IEEE/EG International Symposium on Volume Graphics (2010)
R. Westermann and G. Kindlmann (Editors)

Volumetric Evaluation of Meshless Data From Smoothed Particle Hydrodynamics Simulations

Yun Jang[†], Raphael Fuchs[‡] Benjamin Schindler[§] and Ronny Peikert[¶]

ETH Zürich, Switzerland

Abstract

As an alternative to conventional Eulerian methods in the field of computational fluid dynamics (CFD), smoothed particle hydrodynamics (SPH) has been developed. Its mesh-free method is useful in problems, especially, where a free surface is present. Although researchers are currently able to simulate up to hundreds of millions of particles in a volume, the analysis and visualization of the particle datasets are still limited especially, due to the lack of connectivity between particles and the number of particles. In this paper, we present a volumetric evaluation technique of SPH particle data using hierarchical particle data structures. Our approach does not resample or triangulate the meshless particle data on grid structures, instead, the evaluations are performed in a fragment program with the original SPH kernels used in the CFD simulations. The SPH particle information and the hierarchical data structures are stored in 2D and 3D textures respectively and the 3D texture stores the access pointers, such as texture coordinates, to the 2D textures. To achieve interactive frame rates during interaction we suggest to control the kernel radii while generating the hierarchical data structures. This approach allows us to visualize over a million of SPH particles interactively.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.3]: Smoothed Particle Hydrodynamics, Volume Rendering, Meshless Particle Data—

1. Introduction

Smoothed Particle Hydrodynamics (SPH) is a mesh-free method for simulation which was developed originally in the field of astrophysics [GM77, Luc77], however, it has been applied to areas as diverse as solid mechanics, molecular dynamics, biomechanics and fluid dynamics. Recently, SPH has been studied as an alternative to conventional Eulerian methods in the field of computational fluid dynamics (CFD), and its mesh-free method is useful in problems, especially, where a free surface is present. SPH offers the ability to handle complex solid shapes using boundary representations alone, and also the capacity to handle mixed-fluids and fluid-structure interactions within a unified model. Currently, researchers are able to simulate hundreds of millions of particles and they produce large datasets with several hundreds of time steps. With this data size, analysis and visualization of SPH data has been hampered by the lack of connectivity between particles. Also this leads to time-consuming neighbor searches to retrieve the required information, especially for high-quality volume rendering, where data interpolation and gradient information are important. Two naive approaches to analyze and visualize SPH data could be resampling and triangulation. Resampling SPH data onto uniform grids allows one to apply a broad range of visualization algorithms and implementations in commercial or academic software packages. The major drawback of this resampling approach is the necessary sampling rate. It is not obvious to determine the sampling rate and a constant sampling rate often leads to poor results since the particle density varies over the volume. Another problem is the difficulty in representing solid walls and free surfaces in structured grids. The alternative, to triangulate point-sampled data, is very time-consuming. Triangulation toolkits are not often found in visualization software,

† jangy@inf.ethz.ch
‡ rafuchs@inf.ethz.ch
§ rafuchs@inf.ethz.ch
¶ peikert@inf.ethz.ch

DOI: 10.2312/VG/VG10/045-052

and there are still unsolved issues related to concavities at the domain boundaries. Moreover, triangulation induces an interpolation function which is not compatible with SPH interpolation.

In this work, we study a volume rendering technique for SPH data utilizing hierarchical data structures of particles, which reduce the number of SPH kernels evaluated for each fragment. The hierarchical data structures are stored in 3D textures which are corresponding to input volumes and we pack SPH data in 2D RECT textures. The 3D textures contain access pointers to the 2D RECT textures. In order to add interactivity for the visualization, we control the SPH kernel radii when generating the hierarchical data structures. This reduces the number of SPH kernels which have to be evaluated at a given position in space. Our contributions from this work are as follows.

- We discuss a GPU-based, hierarchical acceleration structure for SPH kernels in order to reduce the number of evaluated SPH kernels for a fragment.
- We present a volumetric evaluation and visualization technique based directly on the SPH kernels.
- The presented technique runs at interactive frame-rates, due to the analysis and control of SPH kernel radii.

In the following section we review related work, then in Section 3 discuss our proposed volumetric evaluation of SPH data. We then present results produced by our system and conclusion with future directions in Section 4 and 5.

2. Related Work

Recently particle based simulations, which are mesh free methods, have been introduced - SPH is one of the most popular mesh free approaches [Mon88] and it follows a Lagrangian formula in its implementation. Due to the relative simplicity of the equations, SPH has become popular in computer graphics, especially for free surface of flow animations [MCG03]. In the visualization community, many particle based techniques have been presented, producing visually stunning images of particle based data. Most of these techniques focus on extracting isosurfaces of particle datasets [BDR98, MNKW07, RRL07, ZK06]. Especially Rosenthal et al. [RRL07] generate *surfels* (surface elements, i.e. points with radius, normal and color information) between selected pairs of neighboring particles using a level set method. An exact isosurface method is presented in [RL08]. For the rendering of particles, Gribble et al. [GIK*07] present ray casting of particles and Ellsworth et al. [EGM04] propose a rendering technique of terascale particles from curvilinear data by minimizing seeks. As an interactive exploration method, Co et al. [CFG*05] present visualization of high dimensional data using scattered plots. Krüger et al. [KKKW05] show a particle system for 3D flows and they present steady 3D flow field on uniform grids using the GPU and store particles on the GPU for interactive exploration. Zhou and Garland [ZG06] present particle based order-independent point rendering from a very large volume of tetrahedral meshes.

In astrophysics applications, Walker et al. [WKM05] and Navrátil et al. [NJB07] present visualization of SPH data from astrophysics simulations. Walker et al. show visualization of particles and contour plots on 2D slices. This work conveys more localized information using the specific slices. Navrátil et al. present visualization of particle data using interpolation on a regular grid. They determine grid resolution automatically to capture sufficient information from the particle data and interpolate data with fixed number of particles to capture more local information.

One of the well-known visualization packages for SPH data is SPLASH [Pri07] which provides a wide range of basic plotting capabilities for scalar or vector attributes of 2D and 3D particle sets. Various types of interactive probes for SPH data analysis are found in [BGM08, JMP*08] and Biddiscombe et al. [BGMT07] discuss a software environment for SPH data visualization offering particle tracing. Schindler et al. [SFBP09] present a method for vortex core line extraction of SPH data without resampling on a grid structure, using a predictor-corrector scheme.

Until recently splatting has been the prevailing volume rendering approach for meshless particle data [Pri07]. Alternatively, slice based volume rendering techniques for point based data have been reported to perform well in GPU implementations. Jang et al. [JWH*04] presented a method to fit radial basis functions (RBFs) to an unstructured volumetric vector field and present a slice-by-slice GPU-based volume rendering technique. For each evaluation point in a slice, a fragment program iterates over all intersecting RBF to compute the color at that position. A similar algorithm by Neophytou et al. [NMM06] computes slices by splatting kernels inside each slab onto a slice. Kähler et al. [KAH07] discuss the combination of a resampled grid with point-based data, achieving interactive rates for a 256^3 structured grid and approximately 10^5 points. They suggest an octree for representation of the grid and use a combination of ray-casting and splatting for volume rendering.

3. Volumetric Evaluation of SPH

In this section, we introduce our volumetric evaluation of SPH data using hierarchical data structures and corresponding GPU texture setup.

3.1. Smoothed Particle Hydrodynamics

SPH was introduced by Gingold and Monaghan [GM77] and Lucy [Luc77] for astrophysical problems, and has later become a general CFD method. A review of SPH theory and application can be found in [Mon05]. Here, we focus on aspects of SPH that are most relevant for visualization.

The *interpolation* rule for a quantity A is

$$A(\mathbf{x}) = \sum_{j=1}^{N} \frac{m_j}{\rho_j} A_j W(\mathbf{x}-\mathbf{x}_j, h_j) \quad (1)$$

where m_j is the mass of the j-th particle, ρ_j its density, A_j the value of A associated with that particle, $\mathbf{x}_j$ its position, h_j its smoothing length, N the number of kernels, and W is the radially symmetric kernel. The summation is done over all particles having the point $\mathbf{x}$ within their kernel support (cf. [Pri07], equation 3). The *normalized interpolation* is obtained by dividing (1) by the interpolation of unity which is

$$\sum_{j=1}^{N} \frac{m_j}{\rho_j} W(\mathbf{x}-\mathbf{x}_j, h_j) \approx 1. \quad (2)$$

The kernels used are the *cubic spline*

$$W(r,h) = \frac{1}{h^3} \begin{cases} 1-\frac{3}{2}q^2+\frac{3}{4}q^3 & 0 \le q \le 1 \\ \frac{1}{4}(2-q)^3 & 1 \le q \le 2 \\ 0 & q \ge 2 \end{cases} \quad (3)$$

and the *quadratic function*

$$W(r,h) = \frac{5}{4\ h^3} \begin{cases} \frac{3}{16}q^2-\frac{3}{4}q+\frac{3}{4} & 0 \le q \le 2 \\ 0 & q \ge 2 \end{cases} \quad (4)$$

where q $= \|r\| /$h. To avoid round-off errors, the interpolation formula is often rewritten as

$$A(\mathbf{x}) = \sum_{j=1}^{N} w_j A_j \mathscr{W}(\mathbf{x}-\mathbf{x}_j, h_j) \quad (5)$$

where $w_j = h_j^{-3} m_j / \rho_j$ and $\mathscr{W}(\mathbf{x}-\mathbf{x}_j, h_j) = h_j^3 W(\mathbf{x}-\mathbf{x}_j, h_j)$ (cf. [Pri07], equation 6). For simplicity, we use the abbreviation $W_j(\mathbf{x})$ for the full interpolation weight of A_j, namely $\frac{m_j}{\rho_j} W(\mathbf{x}-\mathbf{x}_j, h_j)$. Often normalized interpolation is used for free surface or positions where are not located exactly at SPH kernel centers because Equation 2 is not close to one. With this notation, the normalized interpolation is calculated

$$A(\mathbf{x}) = \sum_{j=1}^{N} A_j W_j(\mathbf{x}) \Big/ \sum_{j=1}^{N} W_j(\mathbf{x}) \quad (6)$$

and, using the quotient rule, the gradient can be calculated as

$$\nabla A(\mathbf{x}) = \sum_{j=1}^{N} \big(A_j - A(\mathbf{x})\big) \nabla W_j(\mathbf{x}) \Big/ \sum_{j=1}^{N} W_j(\mathbf{x}). \quad (7)$$

3.2. Hierarchical Data Structure

As shown in Equation 3 and 4, each SPH kernel influences a limited range of a spherical volume. This fact allows us to generate hierarchical data structures of SPH kernels based on locations and radii since there is no need to evaluate kernels which are far from a fragment. Previous work by Jang et al. [JWH*04] suggests using an octree for spatial decomposition to accelerate the evaluation performance. Using an octree forces each subdivision to generate 8 subcells although some of regions do not need to be split. In this case, more texture memory than necessary is consumed to store the decomposed SPH kernels. In this work, instead, we use a binary space partitioning (BSP) tree to decompose SPH kernels. The volume is subdivided at a splitting axis, and the axis is chosen such that the smallest number of SPH kernels have to be duplicated in both subdivisions due to the overlaps with the splitting axis. In other words, we construct a BSP tree with axis aligned splits and we select the splitting axis such that the number of duplicated particles is minimized. Note that the axis must pass through the center of a cell in order to set the tree structure as a 3D texture efficiently. Then we subdivide the volume until the number of SPH kernels in a cell is smaller than a certain number of SPH kernels (N_{max}). This way we can ensure that the evaluation in the fragment program will look up less than N_{max} particles for a fragment evaluation. The left diagram in Figure 1 demonstrates our hierarchical decomposition of SPH kernels in 2D space. The example uses 2 as N_{max} per cell. Depending on the SPH kernel radii, however, sometimes it is not possible to generate all hierarchical cells with less than N_{max} SPH kernels due to the available GPU memory. Therefore, we have to stop generating more hierarchical tree cells at a certain hierarchical level (l_{max}), which means that some hierarchical tree cells have more than N_{max} SPH kernels and this can degrade the performances. We will discuss this issue in more detail in Section 3.4.

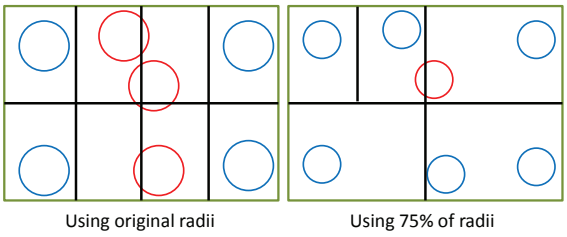


Figure 1: *Hierarchical space decomposition of SPH kernels in 2D case. Circles indicate SPH kernels and bold black lines are decomposition axes. In this example, we split space until each decomposed cell has less than two SPH kernels. The left decomposition is using original SPH kernel sizes and the right decomposition is with 75% of SPH kernel radii. Red kernels are duplicated in the corresponding cells.*

3.3. Texture Encoding and Per Pixel Evaluation

Once the hierarchical structure is generated on the CPU, we store the hierarchical data structure in a 3D texture and the SPH data in RECT textures to obtain a GPU representation. Since we use the centers of cells to decompose the volume as discussed in Section 3.2, the hierarchical data structure is represented as of power of 2 dimensions. For example, as-

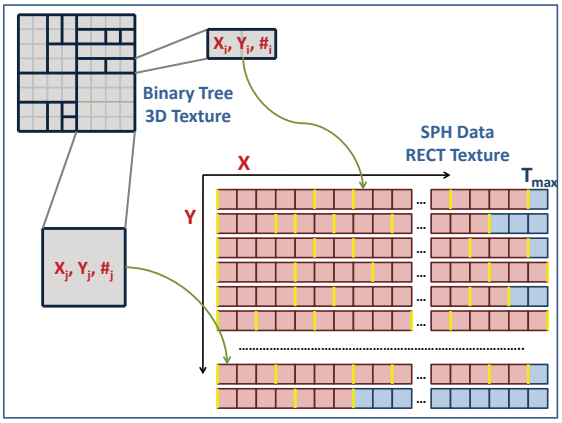


Figure 2: *Texture layouts of the hierarchical tree texture and RECT textures for SPH data. The hierarchical texture is a 3D texture (shown here as 2D) and it contains texture coordinates (X and Y) of RECT textures and the number of SPH kernels (#). The RECT texture is composed of lists of SPH kernels and a list of SPH kernels for one hierarchical cell is placed between two yellow bars. Note that red indicates that texture cells are filled and blue indicates that texture cells are empty in the RECT texture. T_{max} is the maximum RECT texture size. In this figure, as an example, we show two hierarchical cells and their corresponding RECT texture cell positions with green arrows.*

suming that the decomposition levels are l_x, l_y, l_z along x, y, z respectively, the hierarchical structure 3D texture is a $2^{l_x} \times 2^{l_y} \times 2^{l_z}$ volume. Typically l is set as 6 and the size of the 3D texture is about 3 MB, which is small compared to actual GPU memory. This 3D texture is fetched in a fragment program in order to obtain texture coordinates of the RECT textures for the SPH data. Figure 2 presents an example of the hierarchical tree texture at the top left corner. The RECT textures store the SPH kernel data and the order of RECT texture packing is based on the lists of SPH kernels from the hierarchical tree structure. A list of SPH kernels for one hierarchical cell is placed together and then another list of SPH kernels for next hierarchical cell is inserted next to the previous list. Since there is a maximum size (T_{max}) of RECT texture according to graphics hardware, the X texture coordinate can not exceed this maximum size. Therefore, if the accumulated number of SPH kernels is larger than T_{max} along X, we simply move the new list of SPH kernels to next Y position. This texture packing strategy is found in Figure 2. In the RECT texture, blue texture cells indicate that they are empty because the accumulated number of SPH kernels to be placed is larger than T_{max}, therefore, we leave the texture cells as empty. The number of RECT textures varies according to SPH kernels and data values given in the SPH datasets.

The 3D and RECT textures are sent to graphics hardware and we evaluate data values in a fragment program. First

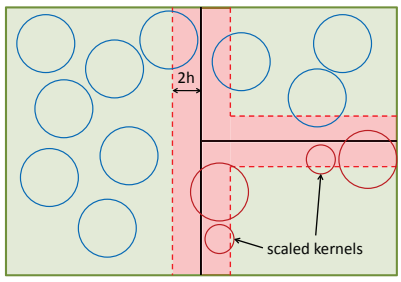


Figure 3: *Error regions (red) in the hierarchical structure. Scaling the radii of SPH kernels causes errors and the error regions are located around the boundaries between hierarchical cells.*

we fetch the 3D hierarchical tree texture at a position of a fragment. The fetched values are the texture coordinates of RECT textures and the number of SPH kernels in the hierarchical cell. Then, we fetch the RECT textures at the corresponding texture coordinates and obtain SPH data values, such as physical quantities and kernel information. We iterate this process over the number of SPH kernels influencing the fragment in order to compute the SPH interpolations shown in Equation 5.

3.4. Adding Interactivity

The SPH datasets used in this work have relatively large kernel radii and each SPH kernel influences broad area of the simulation volume, which means that it is difficult to generate all hierarchical cells with fewer than a certain number of SPH kernels (N_{max}) due to the large overlaps. Therefore, this limits interactivity in the evaluation of SPH kernels. In order to add interactivity, we suggest a control of the SPH kernel size while generating the hierarchical tree structure. By doing so, it is possible to reduce the number of the hierarchical tree cells and the number of duplicated SPH kernels. Therefore, we are able to use less texture memory and to increase the hierarchical level (l_{max}), so that we can decompose the volume into finer structures. Figure 1 shows how the reduced SPH kernel sizes affect the hierarchical tree structure. In the example, 100% (left) and 75% (right) of SPH kernel radii are used. As presented in the figure, using 75% of radii produces fewer hierarchical cells and fewer duplicated SPH kernels. We will refer to this process as *radius scaling* and the numbers in figures and texts are shown as percentages. Note that this radius scaling is applied only when the hierarchical structures are generated and the original radii are used to evaluate Equation 1.

It is obvious that reducing SPH kernel radii means visualization quality degradation. Figure 3 presents an example for the error regions in the hierarchical space. In the example, all errors (red regions) are created around the hierarchical boundaries with the range $2h$ where h is the smoothing length shown in Equation 1. The scaled radii kernels are only excluded in the evaluation within only $2h$ from the bound-

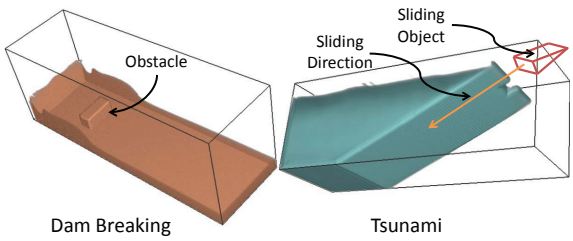


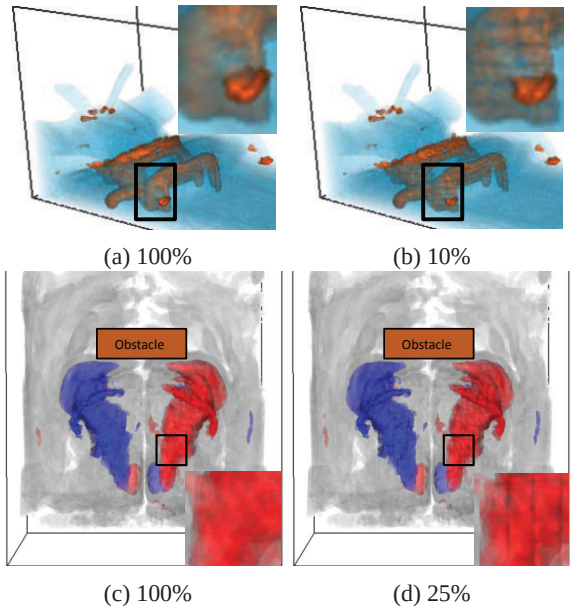
Figure 4: *Two test cases used in this work. The left presents a geometry setting of Dam Breaking simulation and the right shows the one of Tsunami simulation. An obstacle in the Dam Breaking simulation is placed in a way of water flow and a sliding object in the Tsunami simulation is moving down along the sliding direction.*

aries; therefore, the green areas present correct evaluations all the time.

4. Results

We have tested our volumetric evaluation system of SPH data on Xeon CPU 2.66GHz processor with Nvidia GeForce GTX 295 graphics hardware and for a volume rendering we extend ray casting technique [HSS*05] using Nvidia Cg shader. We use two SPH datasets as test cases including Dam Breaking and Tsunami simulations. The Dam Breaking data is from an SPH simulation of the SPHERIC dam break case [KFV*05]. It has 670,575 fluid particles (SPH kernels) with 87 time steps and the cubic spline kernel is used. We use 3 RECT textures, (x,y,z,w), $(velocity_x, velocity_y, velocity_z, h)$, and $(vorticity_x, vorticity_y, vorticity_z, pressure)$, to store the Dam Breaking data for each time step. Note that the SPH kernels have different h values. Solid boundaries are modeled with solid particles, while the air contains no particles. The Tsunami data was generated in an SPH simulation of the creation of a tsunami [RD08]. In this simulation, a wedge is sliding downward which is idealizing a section of earth falling. The number of fluid particles is 1,206,100 per time step. The number of time steps is 248 and the kernel is a quadratic function within a support radius of $2h$. We also use 3 RECT textures, (x,y,z,w), $(velocity_x, velocity_y, velocity_z)$, and $(vorticity_x, vorticity_y, vorticity_z, pressure)$, to store this Tsunami data at each time step. Note that the SPH kernels have a fixed h value. Figure 4 shows the two simulation conditions. The obstacle for the Dam Breaking case and the wedge (sliding object) for the Tsunami case create wave turbulent during simulations.

Figure 5 presents a comparison of vorticity magnitude (top row) and helicity (bottom row) for two radius scalings. The vorticity magnitude is computed as $|\vec{\ }| = |\vec{\nabla} \times \vec{v}|$ where $\vec{v}$ is velocity and the helicity is calculated as $H = \vec{v} \cdot \vec{\ }$. Note that the helicity is computed on the fly using velocity and vorticity stored in the RECT textures, whereas, the vorticity vectors are simply fetched from the texture. (a) and (b)



(a) 100% (b) 10%

(c) 100% (d) 25%

Figure 5: *Comparison of vorticity magnitude (top row) and helicity (bottom row) with velocity and vorticity values given in Dam Breaking data (25^{th} time step) according to radius scaling.*

show vorticity magnitudes colored as orange (high vorticity magnitude) and blue (low vorticity magnitude). (a) is generated with 100% of radius scaling and (b) is with only 10% of the radius scaling. Except the shading discontinuities between hierarchical cells, the major shape of the high vorticity magnitude area is almost identical. (c) and (d) present the helicity. Red color indicates positive helicity values and blue negative helicity values. (c) shows the result with 100% of radius scaling and (d) presents the result with 25% of radius scaling. Even though the shading discontinuities are shown between the hierarchical cells, the overall helicity structures are very similar. Figure 6 proves that the errors caused by the radius scaling are shown only in the hierarchical boundaries. There is no error outside $2h$ range even for the radius scaling 1% as presented in Section 3.4.

Figure 7 shows quality comparisons over various radius scalings using 3 time steps of Dam Breaking data and we select 3 different radius scalings (10%, 60%, and 100%). Vorticity magnitude is used to color the volume and high vorticity values are colored as orange and low vorticity values are colored as blue. Note that the vorticity vectors are precomputed and stored in one of the RECT textures as mentioned above. As shown in the figure, although the artifacts are visible when using 10% radius scaling, the high vorticity features are well preserved. Also it is difficult to find visual differences between 60% and 100% radius scaling for this Dam Breaking case. The performance and the statistics of the hierarchical tree structures are summarized in Table 1 with $l_{max} = 5$ and $N_{max} = 100$. Small radius scaling acceler-

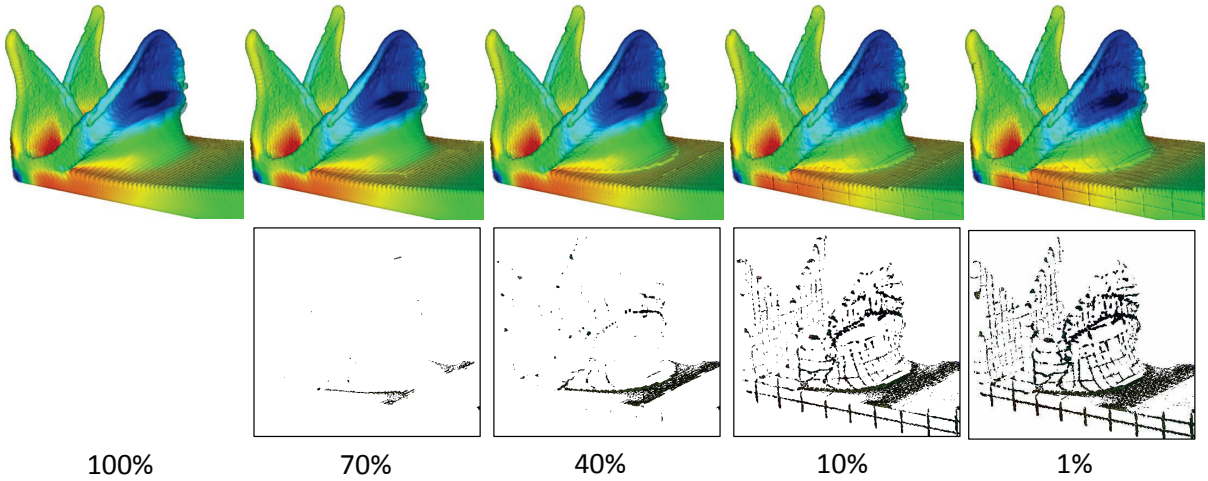


Figure 6: *Error analysis for the Dam Breaking data. Top row shows the five different radius scaling images and bottom row present the differences between 100% and rest of radius scaling images.*

ates the evaluation performances and reduces texture memories.

Similarly, the Tsunami data is used to generate Figure 8 and Table 1 with $l_{max} = 6$ and $N_{max} = 150$. We use again 3 different radius scaling factors, 10%, 40%, and 70%, and pressure contour volumes with 3 isovalues are visualized. Although we use only 70% for radius scaling, it is hard to find artifacts in the figure. However, the isosurface rendering technique is highlighting the artifacts resulting from smaller radius scalings. Nevertheless, overall structures are preserved even using 10% and 40% for radius scalings.

5. Conclusion and Future Work

We have presented a direct volumetric evaluation of SPH data without resampling or triangulation. We first generated a hierarchical structure of SPH kernels in order to reduce the number of SPH kernels evaluated per fragment, then stored the hierarchical structure in a 3D texture. The SPH data are stored in RECT textures according to the hierarchical structure and using the 3D texture allows us to easily find texture coordinates of the RECT textures. Since the SPH kernels used in this work are relatively large, one SPH kernel covers a abroad range of the volume, which keeps us from evaluating the SPH kernels interactively. To improve performances besides the hierarchical data structure we suggested radius scaling, which is a control of the SPH kernel radius, and this enables us to evaluate the SPH kernels interactively with image quality penalties. As future work we would like to extend our system for more complex flow property computations, such as vortex core and flow separation.

Acknowledgments

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 under grant 200021_124642 and grant 200021_127022.

References

[BDR98] BACKES A., DAHR A., RUMPF M.: Interactive visualization of particle systems. In *CGI '98: Proceedings of the Computer Graphics International 1998* (1998), IEEE Computer Society, pp. 88–95. 2

[BGM08] BIDDISCOMBE J., GRAHAM D., MARUZEWSKI P.: Visualization and analysis of SPH data. *ERCOFTAC Bulletin 76* (2008), 9–12. 2

[BGMT07] BIDDISCOMBE J., GEVECI B., MARTIN K., THOMPSON D.: Time dependent processing in a parallel pipeline architecture. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1376–1383. 2

[CFG*05] CO C. S., FRIEDMAN A., GROTE D. P., VAY J.-L., BETHEL E. W., JOY K. I.: Interactive methods for exploring particle simulation data. In *EuroVis* (2005), pp. 279–286. 2

[EGM04] ELLSWORTH D., GREEN B., MORAN P.: Interactive terascale particle visualization. In *VIS '04: Proceedings of the conference on Visualization '04* (2004), IEEE Computer Society, pp. 353–360. 2

[GIK*07] GRIBBLE C. P., IZE T., KENSLER A., WALD I., PARKER S. G.: A coherent grid traversal approach to visualizing particle-based simulation data. *IEEE Transactions on Visualization and Computer Graphics 13*, 4 (2007), 758–768. 2

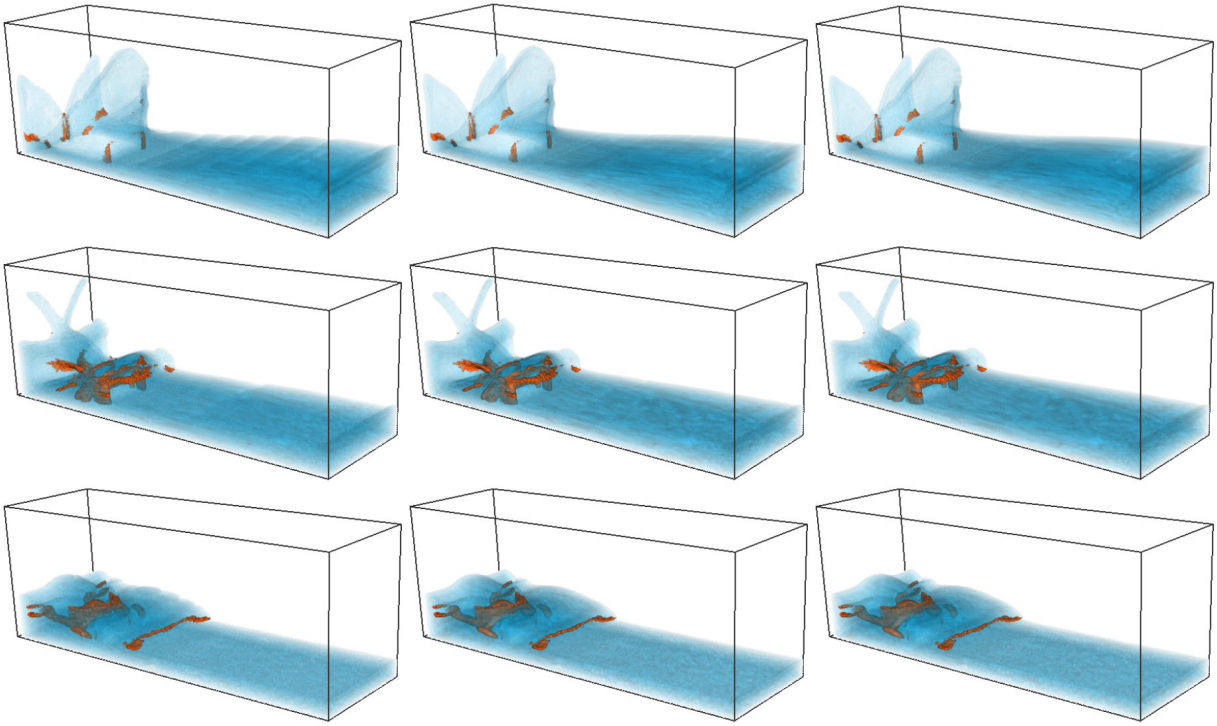
[GM77] GINGOLD R. A., MONAGHAN J. J.: Smoothed particle hydrodynamic: theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society 181* (1977), 375–389. 1, 2

[HSS*05] HADWIGER M., SIGG C., SCHARSACH H., BÜHLER K., GROSS M. H.: Real-time ray-casting and advanced shading of discrete isosurfaces. *Computer Graphics Forum 24*, 3 (2005), 303–312. 5

[JMP*08] JANG Y., MARONGIU J.-C., PARKINSON E., GERVAIS N., GARCIN H.: Analysis of SPH and mesh based simulations using point based post processing tool. In *3rd International SPHERIC SPH Workshop* (June 2008). 2

Table 1: *Hierarchical decomposition and performance statistics for the Dam Breaking and Tsunami SPH data according to the radius scaling. The maximum level (l_{max}) of hierarchy is 5 and 6 and N_{max} is 100 and 150 for the Dam Breaking data and Tsunami data respectively. # of Cells indicates the number of hierarchical tree cells and # of Kernels include all duplicated SPH kernels stored in the RECT textures. The speeds are measured on 600x600 viewport with 256 ray samplings.*

	Radius scaling	100%	60%	10%	100%	60%	10%	100%	60%	10%
Data	Time Step	# of Cells	# of Cells	# of Cells	# of Kernels	# of Kernels	# of Kernels	speed (fps)	speed (fps)	speed (fps)
Dam	13^{th}	9,103	7,851	6,658	5,300,642	2,743,733	891,497	1.5	2.8	8.0
	23^{rd}	7,867	7,443	6,644	5,306,766	2,765,304	896,438	1.6	2.7	8.0
	35^{th}	7,230	6,768	6,364	5,318,221	2,777,993	898,931	1.6	2.8	9.2
Tsunami	90^{th}	129,642	120,208	16,067	59,912,315	19,766,831	1,944,583	0.5	2.3	9.1
	135^{th}	131,219	121,065	16,579	60,028,508	19,890,783	2,011,191	0.4	2.7	8.0
	195^{th}	130,187	121,152	15,846	60,081,964	19,914,904	1,979,655	0.4	2.4	7.9



(a) 10% (b) 60% (c) 100%

Figure 7: *Screen shots of the Dam Breaking data. Vorticity values are visualized and high vorticity volume is colored as orange using 13^{th}, 23^{rd}, and 35^{th} time steps from top to bottom. Column (a), (b), and (c) show the screen shots with 10%, 60%, and 100% of SPH kernel radii.*

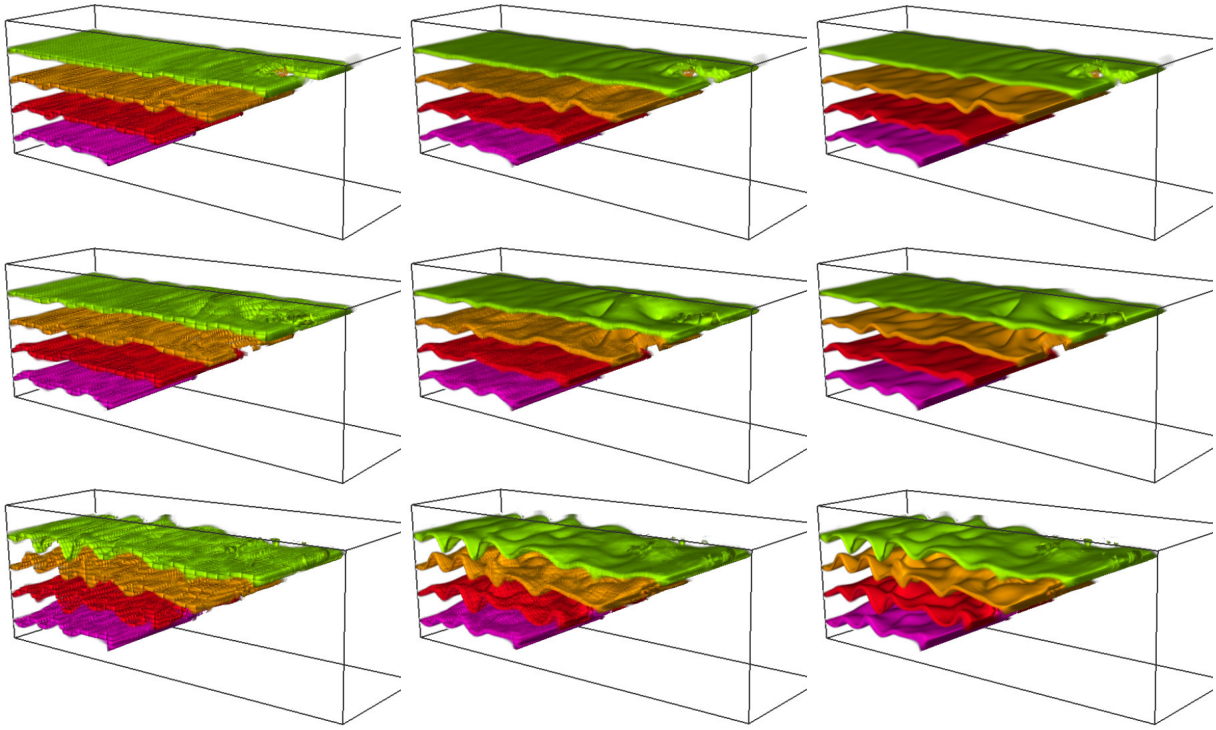
[JWH*04] JANG Y., WEILER M., HOPF M., HUANG J., EBERT D. S., GAITHER K. P., ERTL T.: Interactively visualizing procedurally encoded scalar fields. In *EG/IEEE TCVG Symposium on Visualization VisSym '04* (2004), pp. 35–44, 339. 2, 3

[KAH07] KAEHLER R., ABEL T., HEGE H.-C.: Simultaneous gpu-assisted raycasting of unstructured point sets and volumetric grid data. In *Proc. of IEEE/EG Symposium on Volume Graphics 2007* (2007), p. 49Ű56. 2

[KFV*05] KLEEFSMAN K. M. T., FEKKEN G., VELDMAN A. E. P., IWANOWSKI B., BUCHNER B.: A volume-of-fluid based simulation method for wave impact problems. *Journal of Computational Physics 206*, 1 (2005), 363Ű–393. 5

[KKKW05] KRÜGER J., KIPFER P., KONDRATIEVA P., WESTERMANN R.: A particle system for interactive visualization of 3D flows. *IEEE Transactions on Visualization and Computer Graphics 11*, 6 (11 2005), 744–756. 2

[Luc77] LUCY L. B.: A numerical approach to testing the fission hypothesis. *The Astronomical Journal 82*, 12 (1977), 1013–1924.



(a) 10% (b) 40% (c) 70%

Figure 8: *Screen shots of the Tsunami data. Pressure values are visualized with contour volumes (four isovalues) using* 90^{th}, 135^{th}, *and* 195^{th} *time steps from top to bottom. Column (a), (b), and (c) show the screen shots with 10%, 40%, and 70% of SPH kernel radii.*

1, 2

[MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2003), Eurographics Association, pp. 154–159. 2

[MNKW07] MEYER M., NELSON B., KIRBY R. M., WHITAKER R. T.: Particle systems for efficient and accurate high-order finite element visualization. *IEEE Transactions on Visualization and Computer Graphics 13*, 5 (2007), 1015–1026. 2

[Mon88] MONAGHAN J.: An Introduction to SPH. *Computer Physics Communications 48* (1988), 89–96. 2

[Mon05] MONAGHAN J. J.: Smoothed particle hydrodynamics. *Reports on Progress in Physics 68* (2005), 1703–1759. 2

[NJB07] NAVRÁTIL P. A., JOHNSON J., BROMM V.: Visualization of cosmological particle-based datasets. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1712–1718. 2

[NMM06] NEOPHYTOU N. ., MCDONNELL K., MUELLER K.: *On the simplification of Radial Basis Function fields for volume rendering: some practical insights*. Tech. rep., Stony Brook University, 2006. 2

[Pri07] PRICE D. J.: SPLASH: An interactive visualisation tool for Smoothed Particle Hydrodynamics simulations. *Publications of the Astronomical Society of Australia 24* (2007), 159–173. 2, 3

[RD08] ROGERS B. D., DALRYMPLE R. A.: SPH Modeling of Tsunami Waves. In *Advanced Numerical Models for Simulating Tsunami Waves and Runup* (2008), World Scientific Publishing, pp. 75–100. 5

[RL08] ROSENTHAL P., LINSEN L.: Smooth surface extraction from unstructured point-based volume data using pdes. *Visualization and Computer Graphics, IEEE Transactions on 14*, 6 (Nov.-Dec. 2008), 1531–1546. 2

[RRL07] ROSENTHAL P., ROSSWOG S., LINSEN L.: Direct surface extraction from smoothed particle hydrodynamics simulation data. In *Proceedings of the 4th High-End Visualization Workshop* (2007). 2

[SFBP09] SCHINDLER B., FUCHS R., BIDDISCOMBE J., PEIKERT R.: Predictor-corrector schemes for visualization ofsmoothed particle hydrodynamics data. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (2009), 1243–1250. 2

[WKM05] WALKER R., KENNY P., MIAO J.: Visualization of Smoothed Particle Hydrodynamics for Astrophysics. In *Theory and Practice of Computer Graphics 2005* (June 2005), Lever L., McDerby M., (Eds.), Eurographics Association, pp. 133–138. 2

[ZG06] ZHOU Y., GARLAND M.: Interactive point-based rendering of higher-order tetrahedral data. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 1229–1236. 2

[ZK06] ZHANG H., KAUFMAN A.: Interactive point-based isosurface exploration and high-quality rendering. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 1267–1274. 2