Workshop on Virtual Reality Interaction and Physical Simulation VRIPHYS (2012)
J. Bender, A. Kuijper, D. W. Fellner, and É. Guérin (Editors)

An Efficient Surface Reconstruction Pipeline for Particle-Based Fluids

Gizem Akinci   Nadir Akinci   Markus Ihmsen   Matthias Teschner

University of Freiburg

Abstract

In this paper we present an efficient surface reconstruction pipeline for particle-based fluids such as smoothed particle hydrodynamics. After the scalar field computation and the marching cubes based triangulation, we post process the surface mesh by applying surface decimation and subdivision algorithms. In comparison to existing approaches, the decimation step alleviates the particle alignment related bumpiness very efficiently and reduces the number of triangles in flat regions. Later, the subdivision step ensures that the non-smooth regions are smoothed in a performance friendly way which allows our approach to run significantly faster by using lower resolution marching cubes grids. The presented pipeline is applicable to particle position data sets in a frame by frame basis. Throughout the paper, we present both visual and performance comparisons with different parameter settings, and with a state-of-the-art surface reconstruction technique. Our results demonstrate that in comparison to other approaches with comparable surface quality, our pipeline runs 15 to 20 times faster with up to 80% less memory and secondary storage consumption.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

1. Introduction

Particle-based fluid simulation techniques are used in a broad range of applications, e.g. scientific research areas, feature films and video games. As the popularity of these techniques has increased in the last few years, simulation scenarios have been becoming more diverse and more complex.

Surface reconstruction from particle sets is a well-studied problem in the literature. However, it still remains as one of the main bottlenecks in production pipelines due to the large computational time and memory requirements. Marching cubes (MC) [LC87] based polygonization is one of the most commonly used techniques due to its simplicity, and there exist various ways to compute the scalar field for the underlying grid, e.g. [ZB05, APKG07, SSP07, YT10, OCD11]. Akinci et al. [AIAT12] showed that the mesh quality, computation time and memory consumption are strongly influenced by two parameters: the cell size of the MC grid and the influence radius which indicates the maximum distance of particles contributing to the scalar field at MC grid vertices.

It is known that surfaces generated for particle-based fluids usually suffer from bumpiness due to irregular particle placement. Bump-free surfaces with smooth features can be generated using smoothing operations, e.g. [YT10, BGB11] or by using very small cell sizes and large influence radii, e.g. [AIAT12]. However, most of these approaches either trade off quality with efficiency, or they overlook the amount of detail that the surface provides, which means that the generated surface covers the particles only roughly.

Our contribution. In this paper, we focus on the efficient reconstruction and processing of surfaces of particle data sets to achieve not only smooth, but also detailed surfaces. For this aim, we propose to combine the existing scalar field computation approaches, in particular [SSP07], with two post-processing steps: decimation and subdivision. This is motivated by an improved representation of smaller surface details, reduction of bumps in flat regions, reduced overall computation time for the surface reconstruction and reduced memory consumption for the resulting mesh. Fig. 1 shows that our approach is able to produce high quality fluid surfaces up to 20 times faster than other approaches

DOI: 10.2312/PE/vriphys/vriphys12/061-068

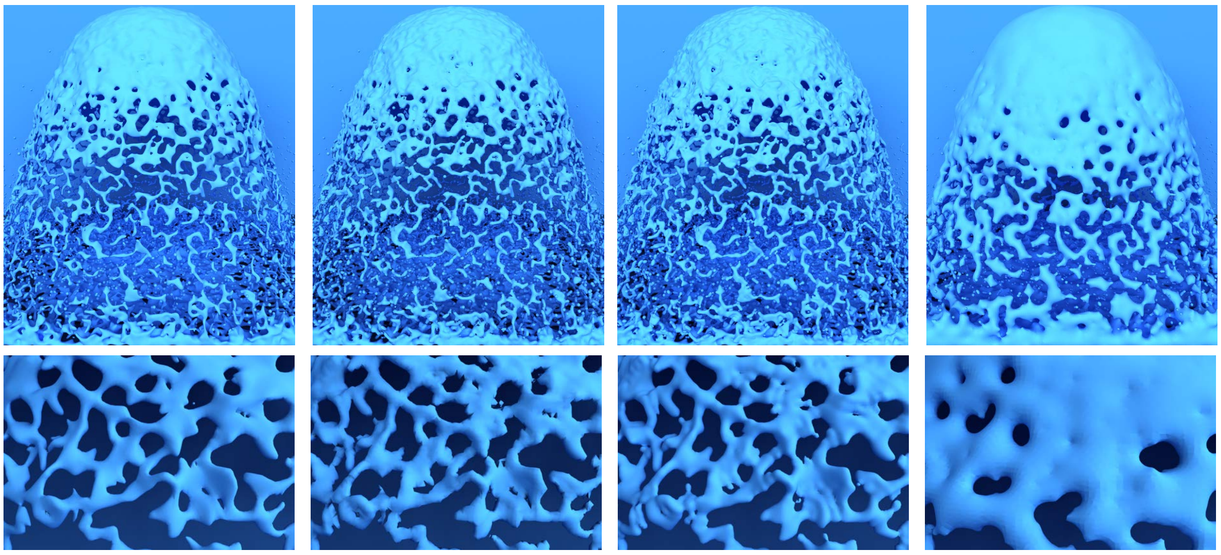

Figure 1: *The Fountain scene with up to 100k particles. mcs and r denote the MC cell size and the particle radius, respectively. (a) The result of our pipeline. We post process the surface mesh that is reconstructed by using [SSP07] with mcs = r. (b) [SSP07] with mcs = r without applying post processing. (c) [SSP07] with mcs = r/2 without applying post processing. (d) [BGB11] where the marching tetrahedra grid resolution is the same as we employ. The average surface reconstruction time per frame for each approach is: 12.75, 8.5, 200 and 240 seconds; while the average number of generated triangles are: 480k, 600k, 2.3m and 1.75m, respectively. In comparison to (b), our post processing pipeline increases the quality of the surface with only a small computational overhead, while being even more efficient in terms of the number of generated triangles. In comparison to other two approaches with comparable quality, our approach runs 15 to 20 times faster with up to 80% improved memory and secondary storage consumption.*

with comparable surface quality, while being very efficient in terms of memory and secondary storage consumption.

We address the bumpiness problem by applying a feature sensitive mesh decimation algorithm. This step allows the decimated mesh to remain faithful to the original topology of the reconstructed surface, while it looks smooth in flat regions. However, the decimation step might sharpen the edges of some mesh features (see Fig. 2, middle). So as to regain the smoothness of such sharp features, we apply subdivision to the decimated mesh. The resolution of the decimated mesh is already adaptive in a way that flat regions are sampled with larger and less triangles, while high curvature regions are sampled with smaller and more triangles. After the subdivision, those high curvature regions with fine details are sampled with even more triangles which are distributed smoothly. Therefore, even though the number of triangles after this step is less than the number of triangles of the input mesh, we achieve a triangle mesh that is even smoother than the input mesh (see Fig. 2, right). Since the subdivision step ensures the smoothness of mesh features, we gain a significant performance by using lower resolution marching cubes grids (see Fig. 1, (a) and (c) for the comparison). Furthermore, we handle the isolated particles as pre-tessellated spheres. By extracting them from the main surface computations, the MC grid shrinks, the performance increases and the memory consumption decreases.

2. Related Work

In this work, we polygonize fluid surfaces using MC [LC87]. Although there are variants of MC, e.g., marching tetrahedra [CP98, TPG98] or marching triangles [HI97], MC is the commonly preferred technique due to its simplicity and efficiency.

There exist various approaches that address proper scalar field computation for fluid particles. Within the context of these approaches, Zhu and Bridson [ZB05] presented a signed distance field computation approach where the scalar values of MC grid vertices are computed by considering the contribution of neighboring particles. While this technique improves the classical bumpy appearance of former methods, e.g. blobbies [Bli82], it suffers from artifacts in concave regions. This problem is addressed by Solenthaler et al. [SSP07] and Onderik et al. [OCD11]. Adams et al. [APKG07] presented a distance-based surface tracking technique, which is particularly suitable for adaptively

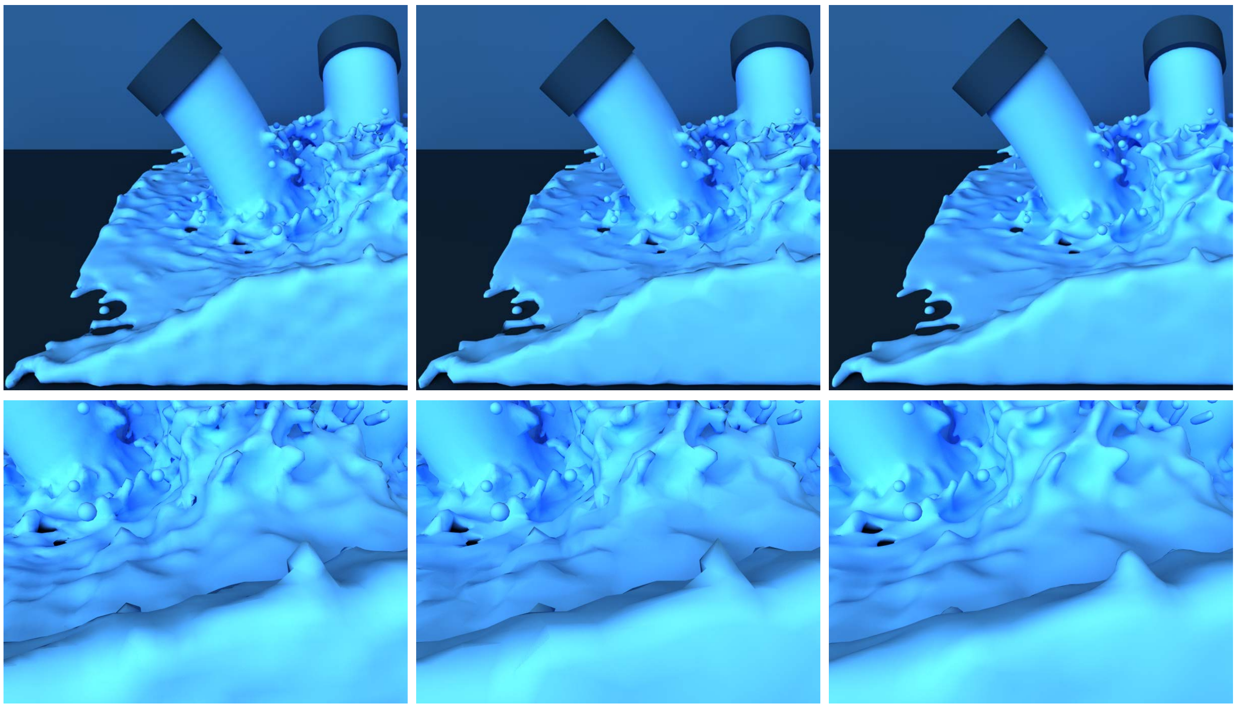

Figure 2: *The Tap scene with up to 60k particles. The surface reconstruction is the first step of the pipeline (left). The decimation step reduces bumps in flat regions while it remains faithful to features of input mesh (middle). Finally, undesired sharp features are smoothed by the subdivision step (right). Particles without neighbors are rendered as pretessellated spheres in all images.*

sampled particle sets, e.g. [APKG07]. An alternative surface reconstruction method was proposed by Williams in [Wil08]. This method considers the surface reconstruction as a constraint optimization problem and is useful for obtaining smooth surfaces. However, the method has temporal coherency problems that cause the surface to look oscillating in animation sequences. Later, Yu and Turk [YT10] proposed an anisotropic kernel approach, and addressed the bumpiness problem by using a variant of a Laplacian smoothing to alleviate the effect of irregular particle placement. This method improves the bumpy appearance of fluid surfaces, however, it is computationally expensive when compared to [ZB05, SSP07, OCD11]. Inspired by [Wil08], Bhattacharya et al. [BGB11] proposed a level set method, where the fluid surface that lies between inner and outer surface approximations is processed by Laplacian smoothing. The main downside of this method is, the performed surface approximations and smoothing steps cause the surface to cover the underlying particles coarsely, which loses the details of the particle set (see Fig. 1, (d) and Fig. 3, (d)).

In fluid animation, surface reconstruction has usually been a bottleneck due to its large memory consumption and high computation time. These issues have been addressed by various researchers. Sparse block grids [Bri03] and dynamic tubular grids [NM06] have been proposed in order to target the computational effort to the narrow-band area, where the surface is actually defined. Later, Nielsen et al. [NNSM07] have presented an out-of-core technique so as to handle large resolutions. Recently, Akinci et al. [AIAT12] proposed a parallel method where the scalar field is efficiently constructed only in the narrow-band.

Within the context of mesh decimation, there exist various approaches to mention, e.g. vertex decimation technique by Schroeder et al. [SZL92], vertex clustering technique by Rossignac and Borrel [RB93], and a rich research on edge contraction techniques, e.g. [HDD*93, Gue95, GH97]. Among the edge contraction techniques, the method of Garland and Heckbert [GH97] has particularly gained attention where their method associates error quadrics to mesh vertices, and computes edge contraction costs accordingly. Error quadric of each vertex is computed as the sum of the squared distances of the vertex to the planes of triangles that meet at the vertex. This method has been further improved in [GH98, Hop99] for associated vertex attributes. In our pipeline, our aim is to reduce bumps in flat regions without affecting the features that characterize the mesh. Therefore, we need a method that distinguishes curvature regions from low amplitude features (e.g. bumps) in flat regions. Many of the aforementioned decimation methods trade off efficiency with quality. Therefore, we prefer to use the quadric error metric technique of Garland and Heckbert since the method is able to simplify even very complex models rapidly, and it maintains high fidelity to the original mesh while reducing bumps effectively in flat regions. After the decimation step, undesired sharp edges may occur in the surface mesh. In order to smooth those sharp features, we suggest to subdivide the decimated mesh at the final step. Doo-Sabin [DS78], Catmull-Clark [CC78], Loop [Loo87] and butterfly [DLG90] are among the most commonly employed subdivision techniques in the field. In our experiments, we use Loop's subdivision scheme since it is easy to implement, and it produces smooth surfaces in only few subdivision steps. There are only few researchers who apply decimation or subdivision on fluid surfaces for improving the mesh quality [TFK*03, BBB10]. The aim of our paper is to present a pipeline with integrated decimation and subdivision, which focuses on the reconstruction of high quality fluid surfaces while ensuring computation time and memory consumption efficiency.

3. The Pipeline

In this section, we describe a simple yet efficient pipeline for the surface reconstruction of particle-based fluids. Details of the scalar field computation and triangulation are discussed in Sec. 3.1. Later, decimation and subdivision steps are explained in Sec. 3.2 and Sec. 3.3, respectively. Results are further improved by extracting isolated particles and handling them differently, which is explained in Sec. 3.4.

3.1. Scalar Field Computation and Triangulation

As stated previously, any of the aforementioned scalar field computation techniques can be used in our pipeline. In our experiments, however, we prefer [SSP07] over other techniques, since the method removes the artifacts that arise in concave regions efficiently and covers the underlying particles faithfully.

In [SSP07], the isosurface of the scalar field around a grid point $\mathbf{v}$ is defined as

$$\phi(\mathbf{v}) = |\mathbf{v} - \bar{\mathbf{v}}| - rf. \tag{1}$$

In Eq. 1, r denotes the radius of particles which is computed as the half of the particles' equilibrium distance. The weighted average of the nearby particle positions $\bar{\mathbf{v}}$ is computed as

$$\bar{\mathbf{v}} = \frac{\sum_i \mathbf{x}_i k(|\mathbf{v} - \mathbf{x}_i|/R)}{\sum_i k(\mathbf{v} - \mathbf{x}_i|/R)}, \tag{2}$$

where R denotes the influence radius, i is the contributing particles that reside within distance R and k is the kernel function where $k(s) = \max(0, (1 - s^2)^3)$. Further, f is a factor to handle the potential artifacts in concave regions, which

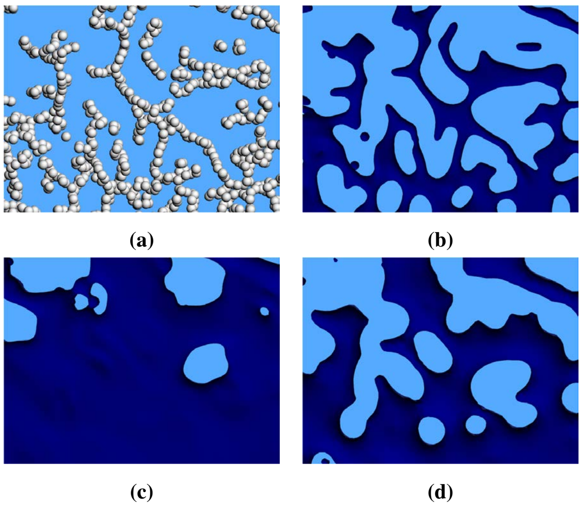

Figure 3: *A close-up of the corner breaking dam (CBD) scene. (a) The underlying particle set. (b) By using an influence radius $R = 4r$, we cover the underlying particle set as tightly as possible. (c) Doubled influence radius causes the surface to cover the particle set coarsely. (d) Similar to (c), with the method of Bhattacharya et al. [BGB11], the surface is roughly covered and the details of the underlying particle set are not visible.*

is computed as

$$f = \begin{cases} 1 & EV_{max} < t_{low} \\ \gamma^3 - 3\gamma^2 + 3\gamma & otherwise \end{cases} \tag{3}$$

with $\gamma = \frac{t_{high} - EV_{max}}{t_{high} - t_{low}}$, where t_{high} and t_{low} are user defined threshold values. Here, EV_{max} denotes the largest eigenvalue of $\nabla_{\mathbf{v}}(\bar{\mathbf{v}})$.

Surface fitting. The generation of smooth and bump free surfaces has been one of the main concerns in the field, e.g. [BGB11, AIAT12]. However, the point that has been usually overlooked is the amount of detail that the surface provides, which can be achieved by covering the underlying particle set as good as possible. Therefore, the employed scalar field computation technique should fit the particle set faithfully. Besides, a proper setup of the influence radius significantly affects the quality. In order to show this effect, we experimented with two different values of R in our pipeline. Fig. 3, (b) shows that with $R = 4r$, the underlying particle set is covered properly. However, when we double R (see Fig. 3, (c)), the particle set is covered very roughly. Such a large influence radius alleviates the bumpiness problem by smoothing out surface details which is an undesired result. For a fair comparison, both settings were tested within our pipeline. However, in terms of bumpiness problem, post processing steps do not improve the results of the large influence radius since the small scale surface details are already eliminated

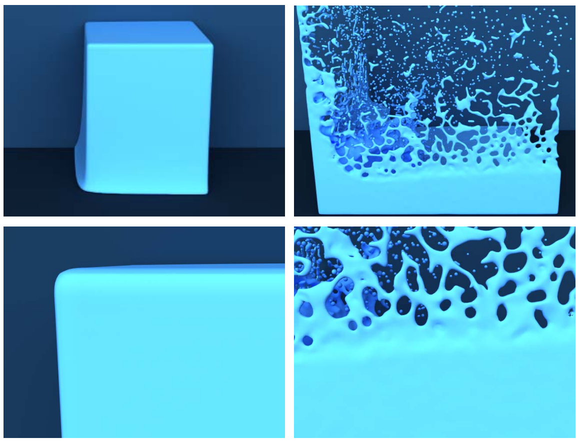

Figure 4: *The corner breaking dam (CBD) scene with 130k particles. Close-up views in the bottom row show that our pipeline produces perfectly smooth and bump-free surfaces in flat regions.*

by the initial surface reconstruction step. Similar to the results of this setting, the method of Bhattacharya et al. also covers the particle set coarsely and the surface details are again lost (see Fig. 3, (d)). Despite the fact that better coverage of underlying particles increases the bumpiness problem, our pipeline ensures that the bumps in flat regions are alleviated (see Fig. 4). In addition, proper fitting of the surface also prevents temporal coherency problems which are especially visible in the coarsely covered surfaces. This problem can be seen in the accompanying video, where we show the results for the CBD scene with both influence radii that are discussed above.

3.2. Decimation

The aim of this step is to reduce bumps in flat regions while maintaining fidelity to the original mesh. Therefore, we need a method that distinguishes curvature regions from low amplitude features (e.g. bumps) in flat regions. Quadric error metric technique [GH97, GH98, Hop99] is a suitable choice for our aim, since the distance-to-original-mesh based decimation priority works properly in our case. Besides, the method is able to simplify even very complex surfaces rapidly.

Quadric error metric based decimation technique iteratively contracts vertex pairs. In other words, two relatively close vertices, v_1 and v_2, are moved to a new position $\bar{v}$, these two vertices' incident edges are connected to $\bar{v}$, and v_1 and v_2 are deleted at the end. According to the algorithm, contraction costs are computed in order to select the correct vertex pair for contraction. For this aim, a symmetric 4x4 matrix $\mathbf{Q}$ is associated with each vertex in the beginning; and this matrix is used to define the error at the given vertex position $\mathbf{v}$ as $\mathbf{v}^T\mathbf{Q}\mathbf{v}$. The matrix $\bar{\mathbf{Q}}$ that needs to be associated with the

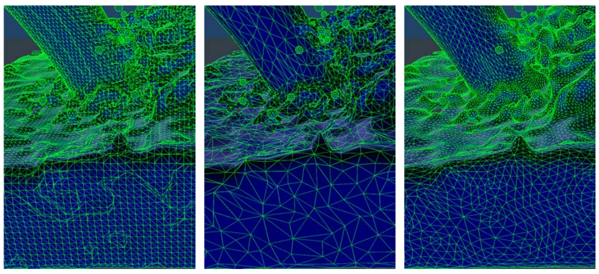

Figure 5: *The resolution of the reconstructed mesh (left) becomes adaptive after decimation (middle) and preserves its adaptivity after subdivision (right). Note that the mesh size is reduced without affecting the topology of the mesh; however, the result is smoother than the original mesh. The isolated particles are reconstructed as pre-tessellated spheres in all cases.*

new vertex $\bar{v}$ can be computed with a simple additive rule as $\bar{\mathbf{Q}} = \mathbf{Q}_1 + \mathbf{Q}_2$. The error $\bar{\mathbf{v}}^T\bar{\mathbf{Q}}\bar{\mathbf{v}}$ of this target vertex becomes the cost of contracting the given pair. While contracting the pair v_1 and v_2, $\bar{v}$ is chosen along the line segment v_1v_2 so that it has the minimal cost. After the computation of contraction costs, the pairs are stored in a heap which is keyed based on the cost. The pair with the minimum cost is kept at the top of the heap, processed first, and the cost of the pairs involving v_1 and v_2 are updated accordingly. On curvature regions which characterize the surface mesh, the distance between $\bar{v}$ and the original mesh is large enough which increases the collapse cost. However, $\bar{v}$ on flat or lower amplitude regions (e.g. bumps) produces less collapse cost, which pushes them on top of the heap.

A sequence of pair contractions are applied until the simplification goal is satisfied. In our experiments, we reduce the total number of triangles by 80%. After this step, the mesh resolution becomes adaptive in a way that the details in high curvature regions are still preserved, while the bumpiness in relatively flat regions is significantly alleviated by using less and larger triangles in those regions (see Fig. 5, middle).

3.3. Subdivision

The decimation step alleviates the bumpiness problem, together with an effective reduction in the mesh size. However, this step causes the edges of some mesh features to sharpen, and the mesh smoothness is not fully preserved. These sharp edges can cause flickering artifacts during rendering, especially on raytraced transparent surfaces. In order to resolve this problem, we apply one more post processing step which subdivides the surface mesh, and improves the non-smooth parts efficiently.

For the subdivision, we employ Loop's subdivision scheme [Loo87] since the method is easy to implement, and it is able to produce smooth surfaces very efficiently. In the first pass of this scheme, new vertices are added at the midpoints of the edges of the triangles. In the second pass, new edges are added to complete the subdivision into four new triangles. In the final pass, Loop's mask is applied. In other words, vertex positions are averaged so that each vertex in the triangle mesh has a new position which is computed based on the number and relative positions of its neighbors. At the end of this step, a smoothing effect is achieved.

According to our experiments, the reduction in the original mesh size is usually around 20% after the decimation and subdivision. However, due to the fact that the adaptive mesh is sampled with more and evenly distributed triangles in high curvature and detailed regions (see Fig. 5, right), we achieve even smoother surfaces when compared to the input mesh (see Fig. 2, left and right), and still ensure that the main memory and the secondary storage are consumed efficiently. In addition, since the subdivision step ensures the smoothness of mesh features, we do not require a higher resolution of the MC grid. This results in a significant performance speed up.

3.4. Isolated Particle Extraction

The MC approach can require very large grids, if the simulation causes the particles to splash too distant from each other. Such scenarios are especially inefficient in terms of memory consumption. To ameliorate this issue, we perform a simple yet efficient step. In our pipeline, we consider a particle as isolated, if it has no neighbors. We determine such particles in the beginning, and exclude them from the steps mentioned in the above sections. After creating one sphere with radius r initially, we transform this sphere to all positions which belong to isolated particles. A sphere can be tessellated with triangles in an icosahedron form, and can be subdivided until a user defined threshold. This smooth approximation also prevents rendering artifacts which are especially visible on raytraced transparent surfaces of isolated particles (see Fig. 6). The cornered edges that cause such rendering artifacts occur due to insufficient MC grid resolutions and cause serious flickering artifacts in the animation. So as to generate smooth, spherical shapes for the isolated particles, very high resolution MC grids are required. However, this results in very large computation times and memory consumptions, while the tessellation of such particles might still be insufficient (see Fig. 6, middle).

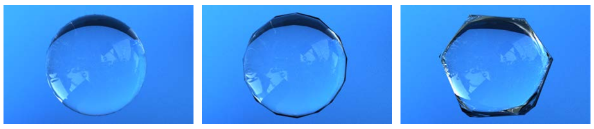
Figure 6: *Transparent rendering of an isolated particle using a pre-tessellated sphere (left), and MC triangulation with $mcs = r/2$ (middle) and $mcs = r$ (right).*

Scene	Tap	Fountain	CBD	Ship
$\text{time}_{\text{sf-t}}$	2.4	8.5	6	66
time_{dec}	0.77	3.5	1.5	18
$\text{time}_{\text{subdiv}}$	0.1	0.75	0.35	3.6
$\text{time}_{\text{total}}$	3.27	12.75	7.85	87.6

Table 1: *Average per frame timings in seconds for each scene. $time_{sf\text{-}t}$ denotes the computation time for scalar field and triangulation; while $time_{dec}$ and $time_{subdiv}$ stand for the computation time of decimation and subdivision, respectively.*

4. Implementation Details

For the fluid simulation, we employed the PCISPH method of Solenthaler et al. [SP09] and for the Ship scene, we used the two-way coupling method of Akinci et al. [AIA*12]. The ship model is courtesy of www.thefree3dmodels.com. For an efficient neighborhood search over particles, we prefer the compact hashing method proposed by Ihmsen et al. [IABT11]. All the scenes were rendered using mental ray v3.9.4 [NVI].

5. Results

In this section, we discuss the results of our pipeline in terms of visual quality and performance on four different scenes: Fountain, Tap, CBD and Ship. Experiments have been performed on an Intel Xeon X5680 CPU with 24GB RAM. A detailed computation time analysis of the scenes is shown in Tab. 1.

Our first scene is the Fountain with up to 100k fluid particles (see Fig.1). Using this scene, we firstly compare the result of our pipeline with [SSP07], which is reconstructed using the MC cell size of $r/2$. Neither decimation nor subdivision is applied to the latter. Fig. 1, (a) and (c) show that our pipeline produces comparable results even without using such a high resolution grid. While our pipeline computes one frame within 12.5 seconds on average, this time increases to 200 seconds for the latter. Besides, the average number of generated triangles are 480k and 2.3m for each approach, respectively. High resolution MC grids ensure the smoothness of mesh features. However, bumpiness problem remains (see Fig.1, (c)). Secondly, we compare our result with a more recent surface reconstruction method [BGB11]. Our motivation for applying [BGB11] is that, the results of this method are already very smooth and bumpiness is alleviated without applying any post processing steps. However, as illustrated in Fig. 1, (d), the particle data set is not exactly covered but smoothed out due to the surface approximations and the smoothing steps of the algorithm. Besides, the average surface reconstruction time per frame of this approach is 240 seconds, which is almost 20 times slower in comparison to our pipeline. In addition, the number of generated trian-

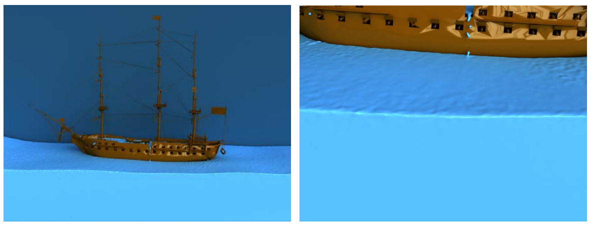

Figure 7: *The Ship scene with 4 million particles.*

gles is 1.75m, which again shows the memory and secondary storage consumption efficiency of our method.

Our second experiment is the Tap scene where the fluid is poured from three taps as shown in Fig. 2. This is the smallest test scene with up to 60k particles. Before the post processing steps, the fluid covers the underlying particle set tightly, however, it looks bumpy in flat regions (Fig. 2, left). Note that after the whole pipeline is applied, the result is less bumpy on sides and inflow, and it is smoother, but still detailed even for such a low particle resolution (Fig. 2, right).

Our next scene is a corner breaking dam (CBD) with 130k particles (see Fig. 4). In order to test the effect of the influence radius R on the surface quality, we experimented with two different parameter settings on this scene as shown in the accompanying video. Using $R = 4r$, the computation takes 7.85 seconds per frame; while it takes 58 seconds for the setting with doubled influence radius. The computation time increases for the latter, since a larger influence radius causes to traverse over more particles in the neighborhood of the grid vertex. Another disadvantage of the doubled influence radius is that, the particle set is coarsely covered and the surface details are lost, which leads to temporal coherency artifacts.

Finally, we present our largest scene, the Ship (see Fig. 7), which was simulated with 4 million fluid particles. Our results show that even for such a large scale simulation, the average computation time is reasonable with 87.6 seconds per frame.

6. Conclusion and Future Work

Our method post processes the fluid surfaces so that the particle alignment related and undesired bumps in flat regions are removed, while the characteristic features of the mesh are maintained. After the subdivision step, we obtain a surface whose features are even smoother than the input mesh. Our parameter setup and choice of the employed methods allows us to preserve the surface details, and finally to have a high quality surface. In comparison to other methods with comparable surface quality, our pipeline runs up to 20 times faster with 80% less memory and secondary storage consumption.

The decimation threshold we prefer (80%) may not be optimal in all cases. The reason is that some of the frames allow the mesh to be decimated more than 80% if the surface is perfectly flat or if the high curvature regions are in minority when compared to flat regions. The opposite situation holds if high curvature regions are in majority in the surface. In such a case, using a value smaller than 80% would give better results. However, the threshold we use has been the best choice in all of our test scenes, since larger or smaller values were not appropriate for achieving the same quality in most of the frames. One of the next steps to improve our method can be using an adaptive threshold which depends on the average curvature of the surface in every frame.

After post processing steps, it is expected to have temporal incoherence in subsequent frames. Even though we did not observe such a behavior in our test scenes, the effect of the post processing steps in temporal coherence can be analyzed in the future.

Although we exclude the isolated particles from the main computational steps, the MC grid can still remain very large depending on the frame, and this causes a large memory footprint when performing surface reconstruction. In the future, we would like to incorporate hashing for the used grid vertices instead of using uniform grid structures.

Another direction for future work can be parallelizing our code and focusing on a GPU implementation.

7. Acknowledgements

We thank the reviewers for their helpful comments. This project is supported by the German Research Foundation (DFG) under contract numbers SFB/TR-8 and TE 632/1-2. We also thank NVIDIA ARC GmbH for supporting this work.

References

[AIA*12] Akinci N., Ihmsen M., Akinci G., Solenthaler B., Teschner M.: Versatile Rigid-Fluid Coupling for Incompressible SPH. In *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* (2012). 6

[AIAT12] Akinci G., Ihmsen M., Akinci N., Teschner M.: Parallel Surface Reconstruction for Particle-Based Fluids. *Computer Graphics Forum* (2012). 1, 3, 4

[APKG07] Adams B., Pauly M., Keiser R., Guibas L.: Adaptively sampled particle fluids. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers* (New York, NY, USA, 2007), ACM Press, p. 48. doi:http://doi.acm.org/10.1145/1275808.1276437. 1, 2, 3

[BBB10] Brochu T., Batty C., Bridson R.: Matching fluid simulation elements to surface geometry and topology. In *ACM Transaction on Graphics* (New York, NY, USA, 2010), vol. 29, ACM, pp. 1–9. doi:http://doi.acm.org/10.1145/1778765.1778784. 4

[BGB11] Bhattacharya H., Gao Y., Bargteil A. W.: A Level-set Method for Skinning Animated Particle Data. In *roceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)* (2011). 1, 2, 3, 4, 6

[Bli82] BLINN J.: A Generalization of Algebraic Surface Drawing. *ACM Trans. Graph. 1*, 3 (1982), 235–256. doi:http://doi.acm.org/10.1145/357306.357310. 2

[Bri03] BRIDSON R.: *Computational aspects of dynamic surfaces*. PhD thesis, Stanford University, 2003. 3

[CC78] CATMULL E., CLARK J.: Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-Aided Design 10*, 6 (November 1978), 350–355. 4

[CP98] CHAN S. L., PURISIMA E. O.: A new tetrahedral tesselation scheme for isosurface generation. In *Computers and Graphics* (1998), 22(1), pp. 83–90. 2

[DLG90] DYN N., LEVIN D., GREGORY J. A.: A Butterfly Subdivision Scheme for Surface Interpolation with Tension Control. *ACM T 9*, 2 (April 1990), 160–169. 4

[DS78] DOO D., SABIN M.: Behavior of recursive division surfaces near extraordinary points. *Computer-Aided Design 10*, 6 (1978), 356–360. 4

[GH97] GARLAND M., HECKBERT P.: Surface simplification using quadric error metrics. In *SIGGRAPH Š97 Proceedings* (1997), pp. 209–216. 3, 5

[GH98] GARLAND M., HECKBERT P.: Simplifying surfaces with color and texture using quadric error metrics. In *In Visualization Š98 Proceedings (1998), IEEE* (1998), pp. 263–269. 3, 5

[Gue95] GUEZIEC A.: Surface simplification with variable tolerance. In *In Second Annual Intl. Symp. on Medical Robotics and Computer Assisted Surgery (MRCAS '95)* (November 1995), pp. 132–139. 3

[HDD*93] HOPPE H., DEROSE T., DUCHAMP T., MCDONALD J., STUETZLE W.: Mesh optimization. In *In SIGGRAPH '93 Proc.* (August 1993), pp. 19–26. URL: http://www.research.microsoft.com/research/graphics/hoppe. 3

[HI97] HILTON A., ILLINGWORTH J.: *Marching Triangles: Delaunay Implicit Surface Triangulation*. Tech. rep., University of Surrey, 1997. 2

[Hop99] HOPPE H.: New quadric metric for simplifying meshes with appearance attributes. In *Proceedings of the 10th IEEE Visualization 1999 Conference (VIS '99)* (Washington, DC, USA, 1999). 3, 5

[IABT11] IHMSEN M., AKINCI N., BECKER M., TESCHNER M.: A Parallel SPH Implementation on Multi-Core CPUs. In *Computer Graphics Forum* (2011), vol. 30, pp. 99–112. doi:10.1111/j.1467-8659.2010.01832. 6

[LC87] LORENSEN W., CLINE H.: Marching cubes: A high resolution 3D surface construction algorithm. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1987), ACM Press, pp. 163–169. doi:http://doi.acm.org/10.1145/37401.37422. 1, 2

[Loo87] LOOP C.: *Smooth Subdivision surfaces based on triangles*. Master's thesis, University of Utah, 1987. 4, 5

[NM06] NIELSEN M. B., MUSETH K.: Dynamic Tubular Grid: An efficient data structure and algorithms for high resolution level sets. *J. Scient. Comput. 26, 3* (2006), 261–299. 3

[NNSM07] NIELSEN M. B., NILSSON O., SÖDERSTRÖM A., MUSETH K.: Out-of-core and compressed level set methods. ACM Transactions on Graphics, Vol. 26, No.4, 2007. 3

[NVI] NVIDIA: Nvidia arc, 2011. mental ray 3.9 [software]. URL: http://www.mentalimages.com/products/mental-ray/aboutmental-ray.html. 6

[OCD11] ONDERIK J., CHLADEK M., DURIKOVIC R.: SPH with Small Scale Details and Improved Surface Reconstruction. In *In SCCG 2011: Proceedings of the Spring Conference on Computer graphics* (ACM Press, New York, USA, 2011). 1, 2, 3

[RB93] ROSSIGNAC J., BORREL P.: Multi-resolution 3D approximations for rendering complex scenes. In *Modeling in Computer Graphics: Methods and Application* (1993), pp. 455–465. 3

[SP09] SOLENTHALER B., PAJAROLA R.: Predictive-corrective incompressible SPH. In *SIGGRAPH '09: ACM SIGGRAPH 2009 Papers* (New York, NY, USA, 2009), ACM, pp. 1–6. 6

[SSP07] SOLENTHALER B., SCHLÄFLI J., PAJAROLA R.: A unified particle model for fluid-solid interactions. *Computer Animation and Virtual Worlds 18*, 1 (2007), 69–82. 1, 2, 3, 4, 6

[SZL92] SCHROEDER W. J., ZARGE J. A., LORENSEN W. E.: Decimation of triangle meshes. In *Computer Graphics(SIGGRAPH '92 Proc.)* (1992), vol. 26(2), pp. 65–70. 3

[TFK*03] TAKAHASHI T., FUJII H., KUNIMATSU A., HIWADA K., SAITO T., TANAKA K., UEKI H.: Realistic Animation of Fluid with Splash and Foam. *Computer Graphics Forum 22*, 3 (2003), 391–400. 4

[TPG98] TREECE G. M., PRAGER R. W., GEE A. H.: Regularised Marching Tetrahedra: Improved Iso-Surface Extraction. In *Computers and Graphics* (1998), 23, pp. 583–598. 2

[Wil08] WILLIAMS B.: *Fluid surface reconstruction from particles*. Master's thesis, University Of British Columbia, 2008. 3

[YT10] YU J., TURK G.: Reconstructing Surfaces of Particle-Based Fluids Using Anisotropic Kernels. In *SCA '10: Proceedings of the 2010 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, 2010), Eurographics Association, pp. 217–225. 1, 3

[ZB05] ZHU Y., BRIDSON R.: Animating sand as a fluid. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers* (New York, NY, USA, 2005), ACM Press, pp. 965–972. doi:http://doi.acm.org/10.1145/1186822.1073298. 1, 2, 3