Contents lists available at ScienceDirect

Computers & Graphics

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

Enhanced narrow band surface reconstruction with anisotropic kernel

Qiaorui Chen[a], Shuai Zhang[a,*], Yao Zheng[a]

[a]*Zhejiang University, Hangzhou, 310000, China*

ARTICLE INFO

Article history:

Keywords: Surface Reconstruction, Narrow Band, Parallel, Particle-based fluids, anisotropic kernel

ABSTRACT

This paper presents a novel pipeline that improves the efficiency of narrowband surface reconstruction for particle-based fluids. Different from previous narrow band methods, we only construct the scalar field in the outer surface region. A method with robust criteria for detecting the outer surface vertices of the scalar field is then proposed to eliminate the redundant inner vertices, leading to a significant reduction in the computational complexity and memory consumption. The presented pipeline is fully parallel with a parallel compact memory scheme to avoid race conditions. Several experiments are conducted to compare our present method with the state-of-the-art approach, and the results indicate that our method saves nearly half of the computing time and ten percent of the memory consumption.

1. Introduction

Particle-based fluid simulations have been gaining increased interest in computer graphics. Particle-based methods can easily simulate complex phenomena, including free-surface flows with large deformations, fragmentation, merging and solids with complex geometries. Among various particle-based approaches, smoothed particle hydrodynamics (SPH)[1] is a popular method for producing realistic results. In the context of SPH, in order to render the simulation results, it usually employs a surface reconstruction pipeline that transforms particles into polygonal meshes.

At present, a common fluid surface reconstruction pipeline can be summarized to two main steps: The first step is to estimate an appropriate scalar field that matches the underlying flow as well as possible. Commonly, these smoothing techniques [2, 3, 4, 5, 6, 7] are employed to construct a scalar field. These techniques consider particles within an influence region in the estimation of scalar values at grid vertices. The second step is to extract the surface (isosurface) from the scalar field with marching cubes (MC) [8]. The employed scalar field grid cell size significantly influences the quality of the reconstructed surface as well as the computation time. Small cell size is required to reconstruct smooth surfaces and to catch fine details, while the computation time grows cubically, which is unacceptable for high-performance demands [9]. As shown in Fig 1, while a small cell size produces artifact-free, high-quality surfaces, a larger cell size are usually used for previewing purposes.

Researchers have been trying to scale the computation complexity and memory consumption with the fluid surface instead of the volume, since only a small part of scalar field grid cells pass through a fluid surface. In summary, there are three main ways: adaptive grid methods[10, 11, 12, 13], narrow band techniques[9, 14, 15]. In adaptive grid methods, the scalar field grids are further subdivided into surface regions to capture finer fluid details, but the implementation is tough, especially for parallelization. In contrast, narrowband methods work with uniform scalar field grids and can be applied with an efficient parallelization technique. And the scalar field estimation is only performed on surface regions.

In these methods, any scalar grid vertex that is close enough to a surface particle is defined as a surface vertex. The surface region is formed by these surface vertices and the scalar

*Corresponding author:
e-mail: shuaizhang@zju.edu.cn (Shuai Zhang), shuaizhang@zju.edu.cn (Shuai Zhang)

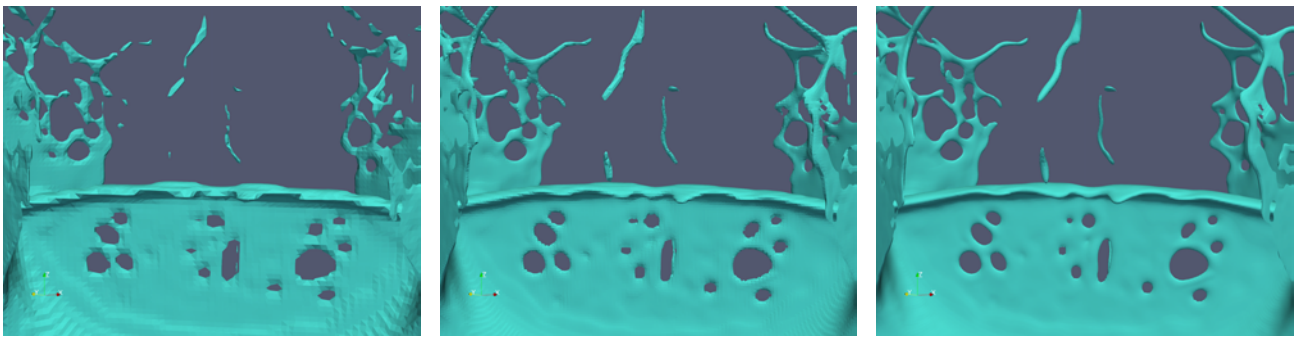

Fig. 1: Close-up view of the reconstructed surface for the Double Dam Break scene with three different MC cell sizes: left 2r, middle r, right r/2 and the smoothing radius is set to 4r. The average surface reconstruction time per frame for the our proposed approach is 1.13, 3.43, 16.9 s, respectively

field estimation is only performed on the surface regions. Because surface vertices are determined only by a fixed distance from the surface particles, there are inner vertices that are located inside the fluid volume. However, the isosurface should coat the fluid particles, which means that the inner surface region does not have an isosurface. Thus, these inner vertices are mostly redundant for extracting isosurface (reconstructing surfaces). Moreover, the time required to compute the values of inner vertices is much longer than that required to compute outer vertices because of the larger number of particles in the vicinity of these inner vertices that involve many more kernel evaluations. As related papers [9] reported, up to 90% of the surface reconstruction time is spent computing vertex values (also called the construction of a scalar field), which is the main bottleneck in surface reconstruction. Therefore, it is significant to eliminate these redundant inner vertices to greatly improve the reconstruction efficiency.

Our contribution The main contribution of this paper is that we propose a method with robust criteria to eliminate inner vertices of scalar field grids without sacrificing the quality of reconstructed surface. Unnecessary processing and data storage for these inner vertices are avoided. The comparison experiments with the state-of-the-art method show that our approach saves nearly half of the computing time and 10% of the memory consumption.

2. Scalar field estimation

In general, the smoothing technique has a strong influence on the reconstructed surface quality. While [2, 4] is faster than [6], it suffers from spurious blobs. The anisotropic kernel method[6] yields smooth, unbroken surfaces, especially for liquid film and filament. It has been widely adopted in the recent papers [7, 16, 17, 18].

Yu and Turk [6] argued that the spherical shape of a smoothing kernel is not suitable to describe the density distribution near a surface. They designed an approach to capture the density distribution more accurately by allowing the smoothing kernels to be anisotropic, that is, replacing the support radius h with a real positive definite matrix $\boldsymbol{G}$. Hence, the value of the scalar field around the query vertex $\boldsymbol{x}$ is defined as

$$\phi(\boldsymbol{x}) = \sum_j \frac{m_j}{\rho_j} W(\boldsymbol{x} - \boldsymbol{x}_j, \boldsymbol{G}_j) \tag{1}$$

where j is the query vertex's neighborhood fluid particle index with support radius h. m_j and ρ_j denote the particle mass and density respectively, where m_j is constant and ρ_j varies and need to be evaluated via isotropic weighting function at each frame[19]. And W is a smoothing kernel of the form

$$W(\boldsymbol{x} - \boldsymbol{x}_j, \boldsymbol{G}_j) = \frac{315\|\boldsymbol{G}_j\|}{64\pi h}(1 - |\boldsymbol{G}_j(\boldsymbol{x} - \boldsymbol{x}_j)|^2)^3 \tag{2}$$

The numbers 315 and 64 ensure the smooth kernel function meet the regularization condition on the support domain and are firstly introduced in reference [19].

To determine the transform matrix $\boldsymbol{G}_i$, the weighted version of principal component analysis (WPCA)[20] is applied to the neighborhood particle positions. WPCA constructs a weighted covariance matrix $\boldsymbol{C}_i$ with a zero empirical mean.

$$\boldsymbol{C_i} = \sum_j \omega_{ij}(\boldsymbol{x}_j - \bar{\boldsymbol{x}}_i)(\boldsymbol{x}_j - \bar{\boldsymbol{x}}_i)^T / \sum_j \omega_{ij} \tag{3}$$

$$\bar{\boldsymbol{x}}_i = \sum_j \omega_{ij}\boldsymbol{x}_j / \sum_j \omega_{ij} \tag{4}$$

The function w_{ij} is an isotropic weighting function with respect to particle i and its neighborhood particle j with support radius h

$$w_{ij} = \begin{cases} 1 - (|(\boldsymbol{x}_j - \boldsymbol{x}_i)/h|)^3, & 0 < |\boldsymbol{x}_j - \boldsymbol{x}_i| < h \\ 0, & othersize \end{cases} \tag{5}$$

Because $\boldsymbol{C_i}$ is a real symmetric matrix, it can be treated by a singular value decomposition (SVD). Then, C_i can be expressed as

$$\boldsymbol{C}_i = \boldsymbol{R}\boldsymbol{\Sigma}\boldsymbol{R}^T \tag{6}$$

$$\boldsymbol{\Sigma} = diag(\lambda_1, \lambda_2, \lambda_3)\ (\lambda_1 \le \lambda_2 \le \lambda_3) \tag{7}$$

To handle some special cases, Yu and Turk [6] adjusted eigenvalues in diagonal matrix $\mathbf{\Sigma}$ and obtained a modified diagonal matrix $\tilde{\mathbf{\Sigma}}$. After that, an anisotropy matrix $\boldsymbol{G}_i$ is constructed to match the smoothing kernel W with the output of WPCA.

$$\boldsymbol{G}_i = \frac{1}{h}\boldsymbol{R}\tilde{\Sigma}^{-1}\boldsymbol{R}^T \tag{8}$$

It is important to point out that isosurface should give a surface representation that coats the fluid particles. If not, the reconstruction of liquid streams, liquid ligaments and droplets will fail and superfluous surface may be reconstructed. As shown in Fig 2, taking 2D as an example. The isosurface with a large cutoff value $\phi(\boldsymbol{x}) = 1$ (red line) does not coat the fluid particles, the redundant surface is reconstructed and the fluid detail (droplet) is lost. On the contrary, the isosurface with a small $\phi(\boldsymbol{x}) = 0.2$ (green line) or moderate cutoff value $\phi(\boldsymbol{x}) = 0.5$ (purple lines) that is warped around the fluid particles reconstructs an artifact-free, detailed and smooth surface. Therefore, we should set a reasonable value of isosurface to wrap the fluid particles. In all the experiments in this paper, the value of the isosurface $\phi(\boldsymbol{x})$ is empirically taken as 0.5 to obtain artifact-free and smooth results.

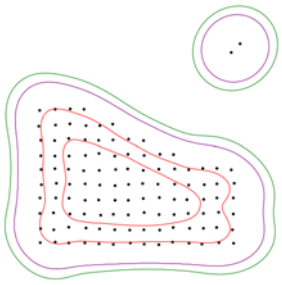

Fig. 2: Isosurface representation with different cutoff value $\phi(\boldsymbol{x}) = 0.2$ (green) $\phi(\boldsymbol{x}) = 0.5$ (purple) and $\phi(\boldsymbol{x}) = 1$ (red)

3. Surface reconstruction pipeline

The computation time for extracting smooth surface is mainly influenced by the number of involving vertices of scalar field grid. As mentioned above, since the isosurface must distribute outside the fluid particles, we should pay more attention to the outer vertices of scalar filed grid outside the fluid particles rather than the inner vertices. Based on this conception, in contrast to previous methods [9, 14, 15], we determine the surface vertices only in outer surface region.

In this section, we present a feasible implementation which extracts the outer surface vertices and robust criteria are proposed to prevent the failure cases (Sect 3.1). Also, a parallel formulation for compacting surface vertices and valid cells into a continuous array is described (Sect 3.2), which avoids inherent data dependencies.

Fig 3 gives our enhanced narrow band surface reconstruction pipeline steps. In the presented pipeline, at the beginning, the surface particles are determined in a pre-processing step, the spatial hashing grid-based strategy [15] is employed. After that, the covariance matrix of each surface particle is constructed via WPCA and the SVD on the matrix is performed to obtain eigenvector and eigenvalue. By analyzing the relationship between eigenvalues, the oriented normal of each surface particle and the distribution of neighbor particles are obtained. In our proposed method, we make full use of these information to extract outer surface vertices. The scalar field estimation is only performed on these extracted vertices. In the end, the MC algorithm[8] is employed to extract surface meshes from the constructed scalar field grid.

3.1. Optimized surface vertex detection

After detection of surface particles, surface particles are utilized to detect surface vertices. In previous methods, any grid vertices of the scalar field that are close to a surface particle are marked as surface vertices and scalar field computation is only performed for these vertices. The close proximity can be defined as the axis-aligned bounding box (AABB) around the particle, which spans a length of h in each direction. As shown in Fig 4(left), the vertices in the AABB box are all considered as surface vertices, including red and black dots. However, approximately half of extracted vertices are inner vertices and the black dots are all inner vertices inside the fluid particles. As mentioned above (Sect 2), the isosurface must be distributed outside the fluid particles, thus the inner vertices are mostly invalid for extracting the isosurface.

In this section, we present an optimized surface vertex detection method to eliminate redundant inner vertices. In point-based surface reconstruction [21], the oriented normals locally define the reconstructed surface to the first order and identify the inside/outside and, hence, the topology of the underlying shape. The inner vertices are inside surface particles. We naturally think of using the oriented normal to eliminate these inner vertices. In our work, the SVD on the covariance matrix of each surface particle is performed, and the fluid particle set is a volume particle presentation. Then the oriented normals can be determined without any complex computations.

For each surface particles i, the normals n_i are assigned by the eigenvector $\boldsymbol{v}_1$ with the smallest eigenvalue, and we use the mean position $\bar{\boldsymbol{x}}_i$ (Eq 4) to obtain the unified outward direction.

$$\boldsymbol{n}_i = sign((\boldsymbol{x}_i - \bar{\boldsymbol{x}}_i) \cdot \boldsymbol{v}_1) \cdot \boldsymbol{v}_1 \tag{9}$$

With the oriented normal n_i, we introduce a fast exclusion test to check whether a vertex is outside the fluid volume; it is formulated as

$$\boldsymbol{n}_i \cdot (\boldsymbol{x} - \boldsymbol{x}_i)^T > 0 \tag{10}$$

where $\boldsymbol{x}$ represents the vertex of the scalar field grid position.

As shown in Fig 4(right), for a surface particle, each vertex located within the bounding box is polled and only the vertices that pass through the exclusion test are considered surface vertices (red dots). Apparently, approximately half of the inner vertices (black dots) are excluded. It reduced nearly half of the invalid threads for the subsequent steps, which consumed most of the surface reconstruction time. While simple, it is sensible.

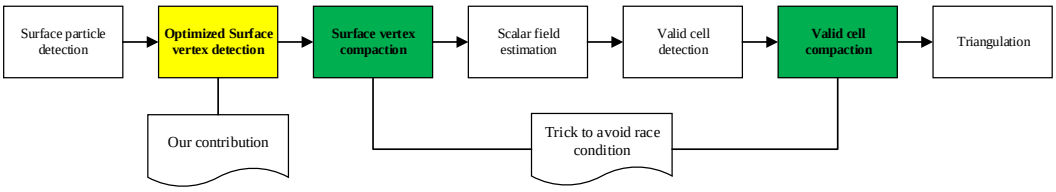

Fig. 3: Our surface reconstruction pipeline

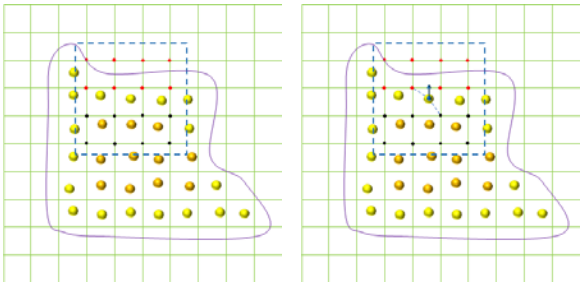

Fig. 4: Illustration of surface vertex detection

.

This idea is heuristic and in order to ensure the correct results in practical application. Two points must to be guaranteed for surface reconstruction. The first point is that each valid MC cell contains a part of isosurface. The second point is that the isosurface are all located in the valid MC cells. The valid MC cell refers to that will be pass to triangulation stage and can generate triangle meshes inside. If the first point is not satisfied, there will be double layer problem. If the second point is not satisfied, there will be a problem of incomplete topology. Therefore, there are three issues that must be considered, otherwise it will lead to failure reconstruction.

1. In valid MC cell detection stage, each surface vertex of scalar field grid corresponds to an MC cell named surface MC cell. Before triangulation, we traverse each surface MC cell and mark the MC cell as a valid MC cell. In MC algorithm[8], it only needs meet the one condition: the scalar field values of the eight corner vertices of the cell are not all bigger or smaller than isosurface value. However, unlike previous methods, the extracted surface vertices are only located on the outer surface region in our pipeline. If we only use this condition, there will be double layers problem, as shown in Fig 5. Taking 2D example in Fig 6(left) to illustrate the cause of the problem, because the value of inner vertices (black dots) is not estimated, in the initial, its value was set to zero, which is less than the value of the isosurface value. Using the common condition, the MC cells (rectangles with orange dotted lines) will be considered as valid cell and internal redundant surface meshes (red line) will be generated, which leads to the problem of double layers. Mistaken valid cell does not contain a part of isosurface (purple line) and it does not meet the first point To solve this problem, on the basis of the original we add a condition: all eight corner vertices of the cell are surface vertices. It prevent the un-estimated vertices participation in the valid cell detection stage, as shown in Fig 6 (right). It ensures that if a surface cell is valid cell, it must contain a part of isosurface. However, the new condition will cause the second and third problems. The isosurface (purple line) is not all located in the valid MC cells and the second point is not satisfied.

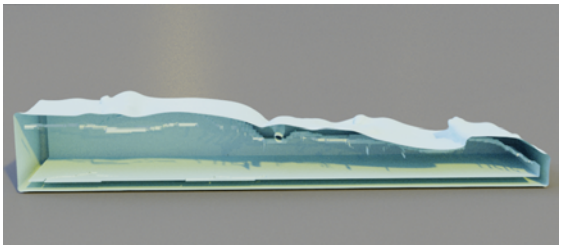

Fig. 5: The reconstructed surface is cut along the Y-axis to present the double layer problem with the Breaking Dam scene

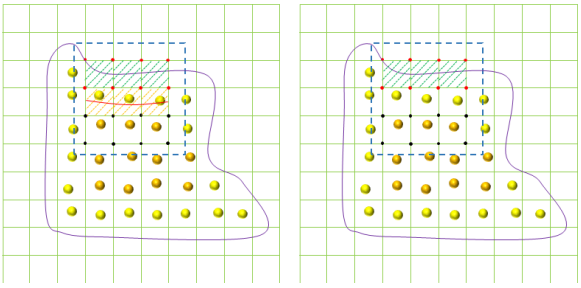

Fig. 6: Illustration of double layers problem

.

2. The second problem is that when dealing with single-layer particles or splash particles, an unclosed surface will appear. As shown in Fig 8, in the thin-fluid-feature area, there is unclosed reconstructed surface. To analyze the problem, taking a 2D example as an illustration in Fig 7 (left), the distribution of particles is single-layered. In this case, their normal direction is consistent, and if we continue to adopt the exclusion test, due to the new condition, only the MC cells (rectangles with green dotted lines) formed by the identified surface vertices (red dots) can be considered as valid cells and can extract isosurface, resulting in an unclosed surface problem, because a half of isosurface is not located in the valid MC cells. To solve this problem, for this class of particles, we discard the exclusion test, and a complete surface

will be reconstructed, as shown in Fig 7 (right).

Now, the difficulty of this problem is how to identify these single-layer particles. Typically, single-layer particles are thin-fluid-feature particles. Thus, the single-layer particles can be identified by identifying the thin-fluid-feature particles. In Eq 7, eigenvalue λ_l measures the variation in the particle position along the direction of each corresponding column $\boldsymbol{v}_l$ of $\boldsymbol{R}$. By comparing and analyzing the relationship between the maximum and minimum eigenvalues, we can obtain the stretching degree of the particle. In the thin-fluid-feature region, stretching is obvious. Thus, the thin-fluid-feature particle can be identified by

$$\lambda_1 \leq \alpha\lambda_3 \tag{11}$$

where α denotes the threshold that determines the degree of thinness. This formulation works well in identifying thin-fluid-feature, as show in Fig 9. Splash particles are identified by the number $|J|$ of particles around this particle; if $|J| < N$, we consider them isolated particles. In our implementation, we set α equal to 0.25 and N to 25.

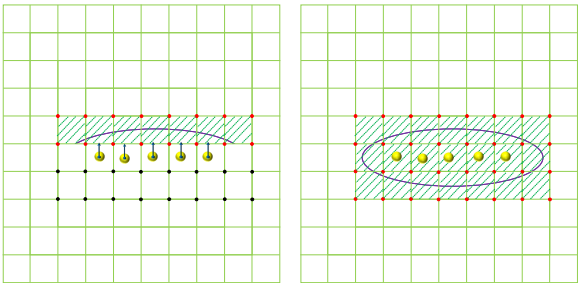

Fig. 7: Illustration of the unclosed surface problem

3. The third problem arises when there is a need to quickly preview the effect in the case of a low-quality surface, which results in a surface with holes. As shown in Fig 10(left and middle), when the MC grid cell size is equal to $2r$ or r, the topology of the mesh is discontinuous in the Breaking Dam scene. And with the increase of the cell size, this problem becomes more obvious. We analyzed the reasons for this problem. As shown in Fig 11(left), in this 2D illustration, the scalar field is coarse, and the purple line represents the isosurface. The isosurface coats the fluid particles as expected; however, the MC grid cell, which is formed by surface vertices (red dots) and inner vertices (black dots), due to the new condition, will not be considered as valid MC cell. As a result, isosurface is not completely located in MC valid cells. Where triangles should be generated, it dose not occur, resulting in surface holes problem.

   To address this issue, the outer surface vertices mark their adjacent vertices that reside in AABB as surface vertices. There are four adjacent vertices in the two-dimensional case, while there are six adjacent vertices in the three-dimensional case. As shown in Fig 11(right), the red surface vertex marks the adjacent vertices to surface vertices in the direction indicated by the yellow arrows. After this operation, this MC grid cell (rectangles with dotted lines) can be considered as valid MC cell and can be passed to the triangulation stage, and surface holes disappear.

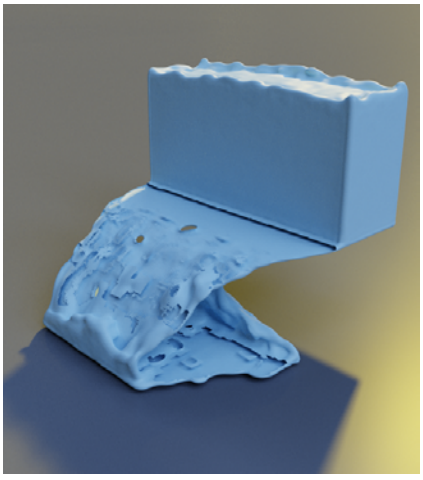

Fig. 8: The failure reconstruction at thin-fluid-feature area. Careful observation shows that there is unclosed reconstructed surface at thin-fluid-feature area

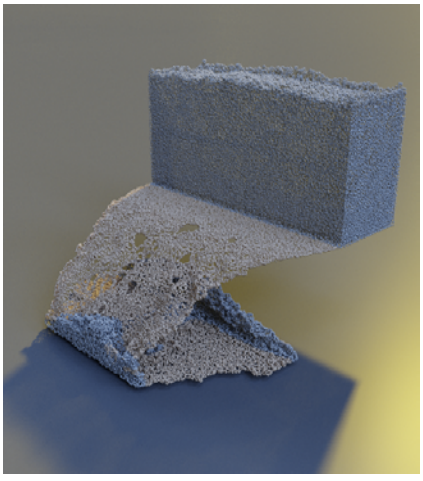

Fig. 9: The dirt-colored particles encodes the identified thin-fluid-feature particles

Our optimized surface vertex detection in surface reconstruction is summarised in Algorithm 1.

3.2. Surface vertex compaction & valid cell compaction

In fact, neither the surface vertices nor valid MC cells account for a small proportion of the grid vertices of the scalar field. Applying conditional branches to make subsequent calculations execute only on these small proportions is unwise and leads to warp divergence. Hence, it is necessary to extract these small proportions into a continuous array. In parallel implementations, a major difficulty is to avoid potential race conditions, i.e., multiple threads should never try to write different data to the same memory address at the same time. In this section, we introduce the *parallel compact memory scheme* to address this issue.

Parallel Compact Memory Scheme. As shown in Fig 12, array A indicates whether a vertex or MC cell should be retained,

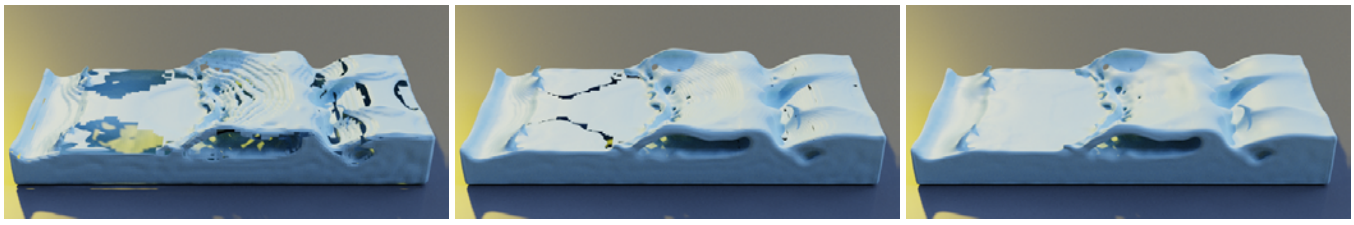

Fig. 10: The surface hole problem presented with the Breaking Dam scene with three different MC cell sizes: left 2r, middle r, right r/2 and the smoothing radius is set to 4r. In the case of no treatment, with the decrease of MC cell size, the problem is alleviated

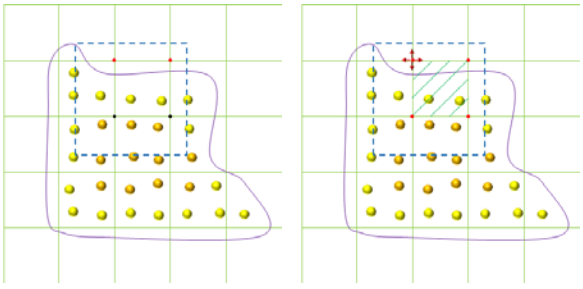

Fig. 11: Illustration of the surface hole problem

where 1 means it will be retained and array *BNum* records how many triangle vertices can be produced in this MC cell. The number of generated triangle vertices is determined according to the MC look-up table [8]. We use the *Thrust* CUDA library to scan *A* and *BNum* and then obtain arrays *AScan* and *BNumScan*, which store the address in *A* of the retained surface vertex and predict the generated triangle vertex storage indices. For a better illustration, the general pseudocode is given in 2.

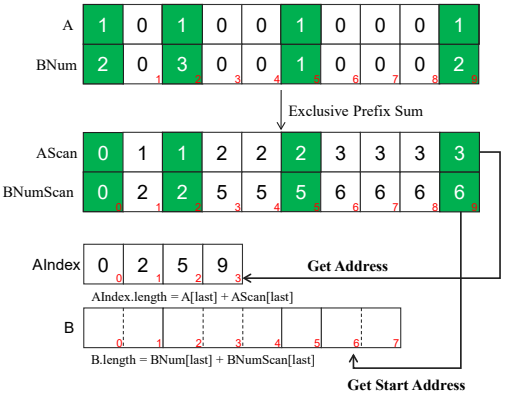

Fig. 12: Parallel Compact Memory Scheme

3.3. Scalar field estimation

After detecting and compacting surface vertices, their surface vertex values are estimated. Each GPU thread is launched for each surface vertex to estimate the value via the anisotropic kernel method[6], which was discussed in Sect 2. This step is the most complex and time consuming. The number of surface vertices determines the elapsed time. Therefore, it is meaningful to remove the unnecessary inner vertices and compact surface vertices in a continuous array to avoid idle threads.

Algorithm 1: Optimized surface vertex detection in surface reconstruction

```
foreach surface particle do
    compute covariance martirx C_i via Eq 3;
    perform SVD to obtain eigen vector v_i and value λ_i;
    record the number J of its neighboor particles;
    if λ_3 ≥ αλ_1 or J < N then
        thin-fluid-feature = true;
    end
    compute AABB;
    foreach vertex in AABB do
        if thin-fluid-feature or n_i · (x − x_i) > 0 then
            mark it as surface vertex;
            foreach adjacent vertex in AABB do
                mark it as surface vertex;
            end
        end
    end
end
foreach surface vertex do
    compute scalar value of surface vertex via Eq 1;
end
foreach surface vertex do
    create a cell;
    if all eight vertices are surface vertices then
        if scalar values of corner vertices are not all
          positive or negative then
            pass the cell to triangulation stage;
        end
    end
end
```

3.4. Valid cell detection

After calculating the value of each surface vertex, we need to determine which MC cells are valid cells. As mentioned above, We traverse each surface MC cell and mark the MC cell as a valid MC cell if it meets the following two conditions:

1. All eight corner vertices of the cell are surface vertices
2. The scalar field values of the eight corner vertices of the cell are not all positive or negative.

Similar to surface vertices, we also compact those valid cells into a continuous array, which is discussed in detail in Sect 3.2.

Algorithm 2: *Parallel compact memory scheme* algorithm

Input: A and BNum
Output: AIndex and B
thrust::exclusive_scan(A) to get AScan;
thrust::exclusive_scan(BNum) to get BNumScan;
AIndex.resize(A[last] + AScan[last]);
B.resize(BNum[last] + BNumScan[last]);
foreach *element i in A* **do**
  if *A[i] > 0* **then**
    AIndex[AScan[i]]=i;
  end
end
foreach *element i in AIndex* **do**
  Index = AIndex[i];
  Start = BNumScan[Index];
  foreach *each generated* b_j *in this element* ($j <$ *BNum[Index])* **do**
    B[Start + j] = b_j;
  end
end

Different from surface vertex compaction, valid cell compaction has additional arrays *BNum* and *B*, where *BNum* is used to record how many triangle vertices can be produced in the valid cell and *B* is used to predict the starting address that saves the generated triangle vertices. After that, triangulation starts, and the MC algorithm[8] is adopted to generate triangles.

4. Experimental results

Our approach and comparison methods[9, 15] are implemented with CUDA 9.0 and is run on a computer with an NVIDIA GPU GeForce RTX 2080 and Intel CPU i7 10700 KF with eight 3.8GHz.

Five different scenarios were simulated by DualSPH [22]: Breaking Dam (BD) (Fig 13), Double Breaking Dam (DBD) (Fig 14), Monkey (MK) (Fig 15), External Force (EF) (Fig 16) and Water Crown (WR) (Fig 17). For all scenes, the spatial grid cell size is set to $2r$, where r denotes the equilibrium distance of the particles. The support radius h is equal to $4r$ with MC cell sizes of $r/2$ or r. All images of the scenes are generated with an MC cell size r. All of the results are rendered by the Blender Cycles render engine (https://www.blender.org/).

Table 1 lists the general information of the five scenarios with two different MC cell sizes in our approach, where $\#particles$, $\#triangles$, $\frac{\#particles_{surf}}{\#particles_{total}}$, $\frac{\#vertices_{surf}}{\#vertices_{total}}$ and $\frac{\#mccell_{valid}}{\#mccell_{total}}$ denote the number of fluid particles, the number of generated triangles, the ratio of surface particles, the ratio of surface vertices and the ratio of valid MC cells, respectively. t_{total} denotes the average total running time of surface reconstruction, including memory allocation as well as memory transfer to and from GPU. From Table 1, neither the surface vertices nor valid cells account for a very small proportion, as expected. Thus, it is necessary to compact these surface vertices or valid cells into a continuous array.

The biggest contribution of our work is that we propose a method with robust criteria to eliminate inner vertices of scalar filed grids without sacrificing the quality of reconstructed surface, leading greatly improvement in efficiency. In order to illustrate the significance of our contribution more clearly, we made a comparison with Yang *et al.*[15] via the control variable method. The only difference between our and Yang *et al.*[15] is that our approach removes most of the redundant inner vertices in surface vertices detection stage.

Table 2 gives the comparison of the average times and memory consumptions between our and the Yang[15] method. t_{sf} (s) and t_{tri} (s) denote the scalar field construction and the triangulation consumption time, respectively. t_{com}/t_{our} represents the ratio of the total time consumption of the comparison methods to our method. *mem* stands for the max GPU memory (GB) consumed during the surface reconstruction. From Table 2, it can be seen that the running time of scalar field computation dominates as expected because the number of estimated vertices is usually tens of millions. From these five experiments, we can intuitively see that our method is approximately twice as fast as Yang's approach. Moreover, it can be observed that in our method, the number of vertices is reduced by approximately 30%, but the time consumption is reduced by more than half. This is because the vertices we remove are all inner vertices and there are far more fluid particles in the neighborhood of the inner vertices than in that of the outer vertices. As shown in Fig 4, the inner vertices (black dots) of scalar filed grid have more fluid particles around them than outer vertices (red dots), which leads to far more kernel evaluations (Eq 2).

It can also be observed that our method has certain advantages in the efficiency of the triangulation stage and the utilization of memory. This is because in the valid cell identification stage, each surface vertex corresponds to an MC grid cell, which is named the surface cell. Note that not all surface cells have the chance to generate triangles. As mentioned above (Sect 3.4), one of the requirements is that all eight corner vertices of this cell should be surface vertices. The MC grid cell corresponding to the inner vertex does not produce triangular patches; that is, this cell is invalid. However, in Yang's approach[15], the inner vertices are all supposed to be surface vertices. Therefore, they generate about 30% idle threads. Moreover, when compacting surface vertices, the storage space occupied by these inner vertices is also redundant. The experiments show that our method saves nearly 10% of the memory consumption.

In addition to comparing with Yang *et al.*[15], in order to illustrate the advance of our pipeline in terms of efficiency, we also made comparisons with Akinci *et al.* [9]. In the Akinci method, it also does not remove inner vertices of scalar filed grid. Apart from this difference, there are also differences in stages of the detection of surface particles and the compaction of surface vertices and valid cells. In the Akinci method, the smooth color filed [19] is employed to identify surface particles. The compaction of surface vertices and valid cells and the later triangulation runs on the CPU to avoid race conditions. On the contrary, our method is purely parallel, and all calculations including triangulation are performed on GPU via employ

the *Parallel Compact Memory Scheme* described in Sect 3.2. In table 2, the experiments show that in triangulation stage, compared with the Akinci method, the speedup of this stage can be up to 60, because the FLOPS (float operation per second) of GPU is much stronger than CPU in the case of without race conditions. However, since most of the computation time is occupied by the scalar filed estimation, the overall speed up is limited by the scalar filed estimation, thus the overall speed up is not improved much. In scalar filed estimation stage, in addition to the speedup caused by removing inner vertices of scalar field grid, the speedup in this stage is also slightly improved. The reason is that in Akinci method, it needs to transfer the array that determines whether a vertex is surface vertex from GPU to CPU and compact it into a continuous array at CPU, and then copy the continuous array back to GPU. The back and forth data transmission results in extra time consumption that can not be ignored.

The comparisons experiments show that the significance and superiority of our work. Most of the computation in surface reconstruction is spent on constructing the scalar filed, whereas the triangulation is rather efficient, thus it is worth considering how to reasonably reduce the amount of calculation in the scalar filed estimation. As for Wu *et al.*[14] method, their focus is on saving memory consumption via complex data structure not on improving efficiency. The method itself does not also achieve complete parallelism and there are atomic operations greatly affecting parallel efficiency. More importantly, they also do not remove inner vertices of scalar filed grid. Therefore, in terms of efficiency, our approach is absolutely faster than it.

5. Conclusions

In this paper, we present techniques to improve the performance of high-quality surface reconstruction for particle-based fluids. Primarily, we present a reliable method to eliminate scalar flied grid inner vertices inside the fluid volume. Therefore, robust criteria are proposed for ensuring the success of surface reconstruction. Our presented pipeline is completely parallel to avoid race conditions. The comparison experiments with the state-of-the-art method indicate that our approach obtains a significant performance improvement. The performance improvement is mainly due to avoiding unnecessary scalar field computations and data storage for scalar field grid vertices that lie inside the fluid volume. In the experiments, although the number of vertices is only reduced by approximately 30%, the time consumption of scalar field estimation is reduced by nearly 50%. Compared with outer vertices, the inner vertices have many more neighborhood fluid particles and thus involve many more kernel evaluations.

References

[1] Koschier, D, Bender, J, Solenthaler, B, Teschner, M. Smoothed Particle Hydrodynamics Techniques for the Physics Based Simulation of Fluids and Solids. In: Jakob, W, Puppo, E, editors. Eurographics 2019 - Tutorials. The Eurographics Association; 2019,doi:10.2312/egt.20191035.

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

[3] Adams, B, Pauly, M, Keiser, R, Guibas, LJ. Adaptively sampled particle fluids. In: ACM SIGGRAPH 2007 Papers. SIGGRAPH '07; New York, NY, USA: Association for Computing Machinery; 2007, p. 48–es. doi:10.1145/1275808.1276437.

[4] Solenthaler, B, Schläfli, J, Pajarola, R. A unified particle model for fluid–solid interactions. Computer Animation and Virtual Worlds 2007;18(1):69–82. doi:10.1002/cav.162.

[5] Onderik, J, Chládek, M, Ďurikovič, R. Sph with small scale details and improved surface reconstruction. In: Proceedings of the 27th Spring Conference on Computer Graphics. SCCG '11; New York, NY, USA: Association for Computing Machinery; 2011, p. 29–36. doi:10.1145/2461217.2461224.

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

[7] Bhattacharya, H, Gao, Y, Bargteil, AW. A level-set method for skinning animated particle data. IEEE transactions on visualization and computer graphics 2014;21(3):315–327. doi:10.1109/TVCG.2014.2362546.

[8] Lorensen, WE, Cline, HE. Marching cubes: A high resolution 3d surface construction algorithm. ACM siggraph computer graphics 1987;21(4):163–169.

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

[10] Manson, J, Schaefer, S. Isosurfaces over simplicial partitions of multiresolution grids. Comput Graph Forum 2010;29:377–385.

[11] Zhou, K, Gong, M, Huang, X, Guo, B. Data-parallel octrees for surface reconstruction. IEEE transactions on visualization and computer graphics 2011;17(5):669–681. doi:10.1109/TVCG.2010.75.

[12] Akinci, G, Akinci, N, Ihmsen, M, Teschner, M. An Efficient Surface Reconstruction Pipeline for Particle-Based Fluids. In: Bender, J, Kuijper, A, Fellner, DW, Guerin, E, editors. Workshop on Virtual Reality Interaction and Physical Simulation. The Eurographics Association; 2012,doi:10.2312/PE/vriphys/vriphys12/061-068.

[13] Canezin, F, Guennebaud, G, Barthe, L. Topology-aware neighborhoods for point-based simulation and reconstruction. In: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation. SCA '16; Goslar, DEU: Eurographics Association; 2016, p. 37–47.

[14] Wu, W, Li, H, Su, T, Liu, H, Lv, Z. Gpu-accelerated sph fluids surface reconstruction using two-level spatial uniform grids. The Visual Computer 2017;33(11):1429–1442. doi:10.1007/s00371-016-1289-x.

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

[16] Wang, X, Ban, X, Liu, X, Zhang, Y, Wang, L. Efficient extracting surfaces approach employing anisotropic kernels for sph fluids. Journal of Visualization 2016;19(2):301–317. doi:10.1007/s12650-015-0317-7.

[17] Wang, X, Ban, X, Zhang, Y, Pan, Z, Liu, S. Anisotropic surface reconstruction for multiphase fluids. In: 2017 International Conference on Cyberworlds (CW); vol. 1. Los Alamitos, CA, USA: IEEE Computer Society; 2017, p. 118–125. doi:10.1109/CW.2017.30.

[18] Lee, J, Kim, JH, Lee, HY, Kim, SJ. Realistic fluid representation by anisotropic particle. Journal of visualization 2019;22(2):313–320. doi:10.1007/s12650-018-0535-x.

[19] 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. SCA '03; Goslar, DEU: Eurographics Association; 2003, p. 154–159.

[20] Koren, Y, Carmel, L. Visualization of labeled data using linear transformations. Proceedings of the Ninth annual IEEE conference on Information visualizationOctober 2003; Seattle, Washington: IEEE Computer Society; 2003, p. 121–128. doi:10.1109/INFVIS.2003.1249017.

[21] Kazhdan, M, Hoppe, H. Screened poisson surface reconstruction. ACM Trans Graph 2013;32(3). doi:10.1145/2487228.2487237.

[22] Rogers, , B., , D., , Garcia-Feal, , O., , Gomez-Gesteira, , et al. Dualsphysics: Open-source parallel cfd solver based on smoothed particle hydrodynamics (sph). Computer physics communications 2015;187:204–216. doi:https://doi.org/10.1016/j.cpc.2014.10.004.

Table 1: General information of the presented scenarios in surface reconstruction.

			cell size = r					cell size = $r/2$				
Scene	$\#particles$	$\frac{\#particles_{surf}}{\#particles_{total}}$	$t_{total}(s)$	$\frac{\#vertices_{surf}}{\#vertices_{total}}$	$\frac{\#mccells_{valid}}{\#mccells_{total}}$	$\#triangles$	grid res.	$t_{total}(s)$	$\frac{\#vertices_{surf}}{\#vertices_{total}}$	$\frac{\#mccells_{valid}}{\#mccells_{total}}$	$\#triangles$	grid res.
DBD	69.0k	0.235	3.43	0.049	0.008	189k	$201 * 201 * 301$	16.9	0.043	0.004	774k	$403 * 403 * 603$
DB	1.02m	0.193	6.42	0.085	0.013	213k	$324 * 144 * 174$	28.5	0.078	0.007	864k	$648 * 288 * 348$
MK	2.21m	0.166	8.53	0.073	0.012	414k	$256 * 256 * 256$	41.2	0.065	0.006	1.69m	$512 * 512 * 512$
EF	1.88m	0.229	10.3	0.089	0.015	501k	$256 * 256 * 256$	60.5	0.081	0.008	2.04m	$512 * 512 * 512$
WC	2.74m	0.094	11.3	0.018	0.003	344k	$376 * 376 * 426$	69.6	0.015	0.001	1.39m	$752 * 752 * 852$

Table 2: Comparison of the average reconstruction times between Akinci *et al.*[9], Yang *et al.*[15] and our method

		cell size = r					cell size = $r/2$				
Scene	Method	$\frac{\#vertices_{surf}}{\#vertices_{total}}$	$t_{sf}(s)$	$t_{tri}(s)$	t_{com}/t_{our}	$mem(GB)$	$\frac{\#vertices_{surf}}{\#vertices_{total}}$	$t_{sf}(s)$	$t_{tri}(s)$	t_{com}/t_{our}	$mem(GB)$
DDB	**Our**	**0.049**	**2.34**	**0.018**	-	**0.594**	**0.043**	**14.4**	**0.110**	-	**1.01**
	Yang	0.065	4.67	0.022	1.99	0.653	0.067	31.7	0.138	2.20	1.18
	Akinci	0.070	5.78	0.139	2.51	0.483	0.073	35.6	2.53	2.62	0.83
DB	**Our**	**0.085**	**4.37**	**0.020**	-	**0.730**	**0.078**	**24.8**	**0.115**	-	**1.10**
	Yang	0.116	9.94	0.042	2.27	0.897	0.117	61.7	0.300	2.48	1.44
	Akinci	0.120	12.1	0.307	2.83	0.647	0.121	70.1	7.73	3.12	0.911
MK	**Our**	**0.073**	**5.34**	**0.039**	-	**0.998**	**0.065**	**33.8**	**0.237**	-	**2.39**
	Yang	0.106	10.7	0.048	2.00	1.03	0.108	77.9	0.312	2.31	2.50
	Akinci	0.110	15.6	0.344	2.96	0.906	0.111	99.1	9.31	3.18	2.17
EF	**Our**	**0.089**	**7.71**	**0.048**	-	**0.888**	**0.081**	**47.2**	**0.280**	-	**1.66**
	Yang	0.112	14.7	0.056	1.90	1.03	0.113	100.6	0.344	2.13	2.06
	Akinci	0.117	17.8	0.410	2.38	0.791	0.118	118.1	10.3	2.70	1.42
WC	**Our**	**0.018**	**7.4**	**0.060**	-	**1.33**	**0.015**	**45.2**	**0.356**	-	**3.97**
	Yang	0.024	18.3	0.076	2.47	1.38	0.025	117.2	0.494	2.59	4.10
	Akinci	0.027	25.4	0.580	3.25	1.18	0.029	155.8	14.0	3.72	3.70

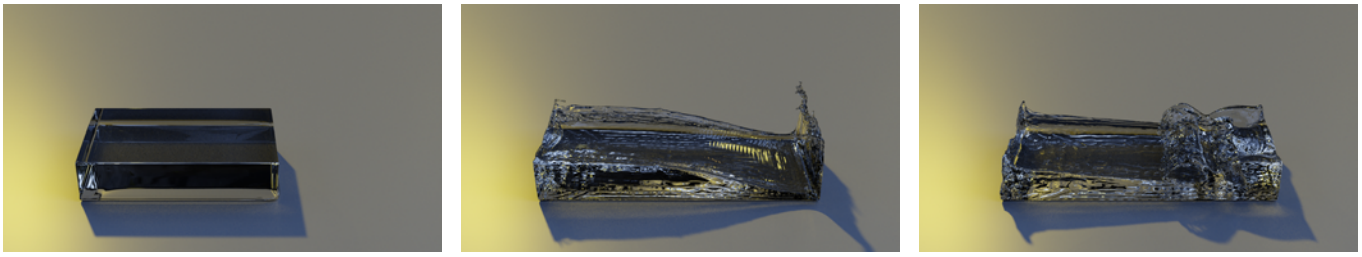

Fig. 13: Breaking Dam scene

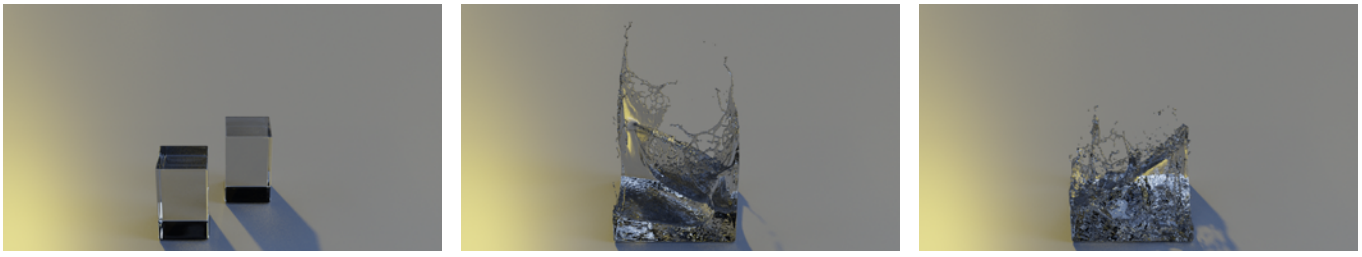

Fig. 14: Double Breaking Dam scene

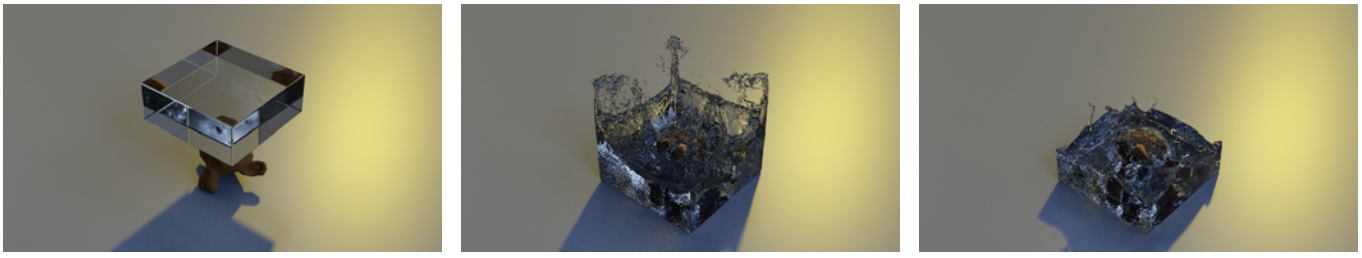

Fig. 15: Water Pouring scene

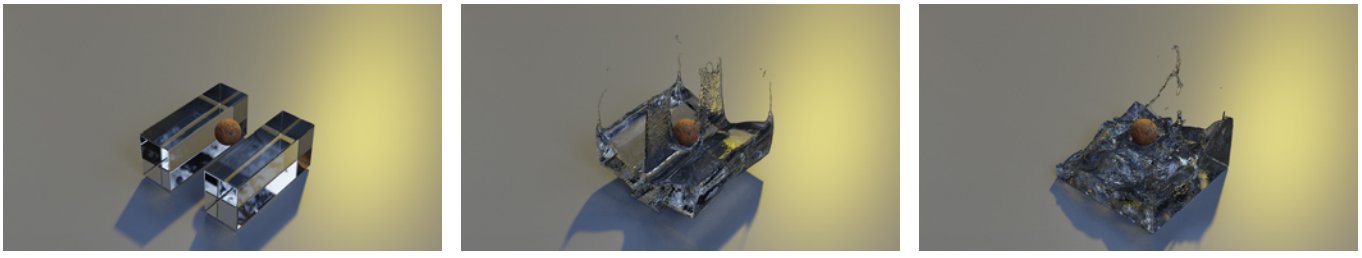

Fig. 16: External Force scene

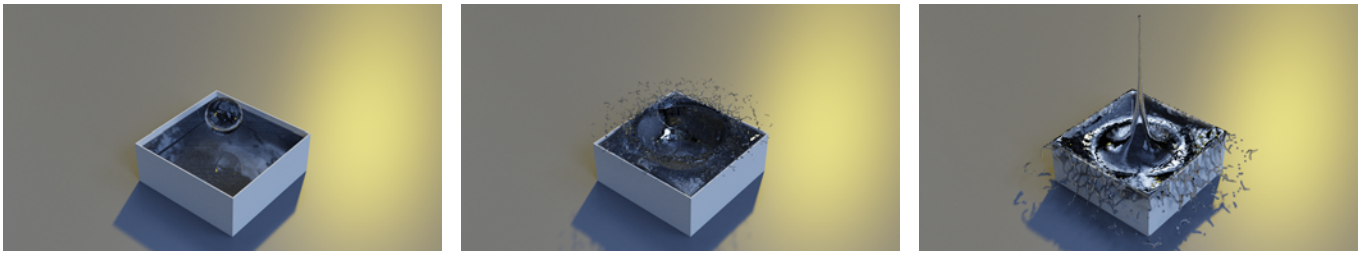

Fig. 17: Water Crown scene