Marching Correctors – Fast and Precise Polygonal Isosurfaces of SPH Data

Benjamin Schindler, Raphael Fuchs
Scientific Visualization Group
ETH Zürich, Switzerland
bschindler,raphael@inf.ethz.ch

Jürgen Waser
VRVis Vienna, Austria
jwaser@vrvis.at

Ronald Peikert
Scientific Visualization Group
ETH Zürich Switzerland
peikert@inf.ethz.ch

***Abstract*—This paper presents the first method for isosurface extraction from smoothed particle hydrodynamics (SPH) data that is exact with respect to the functional representation provided by SPH. The *Marching Correctors* algorithm is an extension of the *Marching Cubes* algorithm which is adapted to the SPH representation and avoids resampling to a full grid. The algorithm operates on a virtual grid of sufficiently high resolution to faithfully reconstruct the fields represented by the SPH data. The virtual grid is efficient in terms of both memory usage and computing time, because cells are only materialized and processed if they are either seed cells or intersected by the isosurface. Besides the virtual grid, a key idea of our algorithm is to add a correction step to the isosurface vertices. An evaluation of the algorithm in terms of accuracy and performance is given based on three SPH datasets. By comparing with [1] on similarly sized data a performance gain of almost two orders of magnitude was achieved. Moreover, it is demonstrated how the correction step effectively reduces the typical artifacts produced by the Marching Cubes method.**

I. Introduction

The extraction and rendering of isosurfaces is one of the basic visualization methods for scalar fields. There exists a huge body of literature on isosurface extraction with specializations for the different types of data discretization. Most of the existing methods deal with data organized in either hexahedral or tetrahedral cells, and only a small fraction addresses meshless data, that is data given on a set of points. In contrast to this, point-based representation on the output side has been treated more extensively, because many isosurface extraction methods generate points more immediately or more naturally than their connectivity. But even for meshless data, polygonal isosurfaces have a number of advantages, since for visualization purposes it is often required to compute additional properties such as volume, curvature or surface area of the extracted surfaces [2]. An example of four isosurfaces from SPH data can bee seen in Fig. 1.

SPH data sets are functionally represented scalar and vector fields which are given on a set of particles. Each particle is represented by the field values and a radially symmetric kernel. As opposed to other functional representations, SPH has kernels with relatively large support radii which means that in order to reconstruct the field at a given location, in the order of a hundred neighbors have to be evaluated.

While reconstruction of the field is comparably expensive, the SPH representation offers the advantage of high-quality gradients which can be computed at relatively little extra cost. We make use of these gradients for optimizing the intersection points generated by the standard Marching Cubes algorithm.

Our contributions are as follows:

- We present the first method for isosurface extraction from SPH datasets where all extracted vertices are guaranteed to be placed at positions where the field assumes the selected iso-value.
- We suggest a virtual grid to structure the processing and guarantee that the resulting mesh is water-tight without requiring resampling on all positions of a full grid. The virtual grid is never stored in memory in its entirety.
- We describe a trimming method at the free surface to get a consistent isosurface boundary.
- The selection of seed cells is the most expensive part of the Marching Correctors algorithm and it is computed on the GPU.
- Vertex normals are computed directly from the SPH representation instead of using a gradient estimation scheme.

In the next section we give some background on isosurface extraction with a focus on SPH data and point-sampled data. In Section III we give an overview of the Marching Correctors algorithm. An evaluation of performance and accuracy is given in Section IV followed by conclusion and future directions in Section V.

II. Related Work

a) Visualization of SPH data: There exist a few visualization packages for SPH data. SPLASH [3] is capable of producing 2D plots of data by projecting particles onto a plane, and 3D plots by integrating the kernel contributions of all particles intersecting a ray through the view pixel. In addition to these image-space methods, there are a few object-space methods available such as streamline plots. ParaView [4] supports SPH data through its meshless extension described in [5]. It has functionality for resampling SPH data onto planes, grids and arbitrary geometric meshes. This resampled data can then be used with grid-based visualization algorithms to generate isosurfaces, integral lines and surfaces, etc. However,

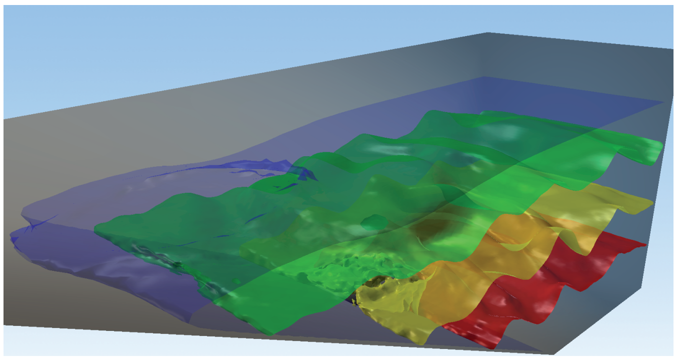

Fig. 1: Three pressure isosurfaces and the free surface of the tsunami data at time step 194, $p = 1000$ (red), 4000 (yellow), 10000 (green).

resampling is expensive, and it does not provide exact gradients.

b) Isosurface Extraction: Together with direct volume rendering [6], isosurfaces are still one of the methods of choice for visualizing scalar fields. Historically, isosurfaces were obtained by generating contours on a set of planar sections and stitching them together [7]. Direct isosurface extraction from volumetric data started with Lorensen and Cline's seminal paper [8] and subsequent modifications for topologically consistent isosurfaces [9], [10]. Optimized methods based on octrees [11], interval trees [12], [13] or extrema graphs [14] avoid the processing of cells not intersected by the isosurface. Several efficient methods use the span-space [15]–[17] for fast and space-efficient retrieval of the set of all intersected cells. Isosurface extraction from time-dependent data requires temporal extensions of the acceleration data structures. Recent methods are based on persistent octrees [18], [19] or on a temporally extended span-space [20]. A 2006 survey of Marching-Cubes methods was given by Newman and Yi [21].

While the problem of point-based isosurface extraction from gridded data has been studied extensively [22]–[24], there is relatively little work on point-based isosurface extraction from point-sampled data. Co and Joy presented an isosurface algorithm for point-sampled data [25] where a set of point splats is generated and rendered by a splatting technique [26]. Rosenberg and Birdwell [27] presented an isosurface method for SPH data. It is restricted to the special case of extracting the free surface. An accurate method for reconstructing free surfaces from SPH data has recently been proposed by Marrone et al. [28]. Rosenthal and Linsen [29] used a similar approach, where the field is reconstructed only on a set of edges between neighbor samples. In two later papers, a level set technique is used for smoothing the isosurface [1] and the smoothing process is combined with the isosurface extraction into a PDE [30]. A follow up [31] improves the performance of their approach by evaluating the level-set function only in a narrow band around the zero-level set.

A main difference between our work and the work of Rosenthal and Linsen is that they deal with point-sampled data, not with functionally represented (kernel-based) data. Also, their level-set approach has the goal of optimizing the smoothness of the isosurface. In contrast, for SPH data the field values can be reconstructed by evaluating all kernels that overlap the query location. It makes no difference whether the query location is a particle position or not, therefore SPH data are not point-sampled data. Since the reconstruction can be done anywhere within the fluid, SPH is an implicit representation of the *exact* isosurface. The goal of an SPH isosurface algorithm must therefore be to generate samples of the exact isosurface and without an extra smoothing step.

III. The Marching Correctors Algorithm

The extraction of the isosurface is performed by processing cells of the virtual grid. The grid exists only virtually and is never stored in its entirety. The algorithm consists of an initialization phase and two main parts as can be seen in Fig. 2. During initialization, an acceleration structure is created for efficient SPH interpolation (see Section III-A). In the first step (see Section III-C), seed cells in the virtual grid are selected. In the second step (see Section III-D), the processing starts at the seed cells and creates the surface geometry.

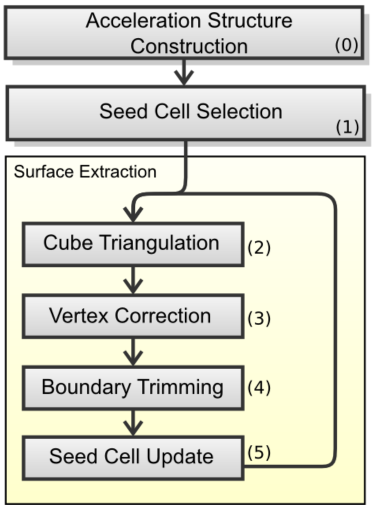

Fig. 2: Algorithm overview. (0) An acceleration structure is computed in a preprocessing step. (1) The scan for seed cell selection is performed on the GPU. (2) The first seed cube in the list is picked and triangulated. (3) The triangle vertices are corrected along the cube edges. (4) Resulting triangles at the boundary are trimmed. (5) The seed cell list is updated.

A. Spatial Data Structure

For efficient SPH evaluation a spatial data structure is needed because of the required neighborhood lookup. In our implementation, we chose to use a regular grid where every particle is only stored in the cell where its center lies. The spacing is chosen to be $2 \cdot \max\{h_i \,|i = 1..N\}$. As a result, a neighborhood lookup has to consider 27 regular grid cells. The factor 2 results from the SPH kernels which usually have an extent of $2h$. A regular grid performs well, since in current engineering applications, particle size is nearly constant. For other types of SPH simulation, a kd-tree can be more performant, although much more expensive to build. This step can be seen as a form of preprocessing and does not have to be repeated when other isosurface levels are requested.

B. Virtual Grid

The presented algorithm is based on a virtual grid consisting of uniform cubic cells. This grid, which is not to be confused with the regular grid used for particle search (as explained in III-A), is never stored in memory entirely but only represented by means of extent, dimension and cube size. The size of a cube is set to $c \cdot \min\{h_i \,|i = 1..N\}$ where $c = 0.5$ per default. The spatial sampling rate given by the particles in our two example datasets was about $0.55h$, $0.65h$ (dam and tsunami) and h (vortex rings). Reasonable values for c are therefore between 0.5 and 1.0. With values higher than 1.0 the topology of the isosurface can start to become simplified. By choosing c, a trade-off is made between level-of-detail and computation time.

A major issue when dealing with SPH is that it is very expensive to interpolate a value. When using Marching Cubes it is normal that a value at a given position is requested multiple times – up to eight times per grid vertex. A lot of performance can therefore be gained by caching values for later reuse. One benefit of the virtual grid is that there is no direct cost associated with it in terms of memory consumption as the grid is only implicitly available. It is therefore not an option to use a full grid for the cache. Instead we resort to a slice based approach, similar to the one used in [27]. As we will see, it is sufficient to keep three slices at once in memory as the cubes are calculated more or less in order (see also Section III-D). Since the set of seed cells is not the complete set of intersected cells, it can happen that a grid point which is in a slice already deleted needs to be evaluated. Such cache misses of course affect performance of the method. At the cost of increased memory consumption, one can increase the number of slices which are kept in memory at once. However, in the three examples, and for all time steps used to create the videos, a cache of three slices was sufficient to get a near-optimal cache hit ratio.

C. Seed Cell Selection

The purpose of this step is to find cells of the virtual grid that are effectively intersected by the isosurface. Selecting seed cells that are not intersected by the isosurface is not a problem, but their number should be small in order to avoid unnecessary computation. False negatives, or missed cells that do intersect the isosurface, are also not a problem as long as there is at least one seed cell selected per connected component of the isosurface. For performance reasons it is desirable to have as many (positive) seed cells as possible to improve the cache-hit ratio. This is in contrast to other work [32] where the objective is a minimal set of seed cells.

The basic idea of the seed cell selection is to find pairs of neighboring particles which have data values below and above the isosurface level. For this we compare each particle with all particles within a radius of h. Once such a particle pair is found, we have to make sure that a cell of the virtual grid which contains the isovalue is selected. By the intermediate value theorem this has happen at least once on the straight line connecting the two points. Therefore, all cells intersected by the straight line between the two particles are added the list of seed cells. This is done using a Bresenham-style iteration.

A performance issue of the SPH representation is now that the evaluation of smoothed data is quite expensive because it requires finding a large set of neighbors, computing distances to them, and evaluating kernel functions. To make seed cell selection fast, we use the data specified at the particle positions. This avoids evaluating kernel functions. SPH kernels have the partition of unity property, which means that smoothed data do not deviate much from the raw data specified at the particle positions. More importantly, by using raw data we still do not miss any small connected component: small components are near local minima or maxima of the scalar field. In the case of a maximum, the smoothed scalar field cannot assume values larger than the largest raw data value that contributes to the weighted sum This means that taking raw instead of smoothed data expands the data range in the vicinity of a local minimum or maximum, which can lead to unnecessary seed cells (false positives), but not to missed isosurface components.

The process of selecting seed cells is very well suited for parallelization, and has been implemented on the GPU using CUDA [33]. The resulting seed cell indices are transferred to the CPU and are stored in a `set` container (from the C++ standard template library), which is internally a tree structure. The `set` container by design cannot store the same key multiple times, so all the duplicate cells are eliminated automatically. The indices are sorted in the tree using a lexicographic ordering of the cube index. This facilitates caching (see III-B) where both lexicographic ordering of the virtual grid indices and efficient insertion are required.

D. Surface Extraction

Given the set of seed cells, the actual extraction of the surface is performed. From a broad view, this process is similar to the Marching Cubes scheme with modifications described in the following subsections.

1) Marching Cubes Table Lookup: For each seed cell, a standard Marching Cubes step is done using a publicly available lookup table by Bourke [34]. This produces a set of consistently oriented triangles that correctly matches triangles generated in adjacent cells. The Marching Cubes method generates vertices by linearly interpolating data on the edges of the cell. A lookup in a second table is made to retrieve the subset of neighbor cells where the isosurface continues. All these cells are added to a queue of cells provided that they have not been visited before. This simple queue mechanism makes sure that only those cells have to be processed which are either seed cells or which effectively contribute to the isosurface. It was already used by Wyvill et al [35] in their isosurface method predating Marching Cubes.

2) Correction Step: Since the SPH representation allows for an accurate evaluation at arbitrary points in the domain, we can apply *correction steps* to the vertices of the triangle mesh. These vertices lie on edges of a cube within the virtual grid, the two end points of which have a pair of values above and below the isosurface level. According to the intermediate value theorem, there must be a point on the edge where the field exactly equals the level. The idea is now to correct the vertex found by linear interpolation toward this point. This is done by evaluating the directional derivative of the field with respect to the coordinate that varies along the edge. The directional derivative is one of the three components of the SPH gradient and it can be computed efficiently together with the field value. With this derivative, a number of Newton-Raphson steps are done, making sure that corrected points stay within the extent of the edge. A small number of iterations leads to points lying on the actual SPH isosurface with high precision.

3) Boundary Trimming: In contrast to grid based data, SPH data has no concise definition of the boundary. Also, even if a concise definition was available, it would not be aligned with the virtual grid used by the Marching Correctors. Solid boundaries, or walls, are sometimes available as an extra set of “solid particles”, or they are just geometrically specified. Free boundaries have to be detected from the set of (fluid) particles, e.g. by the algorithm of Marrone et al. [28]. For an approximation of the free surface, the sum of weights $\sum w_i W(\mathbf{x} - \mathbf{x_i}, h_i)$ can be used and checked against a threshold [27]. In this work, we use this simple technique for detection of all boundaries. However, our framework would also allow for an implementation of Marrone’s method which basically amounts to computing a derived field on grid nodes near the boundary and taking its zero level set. The purpose of boundary detection is that we have to trim the isosurface mesh at the boundary. This is to make sure that all vertices of the mesh not only satisfy the isosurface equation, but are also lying within the fluid volume.

There are two thresholds used, the *node threshold* for the virtual cube vertices and the *vertex threshold* for the isosurface vertices. The node threshold is set to a very low value such as 0.1. Cells of the virtual grid where the sum of weights falls below this threshold at a single vertex are discarded. The vertex threshold (we suggest 0.5) is used for trimming triangles that have at one or two of its vertices a sum of weights below this threshold. Trimming is done as is shown in Fig. 3 and it can result in a quad which has to be split into two triangles.

4) SPH Surface Normal Evaluation: Normal vectors are required for the rendering of a surface with a local lighting model. Accurate surface normals are therefore essential for a correct perception of the isosurface. A standard method is to estimate the normal at a surface vertex by a weighted average of the normals of the adjacent triangles. However, due to

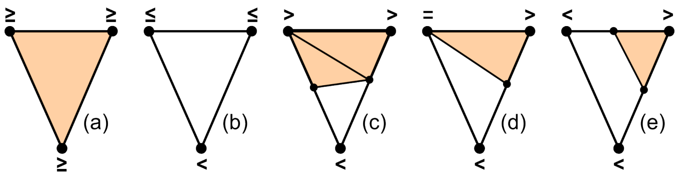

Fig. 3: Different trimming cases. The relation symbols indicate the sum of weights relative to the vertex threshold. If the sum of weights is above or equal to the threshold at all vertices, the triangle is kept (a). Otherwise, if the sum of weights is below or equal at all vertices the triangle is discarded (b). It is trimmed in the three remaining cases (c-e).

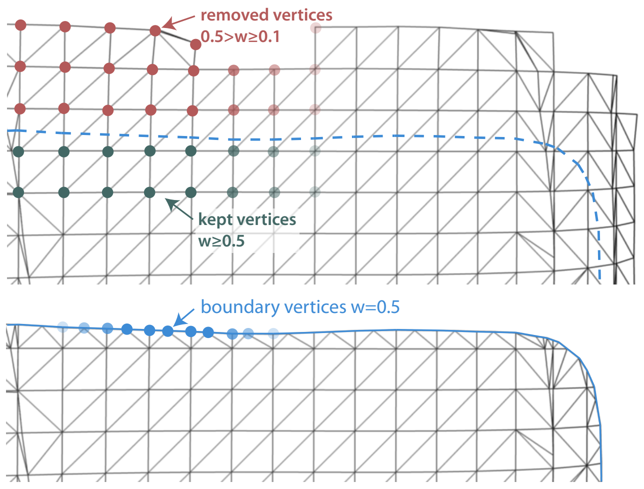

Fig. 4: Comparison of original (top) and trimmed (bottom) wireframe.

the irregularly shaped triangles in Marching Cubes meshes, this simple approach produces poor normals which show as artifacts in the rendered surface. Better isosurface normals are obtained by using the underlying scalar field and estimating its gradient at the vertices of the isosurface. In the special case of SPH data, high-quality gradients can be obtained directly at arbitrary points. Since the calculation of the SPH gradient is more expensive than a simple interpolation, these high-quality normals are not computed in the preview quality mode where mesh vertices are not corrected.

IV. Results

The algorithm has been tested on three datasets, the tsunami dataset from [36], the dam breaking simulation [37], and the vortex rings [38] dataset.

There are two parameters in the algorithm which have a large impact on the performance. First, there is the ability to adapt the virtual grid size. It can be increased from the default $0.5h$ for preview purposes to get better performance. Also, depending on the smoothness of the data, a low value here leads to an unnecessary high triangle count. The second parameter is a scaling factor for the smoothing length h of the SPH data. It has been shown [39] that shrinking the kernel size can give a huge speedup with only a small loss of quality. This will reduce the quality of the SPH interpolation but also make it faster as the particle overlap is reduced. For preview purposes this loss of quality is very often acceptable.

All timings were performed on a 2x Xeon E5430 with 64 gigabytes of RAM running Linux. The GPU used is a Nvidia GTX 295. Even though the GTX 295 is equipped with two GPUs, only one of them was used and the memory transfers to and from the GPU are included in the timings. Also, the CPU portion of the code only uses a single core.

A. *Tsunami Data Set*

The first test dataset is a SPH simulation of the creation of a tsunami [36]. In the simulation, a wedge is sliding downward which is idealizing a landslide. The simulation contains 249 time steps and 1.2 million particles and uses the quadratic kernel [36] with a constant smoothing length of $h = 0.056$.

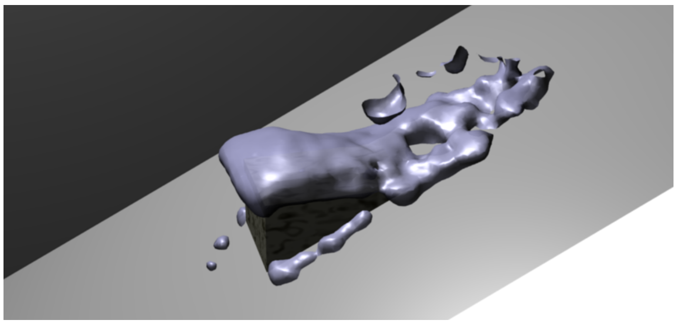
Fig. 6: Vorticity magnitude isosurface in the tsunami data (time step=150, vorticity=5).

B. *Dam Break Data Set*

Our second test data set is an SPH simulation of the SPHERIC dam breaking case [37]. It has 87 time steps, each with roughly 670,000 fluid particles using the *cubic spline* kernel [40]. Solid boundaries are modeled with solid particles, while the air contains no particles. In contrast to the tsunami data set where the smoothing length is fixed for every particle, the dam break data has a variable smoothing length. When varying the kernel size as described in Section IV-D, the smoothing radius of every particle is multiplied by a constant factor.

C. *Vortex Rings Data Set*

The third dataset is a simulation of the collision of two vortex rings at circulation-based Reynolds number equal to 1600 using a vortex method [38]. The initial conditions are set to reproduce the experimental results by [41]. It contains about 10,000,000 particles and uses the M'4 kernel [42] with a radius of $h = 0.0005$. There are no free surfaces in this data.

In the work of Rosenthal et al. [31] they write that it takes about 6 minutes to extract a surface of 47k vertices from a dataset of 8 million particles. Comparing this to our performance in Table I, we find that our method is about 80 times faster, extracting about 150k vertices from a dataset of 10 million particles.

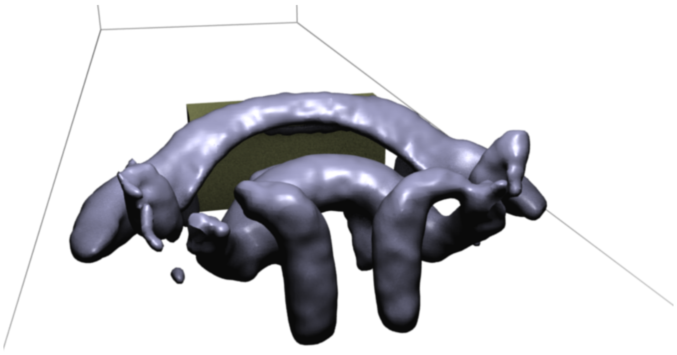
Fig. 7: λ_2 isosurface in the dam break data (time step=30, $\lambda_2 = -150$).

D. *SPH Kernel Size*

In SPH simulations the number of particles overlapping a position within the data is usually very large – on average about 70 for the dam break data and over 200 for the tsunami data. This overlap can be reduced by shrinking the kernel size by a constant factor. Doing so can result in an enormous speedup without sacrificing too much accuracy, which for a preview is certainly sufficient. Next to the original kernel size, we selected the scaling factors 0.75 and 0.5 for testing.

In case of the tsunami dataset, setting the scaling factor to 0.75 reduces the extraction time by almost 40%, as shown in Table II. At a scaling factor of 0.5 the performance is doubled, and the visual quality did not suffer visibly as seen in Fig. 5.

With the dam break data the performance impact of kernel size reduction is similar as seen in Table III. In case of a noisy surface as its the case in the pressure isosurface of the dam break data, a factor of 0.5 can introduce very small isosurface components close to the boundary, where the SPH interpolation quality suffers even more, see Fig. 9. Otherwise, the surface was extracted with almost no visual errors.

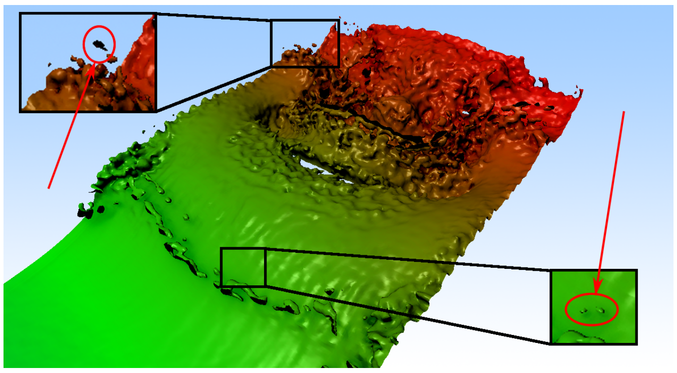
Fig. 9: Effect of excessive kernel scaling in noisy datasets. In the two close-ups the isosurface is replaced by one generated with scaled (0.5) kernel, revealing artifacts.

E. *Virtual Grid Size*

Depending on the resolution of the simulation and the smoothness of the data, a grid as fine as $0.5h$ is not needed.

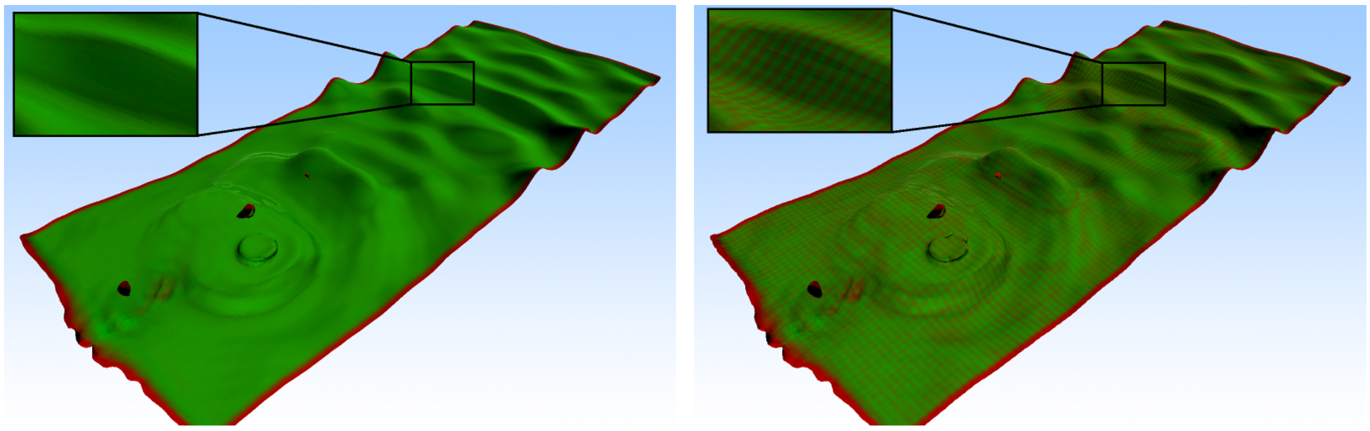

Fig. 5: Pressure isosurface (time step=194, p=3000) in the tsunami data with a virtual grid size of $0.5h = 0.028$ without (a) or with (b) the effect of scaling down the smoothing length by a factor of 0.5. The surface is colored by the sum of the kernels, where green means a value of 1 and red a value of 0.5 or less. Even with the reduced kernel size, the algorithm manages to reproduce the large surface without visible defects and to capture high-frequency details, such as the "splashes" in the front part of the isosurface.

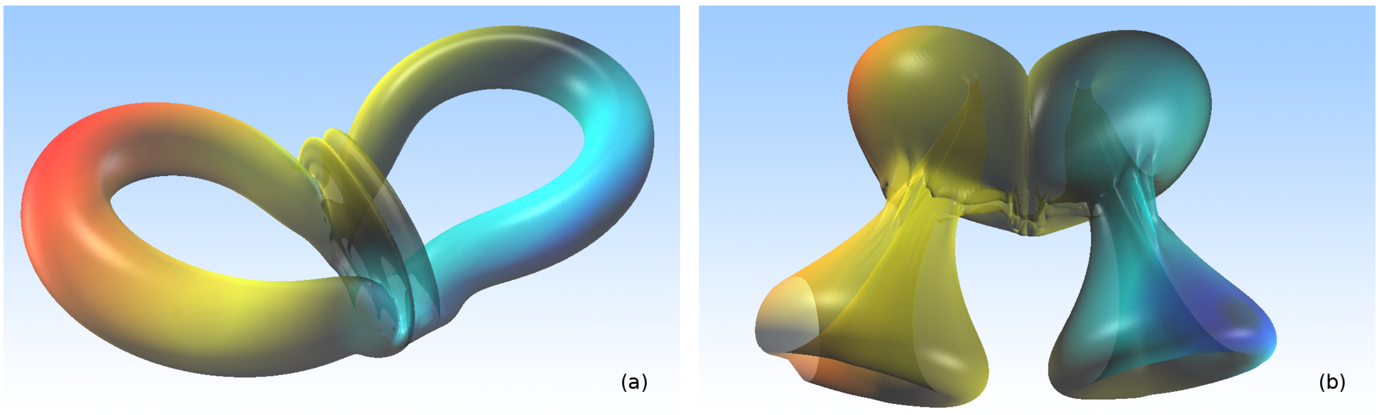

Fig. 8: Transparent rendering of the vorticity isosurface with $\omega = 10$ (a) and $\omega = 0.1$ (b).

In Table IV the grid statistics with two different grid sizes for the tsunami dataset is shown. Even though the number of grid cells is shrunk by a factor of eight, all other numbers are more or less decreased by a factor of four which is to be expected because of the 2D nature of the surface. The decrease in the number of seed cells is not as dramatic since seed cell selection works better with larger grid cells.

The effectiveness of the candidate selection can be observed in Tables IV. The technique selected about 3% of all possible virtual cells, approximately half of them were used for the isosurface. The ratio could be improved a lot by using linear interpolation instead of the Bresenham algorithm in the seed selection. This way however, we cannot guarantee the evaluation of all positive seed cells.

In terms of runtime, the time needed for the surface extraction process is also cut roughly by a factor of four. When shrinking kernel size, the effect is less dramatic because the percentage of time used for data structure management such as candidate lists, cache handling etc. starts to increase.

F. Effectiveness of the Correction Step

In Table V the distribution of the sizes of correction steps is analyzed. It shows that correction steps are mostly in the order of a few percent of the edge length, but can reach up to a third of the edge length. Enabling the correction step roughly doubles the total time for surface extraction as shown in Tables II and III. However, if either a coarser grid is used or the smoothing length is scaled down, this overhead diminishes rapidly. The gain in visual quality by the correction step outweighs the loss by any of these two simplifications (Fig. 5).

V. Conclusion and Future Work

In this paper we presented a fast and precise method for extracting the isosurface geometry in SPH data. We have presented a mesh trimming technique to create the boundary

TABLE I: Computing time (seconds) for a vorticity isosurface ($\omega = 20$) in the vortex rings data, $h = 0.0005$.

h scaling factor	1		0.75		0.5	
Grid size	$0.5h$	h	$0.5h$	h	$0.5h$	h
Grid construction	1.13	1.13	1.4	1.4	2.15	2.15
Seed cell selection	0.72	0.72	0.55	0.55	0.35	0.35
Surface extraction	3.48	0.73	2.90	0.59	1.94	0.39
Total w/o corr.	4.20	1.45	3.45	1.14	2.29	0.74

TABLE II: Computing time (seconds) for a pressure isosurface (time step 149, $p = 2000$) in the tsunami data. The grid construction times raise because of the finer-resolution of the grid when the kernel size becomes smaller. Grid construction time is omitted in the totals as its a preprocessing step.

h scaling factor	1		0.75		0.5	
Grid size	0.028	0.056	0.028	0.056	0.028	0.0562
Grid construction	0.08	0.08	0.1	0.1	0.16	0.16
Seed cell selection	0.7	0.7	0.36	0.36	0.17	0.17
Surface extraction	2.65	0.5	1.72	0.34	0.94	0.19
S.e. with correction	3.45	0.7	3.02	0.65	1.84	0.43
Total w/o corr.	3.35	1.20	2.08	0.7	1.11	0.36
Total with corr.	4.15	1.40	3.38	1.01	2.11	0.60

TABLE III: Computing time (seconds) for a λ_2 isosurface (time step 30, $\lambda_2 = -150$) in the dam break data.

h scaling factor	1		0.75		0.5	
Grid size	0.006	0.012	0.006	0.012	0.006	0.012
Grid construction	0.07	0.07	0.1	0.1	0.17	0.17
Seed cell selection	0.08	0.08	0.05	0.05	0.04	0.04
Surface extraction	0.73	0.17	0.54	0.13	0.38	0.09
S.e. with correction	2.13	0.48	1.63	0.37	0.98	0.23
Total w/o corr.	0.81	0.25	0.59	0.18	0.42	0.13
Total with corr.	2.21	0.56	1.68	0.42	1.02	0.27

TABLE IV: Statistics of virtual grid cells and cells actually used for isosurfaces of the Tsunami data (Table II) and the Dam Break data (Table III) when no particle shrinking is used.

	Tsunami		Dam Break	
Grid size	0.028	0.056	0.006	0.012
Virtual cells	2,401,980	303,408	6,802,912	855,456
Seed cells	73,602	14,917	98,811	41,327
Non-intersected cells	47,007	8,012	38,286	11,151
Total cells used	29,388	7,183	156,810	36,319
Isosurface triangles	56,543	14,535	307,278	73,739

TABLE V: Statistics of the correction step for the tsunami isosurfaces of Table II and for the "dam break" isosurfaces of Table Table III .

	Tsunami		Dam Break	
Grid size	0.028	0.056	0.006	0.012
Mean vector norm	0.000218	0.000807	0.000130	0.000453
Max vector norm	0.009890	0.018360	0.001997	0.005511
St.dev. vector norm	0.000402	0.001253	0.000182	0.000550

of the isosurface at free surfaces. We have also demonstrated that the kernel and the grid size can be reduced or enlarged to get a preview of the surface in very little time. On the other hand, we have shown how a correction step can enhance the precision of the surface significantly. Our method can be applied to any scalar attribute of the SPH particles including derived data such as magnitudes of vectors or dot products of vectors. It is not limited to SPH data, but can be used for other functionally represented data, provided that the computation of gradients based on kernel gradients yields sufficiently accurate results. In comparison to other isosurface geometry extraction methods, our method not only delivers precise results, but also does so within a fraction of time. More specifically, compared to the works of Rosenthal et al. our method runs more than 80 times faster than their method on similarly sized data.

Many extensions and improvements to the Marching Cubes algorithm have been suggested which can be incorporated into the presented approach. Especially notable is recent work improving the topological correctness [43] and the extraction of the surface completely on the GPU [44]. In Section III-B we discuss the slice-based cache we employ to avoid duplicate field evaluations. For very large virtual grids this can require a lot of memory and a more elaborate caching scheme can be beneficial. We plan to investigate this in future work.

Another idea would be an optimization method for speeding up the seed cell selection when the isosurface level is changed. However, unlike in gridded data, seed cell selection is comparably inexpensive because the actual surface extraction requires a lot of computation for evaluating the SPH representation of the field. Another promising approach to optimization could be to exploit temporal coherence between time-steps. This would speed up the visual exploration of time-dependent data.

We also plan to investigate the use of a better free-surface detection method for trimming. The method of Marrone et al. [28] requires the computation of a derived scalar field at grid nodes which are close to both the isosurface and the free surface. The method requires local information including gradients and does therefore not add much to the total computing time.

ACKNOWLEDGMENT

The authors would like to thank Jean-Christophe Marongiu for the "dam breaking" data set and Ben Rogers for the tsunami data set. This work was partially funded by the Swiss National Science Foundation (SNF) under grant 200021_127022 (SPHVis++) and by a grant from the Austrian Science Fund (FWF):P 22542-N23 (Semantic Steering). The project SemSeg acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET-Open grant number 226042.

REFERENCES

[1] P. Rosenthal, S. Rosswog, and L. Linsen, "Direct Surface Extraction from Smoothed Hydrodynamics Simulation Data," in *Fourth High-end Visualization Workshop*, W. Benger, W. Kapferer, S. Venkataraman, R. Heinzl, W. Schoor, M. Tyaki, and G. Weber, Eds. Lehmanns Media - LOB, 2007, pp. 50–61.

[2] H. Carr, B. Duffy, and B. Denby, "On histograms and isosurface statistics," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 12, no. 5, pp. 1259–1266, 2006.

[3] D. J. Price, "SPLASH: An interactive visualisation tool for Smoothed Particle Hydrodynamics simulations," *Publications of the Astronomical Society of Australia*, vol. 24, pp. 159–173, 2007.

[4] A. Henderson, *ParaView Guide, A Parallel Visualization Application*. Kitware Inc. (http://www.paraview.org), 2005. [Online]. Available: http://www.paraview.org

[5] J. Biddiscombe, D. Graham, P. Maruzewski, and R. Issa, "Visualization and analysis of SPH data," *ERCOFTAC Bulletin, SPH special edition*, vol. 76, pp. 9–12, 2008.

[6] M. Levoy, "Display of Surfaces from Volume Data," *IEEE Computer Graphics and Applications*, vol. 8, no. 3, pp. 29–37, 1988.

[7] H. Fuchs, Z. M. Kedem, and S. P. Uselton, "Optimal surface reconstruction from planar contours," *Commun. ACM*, vol. 20, no. 10, pp. 693–702, 1977.

[8] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," *Computer Graphics*, vol. 21, no. 4, pp. 163–169, July 1987. [Online]. Available: http://portal.acm.org/citation.cfm?id=37422

[9] G. M. Nielson and B. Hamann, "The asymptotic decider: resolving the ambiguity in marching cubes," in *VIS '91: Proceedings of the 2nd conference on Visualization '91*. Los Alamitos, CA, USA: IEEE Computer Society Press, 1991, pp. 83–91.

[10] B. K. Natarajan, "On generating topologically consistent isosurfaces from uniform samples," *Vis. Comput.*, vol. 11, no. 1, pp. 52–62, 1994.

[11] J. Wilhelms and A. Van Gelder, "Octrees for faster isosurface generation," *ACM Trans. Graph.*, vol. 11, no. 3, pp. 201–227, 1992.

[12] P. Cignoni, P. Marino, C. Montani, E. Puppo, and R. Scopigno, "Speeding Up Isosurface Extraction Using Interval Trees," *IEEE Transactions on Visualization and Computer Graphics*, vol. 3, pp. 158–170, 1997.

[13] Y.-J. Chiang, C. T. Silva, and W. J. Schroeder, "Interactive out-of-core isosurface extraction," in *Proc. IEEE Visualization*, 1998, pp. 167–174.

[14] T. Itoh and K. Koyamada, "Automatic Isosurface Propagation Using an Extrema Graph and Sorted Boundary Cell Lists," *IEEE Transactions on Visualization and Computer Graphics*, vol. 1, pp. 319–327, 1995.

[15] Y. Livnat, H.-W. Shen, and C. R. Johnson, "A Near Optimal Isosurface Extraction Algorithm Using the Span Space," *IEEE Transactions on Visualization and Computer Graphics*, vol. 2, no. 1, pp. 73–84, 1996.

[16] H.-W. Shen, C. D. Hansen, Y. Livnat, and C. R. Johnson, "Isosurfacing in Span Space with Utmost Efficiency (ISSUE)," in *Proc. IEEE Visualization*. Los Alamitos, CA, USA: IEEE Computer Society, 1996, pp. 287–294.

[17] U. D. Bordoloi and H.-W. Shen, "Space Efficient Fast Isosurface Extraction for Large Datasets," in *Proc. IEEE Visualization*. Los Alamitos, CA, USA: IEEE Computer Society, 2003, pp. 201–208.

[18] Q. Shi and J. JáJá, "Isosurface extraction and spatial filtering using persistent octree (POT)," *IEEE Trans. Vis. Comput. Graph.*, vol. 12, no. 5, pp. 1283–1290, 2006.

[19] C. Wang and Y.-J. Chiang, "Isosurface Extraction and View-Dependent Filtering from Time-Varying Fields Using Persistent Time-Octree (PTOT)," *IEEE Trans. Vis. Comput. Graph.*, vol. 15, no. 6, pp. 1367–1374, 2009.

[20] K. Waters, C. Co, and K. Joy, "Using difference intervals for time-varying isosurface visualization," *IEEE Trans. Vis. Comput. Graph.*, vol. 12, no. 5, pp. 1275–1282, 2006.

[21] T. S. Newman and H. Yi, "A survey of the marching cubes algorithm," *Computers & Graphics*, vol. 30, pp. 854–879, 2006.

[22] Y. Livnat and X. Tricoche, "Interactive point-based isosurface," in *Proc. IEEE Visualization*, 2004, pp. 457–464.

[23] B. von Rymon-Lipinski, N. Hanssen, T. Jansen, L. Ritter, and E. Keeve, "Efficient point-based isosurface exploration using the span-triangle," in *Proc. IEEE Visualization*, 2004, pp. 441–448.

[24] P. Navratil, J. Johnson, and V. Bromm, "Visualization of cosmological particle-based datasets," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1712–1718, 2007.

[25] C. S. Co and K. Joy, "Isosurface Generation for Large-Scale Scattered Data Visualization," in *Proceedings of VMV 2005*, G. Greiner, J. Hornegger, H. Niemann, and M. Stamminger, Eds., 2005, pp. 233–240.

[26] C. Co, B. Hamann, and K. Joy, "Iso-splatting: A point-based alternative to isosurface visualization," in *Proc. Pacific Graphics*, 2004, pp. 325–334.

[27] I. D. Rosenberg and K. Birdwell, "Real-time particle isosurface extraction," in *I3D '08: Proceedings of the 2008 symposium on Interactive 3D graphics and games*. New York, NY, USA: ACM, 2008, pp. 35–43.

[28] S. Marrone, A. Colagrossi, D. L. Touzé, and G. Graziani, "Fast free-surface detection and level-set function definition in SPH solvers," *J. Comput. Phys.*, vol. 229, pp. 3652–3663, 2010.

[29] P. Rosenthal and L. Linsen, "Direct isosurface extraction from scattered volume data," in *EuroVis*, B. S. Santos, T. Ertl, and K. I. Joy, Eds., 2006, pp. 99–106.

[30] ——, "Smooth Surface Extraction from Unstructured Point-based Volume Data Using PDEs," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 6, pp. 1531–1546, 2008.

[31] P. Rosenthal, V. Molchanov, and L. Linsen, "A narrow band level set method for surface extraction from unstructured point-based volume data," in *Proceedings of WSCG, The 18th International Conference on Computer Graphics, Visualization and Computer Vision*, V. Skala, Ed. Plzen, Czech Republic: UNION Agency – Science Press, 2 2010, pp. 73–80.

[32] C. Bajaj, V. Pascucci, and D. Schikore, "Fast isocontouring for improved interactivity," in *Proc. IEEE Visualization*, 1996, pp. 9–46.

[33] NVidia, CUDA technology, http://www.nvidia.com/object/cuda_home_new.html.

[34] P. Bourke, "Polygonising a scalar field," http://local.wasp.uwa.edu.au/~pbourke/geometry/polygonise/.

[35] G. Wyvill, C. McPheeters, and B. Wyvill, "Data structure for soft objects," *The Visual Computer*, vol. 2, pp. 227–234, 1986.

[36] B. D. Rogers and R. A. Dalrymple, "SPH Modeling of Tsunami Waves," in *Advanced Numerical Models for Simulating Tsunami Waves and Runup*, ser. Advances in Coastal and Ocean Engineering, P. L.-F. Liu, H. Yeh, and C. Synolakis, Eds., vol. 10. World Scientific Publishing Co., 2008, pp. 75–100.

[37] K. M. T. Kleefsman, G. Fekken, A. E. P. Veldman, B. Iwanowski, and B. Buchner, "A volume-of-fluid based simulation method for wave impact problems," *Journal of Computational Physics*, vol. 206, no. 1, pp. 363–393, 2005.

[38] W. van Rees and P. Koumoutsakos, "A comparison of vortex and pseudo-spectral methods at high reynolds numbers," (Submitted).

[39] Y. Jang, B. Schindler, R. Fuchs, and R. Peikert, "Volumetric Evaluation of Meshless Data From Smoothed Particle Hydrodynamics Simulations," in *Proc. Volume Graphics 2010*, R. Westermann and G. Kindlmann, Eds., May 2010, pp. 45–52.

[40] J. J. Monaghan, "Smoothed particle hydrodynamics," *Reports on Progress in Physics*, vol. 68, no. 8, p. 1703, 2005. [Online]. Available: http://stacks.iop.org/0034-4885/68/i=8/a=R01

[41] P. R. Schatzle, "An experimental study of fusion of vortex rings," Ph.D. dissertation, California Institute of Technology, Pasadena, California, 1987.

[42] M. Bergdorf and P. Koumoutsakos, "A lagrangian particle-wavelet method," *Multiscale Modeling & Simulation*, vol. 5, no. 3, pp. 980–995, 2006. [Online]. Available: http://link.aip.org/link/?MMS/5/980/1

[43] C. Scheidegger, T. Etiene, L. Nonato, and C. Silva, "Edge flows: Stratified morse theory for simple, correct isosurface extraction," SCI Institute, University of Utah, SCI Technical Report UUSCI-2010-002, 2010. [Online]. Available: http://sci.utah.edu/publications/SCITechReports/UUSCI-2010-002.pdf

[44] C. Dyken, G. Ziegler, C. Theobalt, and H.-P. Seidel, "High-speed marching cubes using histogram pyramids,," *Computer Graphics Forum*, vol. 27(8), pp. 2028–2039, 2008.