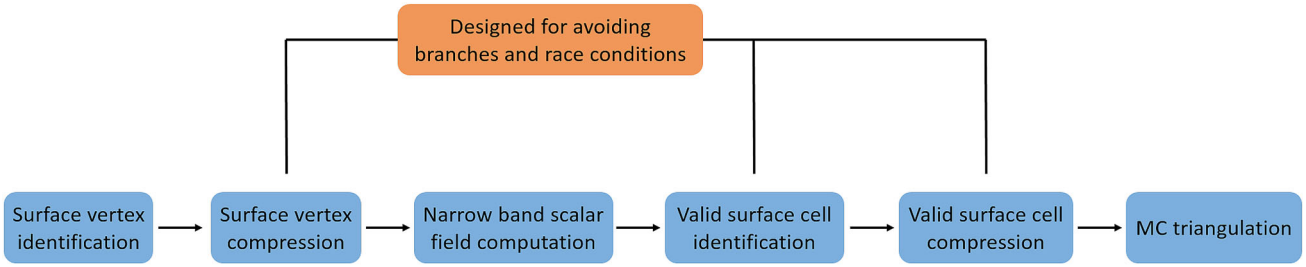


Fig. 3 Our completely parallel narrow band surface reconstruction pipeline

of our algorithm are designed elaborately and execute on GPU parallelly. Our algorithm is divided into six steps and they are surface vertex identification, surface vertex compression, narrow band scalar field computation, valid surface cell identification, valid surface cell compression and MC triangulation, which is shown in Fig. 3. Details of these steps are described in the following subsections.

4.1 Surface vertex identification and compression

The vertices of scalar field grid near to the fluid surface are called surface vertices where each surface vertex corresponds to a cell named surface cell. Surface particles obtained from previous step are utilized to identify surface vertices. For each surface particle $\boldsymbol{x}$, the grid vertices of scalar field covered by a bounding sphere with the center of the particle and the radius of $3r$ where r denotes the equilibrium distance of the fluid particles would be marked as surface vertices. At first, we allocate a tag array named *isSurVerArray* where all of the elements are initialized to 0. A GPU thread will be launched for each surface particle to execute surface vertex identification and if there is the grid vertex i covered by a bounding sphere of a surface particle, then we set the corresponding element $isSurVerArray[i]$ to 1 indicating that it is a surface vertex. As shown in Fig. 4 taking two-dimensional case as example, the purple dots are in the circular range with the surface particle as the center, and so these grid vertices are all surface vertices.

The size of *isSurVerArray* equals to the number of scalar field grid vertices. Different threads may try to write to the same address of the array simultaneously, but we don't need to handle it at all. When two threads detect the same MC vertex and want to set it to true, the vertex will be marked as surface vertex regardless of the write order. In the grid vertices of scalar field, surface vertices only account for a very small part. For the sake of efficiency, using conditional branches to make subsequent operations only execute on these surface vertices is not a thoughtful option leading to a large amount of idle threads, which is not friendly to GPU architecture. Hence, extracting those surface vertices and compacting them into a continuous array is tremendously

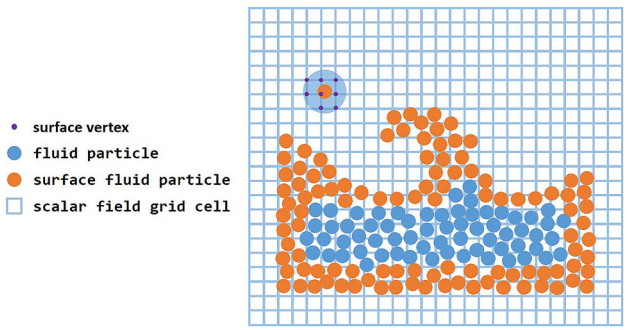


Fig. 4 Surface vertex identification in two-dimensional case. The grid vertices located in the circular extent of at least one surface particle are surface vertices colorized as purple

essential. In the algorithm of Akinci [4], the surface vertices are compressed in a serial manner due to possible race conditions. In this paper, we implement this step in a parallel manner by using the exclusive prefix sum which could predict the address in the continuous array *surVerIndexArray* of each surface vertex where *surVerIndexArray* stores indices of those surface vertices.

Given a binary operator $\bigoplus$ and an array $[a_0, a_1, \ldots, a_{n-1}]$ whose length is n, the inclusive prefix sum sequence of the array is $[a_0, (a_0 \bigoplus a_1), \ldots, (a_0 \bigoplus a_1 \bigoplus \cdots \bigoplus a_{n-1})]$ where the binary operator $\bigoplus$ is arithmetic addition here. Hence, the exclusive prefix sum sequence is obtained by shifting all the elements right one element and setting the first element to 0, and finally the sequence is $[0, a_0, (a_0 \bigoplus a_1), \ldots, (a_0 \bigoplus a_1 \bigoplus \cdots \bigoplus a_{n-1})]$. As shown in Fig. 5, we employ the parallel exclusive prefix sum algorithm to get the prefix sum array *isSurVerScanArray* which exactly stores the address in *surVerIndexArray* of surface vertices. Therefore, the index in *surVerIndexArray* for surface vertex i is $isSurVerScanArray[i]$. Accordingly, all the surface vertices could be successfully compacted into a continuous array in parallel in this way.

4.2 Narrow band scalar field computation

Now that surface vertices have been identified and compressed, only the calculation of the scalar field value of these

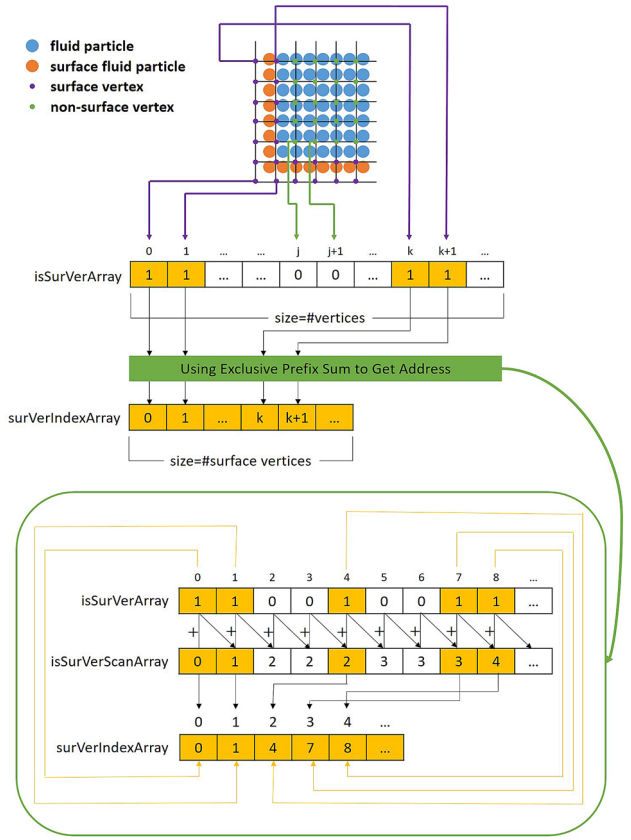


Fig. 5 Surface vertex compression in two-dimensional case. Extract the surface vertices and mark their corresponding *isSurVerArray* with 1. In order to reduce idle threads and conditional branches, exclusive prefix sum is utilized to compact those elements marked as 1 of *isSurVerArray* into a continuous array *surVerIndexArray*

surface vertices is required, which has a pivotal role in the quality of constructed meshes. Our reconstruction method could work well with all previously proposed scalar field construction algorithms. In this paper, we only implemented the method of Yu and Turk [32]. The approach of Solenthaler et al. [26] is ameliorated based on the method of Zhu and Bridson [34] yielding smooth results while being comparatively fast. However, in terms of high-quality scalar field construction, the approach of Yu and Turk [32] fully deserves to be the state of the art. Their scheme applies the weighted version of principal component analysis (WPCA) proposed by Koren and Carmel [14] to the neighborhood particle positions. It starts with constructing a weighted covariance matrix C_i and the weighted mean $\boldsymbol{x}_i^{\omega}$ which are formulated as

$$\boldsymbol{x}_i^{\omega} = \frac{\Sigma_j \omega_{ij} \boldsymbol{x}_j}{\Sigma_j \omega_{ij}} \tag{3}$$

$$C_i = \Sigma_j \omega_{ij} (\boldsymbol{x}_j - \boldsymbol{x}_i^{\omega})(\boldsymbol{x}_j - \boldsymbol{x}_i^{\omega})^T / \Sigma_j \omega_{ij} \tag{4}$$

where the function ω_{ij} is an isotropic weighting function with respect to particle i and j with support r_i. Then, an eigendecomposition is performed on matrix C_i, which obtains the eigenvectors and the eigenvalues of the matrix.

$$C = R \Sigma R^T \quad where \quad \Sigma = diag(\sigma_1, \ldots, \sigma_d) \tag{5}$$

Hence, an anisotropy matrix G_i is constructed to match the smoothing kernel W with the output of PCA.

$$G_i = \frac{1}{h_i} R \widetilde{\Sigma}^{-1} R^T \tag{6}$$

The final definition of surface scalar field is formulated as

$$\Phi(\boldsymbol{x}) = \Sigma_j \frac{m_j}{\rho_j} W(\boldsymbol{x} - \overline{\boldsymbol{x}}_j, G_j) \tag{7}$$

Prior to the construction of scalar field, a 3D variant of Laplacian smoothing is employed to achieve further smooth surface. The updated particle centers $\overline{\boldsymbol{x}}_i$ are calculated by

$$\overline{\boldsymbol{x}}_i = (1 - \lambda)\boldsymbol{x}_i + \lambda \Sigma_j \omega_{ij} \boldsymbol{x}_j / \Sigma_j \omega_{ij} \tag{8}$$

Nevertheless, a major problem with this anisotropic kernel is the computation which is comparatively not friendly to high-performance demand, where the matrix eigendecomposition becomes the most time-consuming part. To solve the eigendecomposition as efficient as possible, we employ and implement a fast algorithm that computes the SVD of 3×3 matrices with minimal branching and elementary floating point operations which was proposed by McAdams et al. [20] and parallelized by Gao et al [9].

Algorithm 2 gives some implementation details of scalar field computation. All the first level for loops are parallel for loops, which means the loop body executes in parallel on GPU. The function called on line 17, named $fastSVDAlgorithm()$, is a fast eigendecomposition implementation adopted from Gao et al [9]. The fast eigendecomposition method significantly improves our reconstruction efficiency. Here, $smoothingRadius$ is the smoothing radius in SPH simulation stage.

4.3 Valid surface cell identification and compression

Each surface vertex corresponds a grid cell of scalar field which is named surface cell. One last step before triangulation is to determine which surface cells could generate surface triangles and which would not as illustrated in Fig. 6. Noticing that not all surface cells could produce triangles for surface meshes, we try to detect those surface cells named valid surface cells which contribute at least one triangle. Apparently, no triangle will be generated when the scalar field values of the eight corner vertices of the surface cell are all positive or negative and we call this kind of surface cell invalid surface cell. Similar to surface vertices in total

Algorithm 2 Scalar field calculation

```
//diffusion smoothing
for i ∈ particlesIdArray do
    x̄[i] ← calculate using Equation(8)
    x^ω[i] ← calculate using Equation(3)
end for
//anisotropic matrices computation
for i ∈ particlesIdArray do
    wSum ← 0.0, x_i^ω ← x^ω[i]
    for j ∈ neighborhood particles of i do
        v ← x[j] − x_i^ω
        w_j ← 1.0 − (√(v · v)/(2 ∗ smoothingRadius))^3
        wSum ← wSum + w_j
        C[i] ← C[i] + w_j · (v · v^T)
    end for
    C[i] ← C[i]/wSum
    //u and w are 3 × 3 matrices while v is a vector
    (u, v, w) ← fastSVDAlgorithm(C[i])
    v ← |v|, ret ← u
    (ret[0][0], ret[1][0], ret[2][0])/ = v.x
    (ret[0][1], ret[1][1], ret[2][1])/ = v.y
    (ret[0][2], ret[1][2], ret[2][2])/ = v.z
    cof ← (v.x · v.y · v.z)^{1/3}/smoothingRadius
    svdMatricesArray[i] ← ret · w · cof
end for
//scalar field function computation
for i ∈ surVerIndexArray do
    vPos ← the position of the vertex i
    sum ← 0.0
    for j ∈ neighborhood particles of i do
        gMat ← svdMatricesArray[j]
        v ← gMat · (x̄[j] − vPos)
        sum ← sum + (315.0/64π) ∗ det(gMat) ∗ (1 − v · v)^3
    end for
    scalarFieldGrid[i] ← sum
end for
```

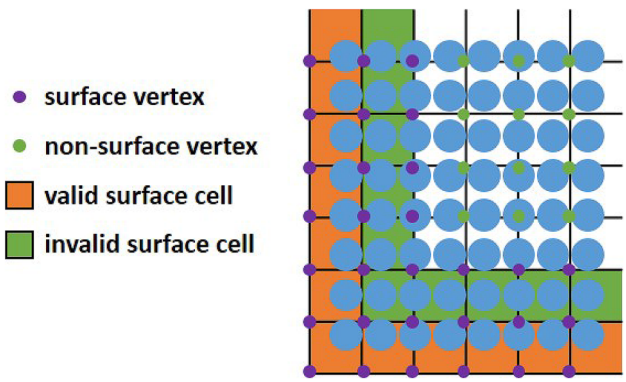


Fig. 6 Valid surface cell identification in two-dimensional case. Only those surface cells colorized as orange could generate surface triangles while the rest of green surface cells could not

grid vertices of scalar field, the valid surface cells account for only a small proportion of the total surface cells. Accordingly, we detect those valid surface cells and compress them into a continuous array to avoid launching idle threads and branching.

At the beginning of this step, *isValidSfCellArray* and *numVerOfCellArray* are allocated and initialized to 0. *isValidSfCellArray* is used to mark the corresponding surface cell whether it's valid or not while *numVerOfCellArray* whose length is equal to the number of surface vertices is for recording the number of triangles generated by MC algorithm of each surface cell. The invalid surface cells definitely produce 0 triangle and the valid surface cells generate at least one triangle on the contrary. For each surface cell i, we query the lookup table of MC algorithm according to the scalar value of the eight corner vertices to get the number of generated vertices *nVer*. If *nVer* is greater than 0, we set the value of *isValidSfCellArray*[i] and *numVerOfCellArray*[i] to 1 and *nVer*, respectively. It should be noted that the surface cell is absolutely invalid surface cell if any of the eight corner vertices are not surface vertices, otherwise a second layer is generated inside the fluid erroneously.

Then, we compress these valid surface cells into a continuous array likewise where the parallel exclusive prefix sum is employed to predict their storage indices in the compressed array. Both the prefix sum sequences of *isValidSfCellArray* and *numVerOfCellArray* are calculated for the sake of convenience of triangulation in parallel.

4.4 MC triangulation

In this stage, the triangulation algorithm is performed only on these valid surface cells. Each valid surface cell launches a thread where the conventional Marching Cubes algorithm is employed to obtain the intersection points of fluid isosurface. In order to prevent writing conflict of different threads, we utilize the prefix sum sequence of *numVerOfCellArray* to get the starting address where the generated vertices are saved. The surface meshes of fluids are finally constructed in this manner.

5 Experiment and analysis

To demonstrate the efficiency of our approach, we implemented both our method and the approach of Akinci [4] with the popular parallel computing platform CUDA 10.1, and applied it to four different scenarios which were simulated by a standard PCISPH solver: Double Breaking Dam (DBD), Bunny, Drop and Hemisphere with cell sizes of $r/2$ and r. We carried out all of the experiments on Intel Core i7 8700 with six 3.2 GHz cores, 16GB RAM and an NVIDIA GeForce RTX 2070 video card. Figure 9 shows several static images of the four scenarios. Here, we employed the scalar field function of Yu and Turk [32] as it could generate much more smooth surfaces. The neighborhood search data structures are established on GPU during the simulation stage, hence they are not taken into account for the benchmark results.

Table 2 Comparison of average reconstruction times between Ak12 and ours with cell size of r

Scene	Method	t-sf (ms)	t-tri (ms)	t-tol (ms)	MEM (GB)
DBD	Ak12	21.7	12.8	34.5	0.21
	ours	**3.2**	**2.5**	**5.7**	0.23
Bunny	Ak12	237.8	52.6	290.4	0.41
	ours	**12.5**	**9.4**	**21.9**	0.40
Drop	Ak12	146.1	70.6	216.7	0.49
	ours	**11.6**	**5.3**	**16.9**	0.52
Hemisphere	Ak12	197.2	88.3	285.5	0.70
	ours	**13.5**	**6.2**	**19.7**	0.71

Table 3 Comparison of average reconstruction times between Ak12 and ours with cell size of $r/2$

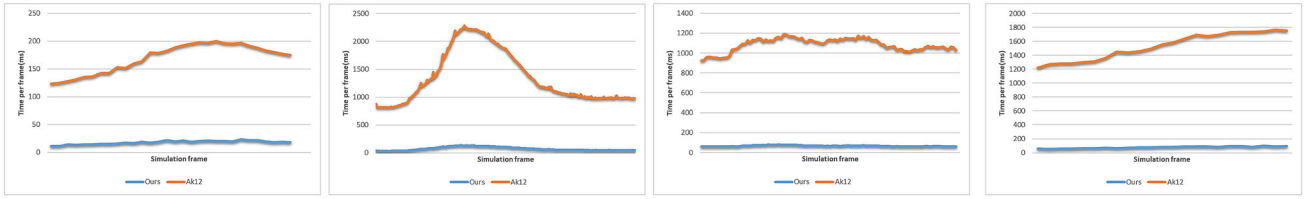
Scene	Method	t-sf (ms)	t-tri (ms)	t-tol (ms)	MEM (GB)
DBD	Ak12	59.5	110.8	170.3	0.43
	ours	**8.97**	**5.59**	**14.56**	0.42
Bunny	Ak12	482.8	1096.5	1579.3	0.95
	ours	**42.7**	**24.4**	**67.1**	0.97
Drop	Ak12	478.8	593.4	1072.2	1.19
	ours	**40.3**	**16.1**	**56.4**	1.22
Hemisphere	Ak12	750.2	827.0	1577.2	2.17
	ours	**47.4**	**16.2**	**63.6**	2.13

t-sf, t-tri, t-tol and MEM represent the same meaning as they do in Table 2
t-sf, t-tri and t-tol represent the consuming time of scalar field computation, triangulation and the total surface reconstruction. MEM denotes the memory consumption of the graphics card

Table 1 lists the general information of the four scenarios with two different cell size where $\#\text{particles}_{\text{surf}}$, $\#\text{particles}_{\text{total}}$, $\#\text{vertices}_{\text{surf}}$ and $\#\text{vertices}_{\text{total}}$ represent the number of surface particles, total particles, surface vertices and total vertices of the scalar field, respectively. #particles and #triangles denote the number of fluid particles and the number of generated triangles. Hence, we use $\frac{\#\text{particles}_{\text{surf}}}{\#\text{particles}_{\text{total}}}$ and $\frac{\#\text{vertices}_{\text{surf}}}{\#\text{vertices}_{\text{total}}}$ to define the ratio of surface particles and the ratio of surface vertices.

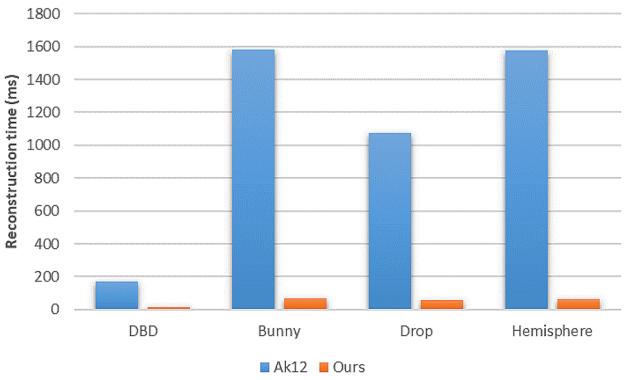
Firstly, we pay attention to the difference of surface particles ratio and surface vertices ratio between the smoothed color field and the method proposed by us, which is shown in the column $\frac{\#\text{particles}_{\text{surf}}}{\#\text{particles}_{\text{total}}}$ and the column $\frac{\#\text{vertices}_{\text{surf}}}{\#\text{vertices}_{\text{total}}}$ of Table 1. As can be seen from Table 1, our method identified much fewer surface particles than the smoothed color field, because our method is capable of accurately extracting those real surface particles. Omittance and misidentification of surface particles from which the smoothed color field suffers are definitely avoided in our scheme, which guarantees the precise identification of surface vertices. Moreover, with the increase in the number of particles, the advantages of our method become more and more obvious. This improvement could be nonnegligible since it lays the foundation of high performance for the subsequent steps.

We simulated four scenarios for experimental comparison. The first scene is the classic double breaking dam scene where two corner fluid blocks collapse and interact with cylindrical obstacles as illustrated in Fig. 9a. The second experiment named Bunny simulates a cylindrical fluid falls into a pool where a Stanford bunny is placed resulting in splashing water around, which is shown in Fig. 9b. And the third scenario is a scene as illustrated in Fig. 9c that simulates a Stanford bunny liquid falls into a water-filled pool. In the last scene, a large block of fluid flows down the slope and interacts with two spherical obstacles. From the first scene to the last scene, the number of fluid particles gradually increased as shown in Table 1. Tables 2 and 3 give the comparison of average reconstruction times of the four scenarios, and Fig. 7 displays the comparison of reconstruction times with cell size of $r/2$ over the simulation frames. From Tables 2 and 3, we could see that the consuming time of scalar field computation dominates as expected. Obviously, it takes much less time to reconstruct surface in our approach compared to [4]. In terms of memory cost, our scheme doesn't require much more memory than [4], because the newly added arrays for surface particles identification only store Booleans and 1 bit for the size of each entry is enough. We successfully achieve more than ten times faster in contrast to [4], where the efficiency is improved by an order of magnitude benefiting from our accurate identification of surface particles and well-designed

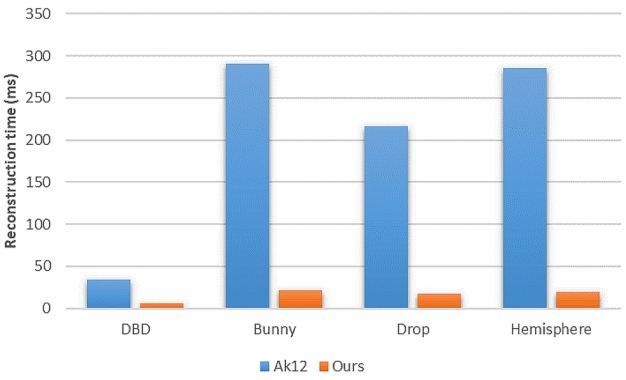


(a) DBD scene **(b)** Bunny scene **(c)** Drop scene **(d)** Hemisphere scene

Fig. 7 Comparison of surface reconstruction time over simulation steps with the cell size of $r/2$ between [4] and our approach



(a) cell size $= r/2$



(b) cell size $= r$

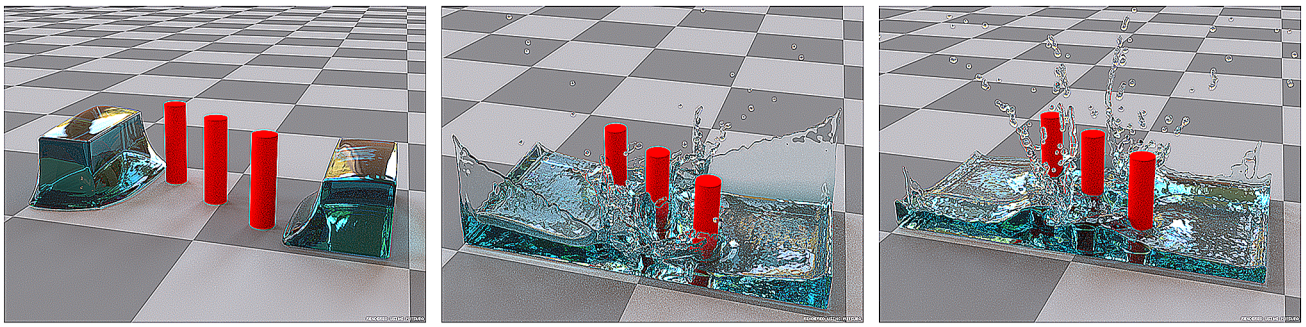
Fig. 8 Average reconstruction time per frame between our method and [4]

completely parallel reconstruction algorithm. As illustrated in Fig. 7, with simulation advancing, the reconstruction time of our method preserves stabilization robustly without too apparent fluctuation, while there is visible volatility during the simulation using the method of [4]. This is because the increasingly detailed and sharp features of fluid occurred with the interaction of fluid and obstacles as well as different fluid blocks, which scales up the reconstruction time to capture these detailed features. At the end of simulation, the fluid reached a stable and balanced state.

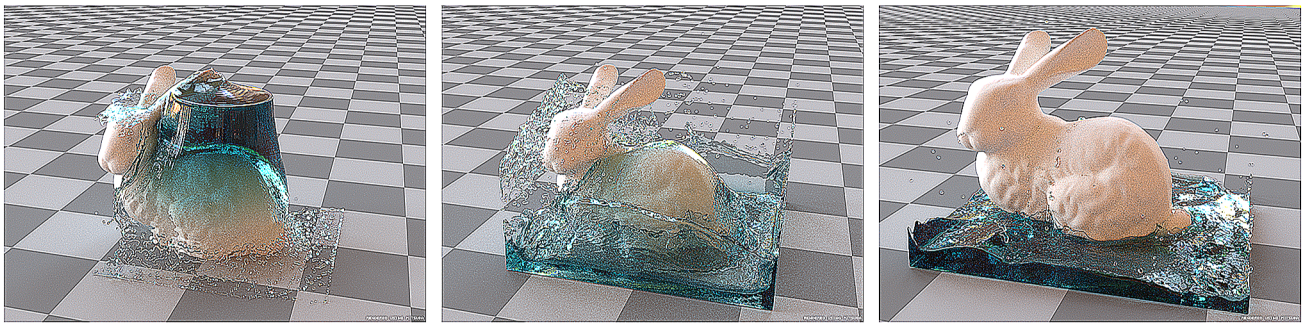
Figure 8 displays the comparison of reconstruction times of the method of Akinci and ours for all the above four scenarios with both cell sizes of $r/2$ and r. It's pretty straightforward to infer that our approach is much more efficient than the other one with achieving remarkable performance improvement. In addition, for finer grid, the reconstruction time of our method doesn't scale up too much. Figure 9 illustrates some of the frames rendered by Mitsuba of all the four scenarios whose surface were constructed by our approach, indicating that we have not sacrificed the quality of the surface meshes in the slightest. The fully parallel pipeline proposed by us also outperforms the ameliorated method presented by [31]. In the paper of [31], they improve the narrow band method of [4] in terms of memory efficiency achieving two times or three times faster than the original one, which is definitely not better than ours. Their scheme still is not parallelized completely, and the smoothed color field is employed to extract surface particles.

6 Conclusions

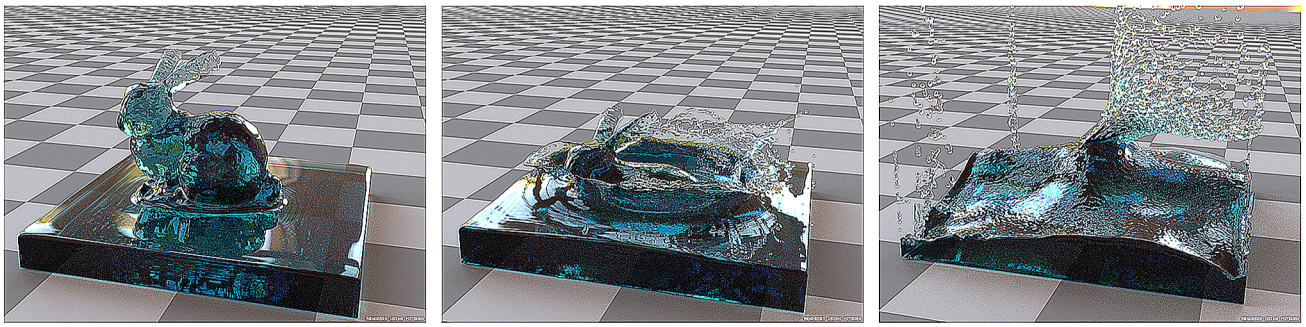
In this paper, we firstly present an accurate and parallel-friendly surface particles identification method that extracts those real surface particles quickly and precisely. Then, a fully parallel surface reconstruction method for particle-based fluids is proposed with the help of parallel exclusive prefix sum algorithm. Each step of our method is well-designed to execute in parallel architecture while avoiding branches, idle threads and race conditions as much as possible. We implement our method and compare it with previous approach, drawing a conclusion that our scheme could reach at least a dozen times faster. One of the reasons of much efficient performance of our approach is the much precise identification of surface particles, which is always ignored in general. The method of surface particles identification is of great simplicity and doesn't involve any sophisticated operation. Generally speaking, our approach successfully obtains significant performance improvement without sacrificing the quality of generated meshes and could work well with large-scale simulations and high-resolution scalar grid.



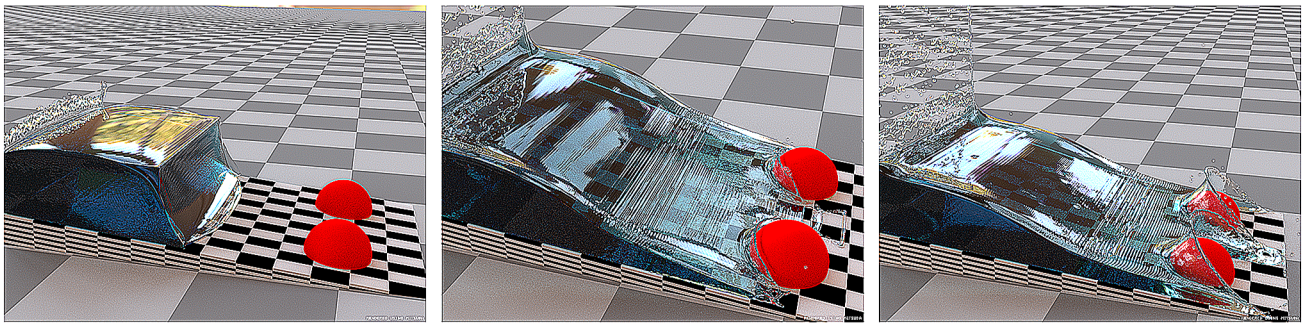
(a) Double breaking dam scene with 0.2M particles.



(b) Bunny scene with 0.9M particles.



(c) Drop scene with 2.4M particles.



(d) Hemisphere scene with 3.4M particles.

Fig. 9 Four experimental scenarios which were simulated by a PCISPH solver. The fluid surface meshes of the scenarios were reconstructed via our approach. These images were rendered by Mitsuba renderer (Jakob, W.: Mitsuba renderer, http://www.mitsuba-renderer.org)

Acknowledgements This work was supported by the Natural Science Foundation of Guangdong Province, China (Grant No. 2019-A1515011075).

Compliance with ethical standards

Conflict of interest We declare that we have no conflict of interest.

References

1. Adams, B., Pauly, M., Keiser, R., Guibas, L.J.: Adaptively sampled particle fluids. ACM Trans. Graph. (TOG) **26**, 48 (2007)
2. Akinci, G., Akinci, N., Ihmsen, M., Teschner, M.: An efficient surface reconstruction pipeline for particle-based fluids. In: Bender, J., Kuijper, A., Dieter, W.F., Guerin, E. (eds.) 9th Workshop on Virtual Reality Interactions and Physical Simulations, pp. 61–68. Eurographics Association, Darmstadt, Germany (2012). https://doi.org/10.2312/PE/vriphys/vriphys12/061-068
3. Akinci, G., Akinci, N., Oswald, E., Teschner, M.: Adaptive surface reconstruction for SPH using 3-level uniform grids. In: Oliveira, M.M., Skala, V. (eds.) 21st International Conference in central europe on computer graphics, visualization and computer vision in co operation with association, pp. 195–204. Plzen, Czech Republic. http://wscg.zcu.cz/WSCG2013/!_2013-WSCG-Fullproceedings.pdf (2013)
4. Akinci, G., Ihmsen, M., Akinci, N., Teschner, M.: Parallel surface reconstruction for particle-based fluids. In: Computer Graphics Forum, vol. 31, pp. 1797–1809. Wiley Online Library (2012)
5. Bhattacharya, H., Gao, Y., Bargteil, A.W.: A level-set method for skinning animated particle data. IEEE Trans. Vis. Comput. Graph. **21**(3), 315–327 (2014)
6. Blinn, J.F.: A generalization of algebraic surface drawing. ACM Trans. Graph. (TOG) **1**(3), 235–256 (1982)
7. Bridson, R.E., Fedkiw, R.: Computational aspects of dynamic surfaces. Ph.D. thesis, Stanford University Stanford, California (2003)
8. Canezin, F., Guennebaud, G., Barthe, L.: Topology-aware neighborhoods for point-based simulation and reconstruction. In: Solenthaler, B., Teschner, M., Kavan, L., Wojtan, C. (eds.) Proceedings of the ACM Siggraph/eurographics symposium on computer animation, pp. 37–47. Zurich, Switzerland. http://dl.acm.org/citation.cfm?id=2982825 (2016)
9. Gao*, M., Wang*, X., Wu*, K., Pradhana, A., Sifakis, E., Yuksel, C., Jiang, C.: Gpu optimization of material point methods. ACM Trans. Graph. (Proceedings of SIGGRAPH ASIA 2018) **37**(6) (2018) (*Joint First Authors)
10. Hadi, N.: Big data simulation for surface reconstruction on CPU-GPU platform. J. Phys. Conf. Ser. **1192**, 012006 (2019)
11. Hadi, N.A., Alias, N.: 3-dimensional human head reconstruction using cubic spline surface on CPU-GPU platform. In: Proceedings of the 2019 4th International Conference on Intelligent Information Technology, pp. 16–20. ACM (2019)
12. Houston, B., Wiebe, M., Batty, C.: Rle sparse level sets. In: ACM SIGGRAPH 2004 Sketches, p. 137. Citeseer (2004)
13. Ju, T., Udeshi, T.: Intersection-free contouring on an octree grid. In: Proceedings of the 14th Pacific Conference on Computer Graphics and Applications, vol. 3. Citeseer (2006)
14. Koren, Y., Carmel, L.: Visualization of labeled data using linear transformations. In: IEEE Symposium on Information Visualization 2003 (IEEE Cat. No. 03TH8714), pp. 121–128. IEEE (2003)
15. Lee, H., Yang, H.S.: Real-time marching-cube-based LOD surface modeling of 3D objects. In: 14th International Conference on Artificial Reality and Telexistence, ICAT 2004 (2004)
16. Lee, W., Hasan, S., Shamsuddin, S., Lopes, N.: Gpumlib: deep learning SOM library for surface reconstruction. International Journal of Advances in Soft Computing and its Applications (IJASCA), vol. 9, no. 2, University of Technology Malaysia (2017)
17. Loop, C.T.: Sparse GPU voxelization for 3D surface reconstruction. US Patent 9,984,498 (2018)
18. Lorensen, W.E., Cline, H.E.: Marching cubes: a high resolution 3D surface construction algorithm. In: ACM Siggraph Computer Graphics, vol. 21, pp. 163–169. ACM (1987)
19. Manson, J., Schaefer, S.: Isosurfaces over simplicial partitions of multiresolution grids. In: Computer Graphics Forum, vol. 29, pp. 377–385. Wiley Online Library (2010)
20. McAdams, A., Selle, A., Tamstorf, R., Teran, J., Sifakis, E.: Computing the singular value decomposition of 3×3 matrices with minimal branching and elementary floating point operations. University of Wisconsin-Madison Department of Computer Sciences, Tech. rep. (2011)
21. Müller, M., Charypar, D., Gross, M.: Particle-based fluid simulation for interactive applications. In: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation, pp. 154–159. Eurographics Association (2003)
22. Nielsen, M.B., Museth, K.: Dynamic tubular grid: an efficient data structure and algorithms for high resolution level sets. J. Sci. Comput. **26**(3), 261–299 (2006)
23. Nielsen, M.B., Nilsson, O., Söderström, A., Museth, K.: Out-of-core and compressed level set methods. ACM Trans. Graph. (TOG) **26**(4), 16 (2007)
24. Onderik, J., Chládek, M., Durikovič, R.: SPH with small scale details and improved surface reconstruction. In: Proceedings of the 27th Spring Conference on Computer Graphics, pp. 29–36 (2011)
25. Solenthaler, B., Pajarola, R.: Predictive-corrective incompressible SPH. ACM Trans. Graph. (TOG) **28**, 40 (2009)
26. Solenthaler, B., Schläfli, J., Pajarola, R.: A unified particle model for fluid–solid interactions. Comput. Anim. Virtual Worlds **18**(1), 69–82 (2007)
27. Velasco, F., Torres, J.C.: Cell octrees: a new data structure for volume modeling and visualization. VMV **1**, 151–158 (2001)
28. Wang, X., Ban, X., Zhang, Y., Pan, Z., Liu, S.: Anisotropic surface reconstruction for multiphase fluids. In: 2017 International Conference on Cyberworlds (CW), pp. 118–125. IEEE (2017)
29. Wiemann, T., Mitschke, I., Mock, A., Hertzberg, J.: Surface reconstruction from arbitrarily large point clouds. In: 2018 Second IEEE International Conference on Robotic Computing (IRC), pp. 278–281. IEEE (2018)
30. Williams, B.W.: Fluid surface reconstruction from particles. Ph.D. thesis, University of British Columbia (2008)
31. Wu, W., Li, H., Su, T., Liu, H., Lv, Z.: GPU-accelerated SPH fluids surface reconstruction using two-level spatial uniform grids. Visual Comput. **33**(11), 1429–1442 (2017)
32. Yu, J., Turk, G.: Reconstructing surfaces of particle-based fluids using anisotropic kernels. ACM Trans. Graph. (TOG) **32**(1), 5 (2013)
33. Zhou, K., Gong, M., Huang, X., Guo, B.: Data-parallel octrees for surface reconstruction. IEEE Trans. Vis. Comput. Graph. **17**(5), 669–681 (2011)
34. Zhu, Y., Bridson, R.: Animating sand as a fluid. ACM Trans. Graph. (TOG) **24**(3), 965–972 (2005)

Wencong Yang received his BE degree in computer science from School of Data and Computer Science of Sun Yat-sen University in 2019. He is currently a ME student at Intelligent and Multimedia Science Laboratory, Sun Yat-sen University. His research interests include fluid simulation and non-photorealistic rendering.

Chengying Gao is an associate professor of School of Data and Computer Science, Sun Yat-sen University. She received the PhD degree in computer science from School of Information Science and Technology, Sun Yat-sen University, in 2003. Her research interests include computer graphics and visual media computing.