IX International Conference on Particle-Based Methods
PARTICLES 2025
J. Bonet, M. Cremonesi, A. Franci, A. Gil (Eds)

SMOOTHED-PARTICLE HYDRODYNAMICS POST-PROCESSING & VISUALIZATION USING PARAVIEW: A SURVEY

LOUIS GOMBERT[*], FRANÇOIS MAZEN[*], GUILLAUME GISBERT[*] AND JEAN M. FAVRE[†]

[*] Kitware Europe
6 Cours André Philip 69100 Villeurbanne, France
e-mail: kitware@kitware.fr, web page: https://kitware.eu

[†] Swiss National Supercomputing Centre
Via Trevano 131
CH-6900 Lugano

Key words: ParaView, Scientific Visualization, 3D graphics, Smoothed-Particle Hydrodynamics

Abstract. Smoothed-particle hydrodynamics (SPH) simulation is a mesh-free method to simulate solid mechanics or fluid flows by approximating their volume with a set of particles and computation kernels. Hence the output of these simulations is usually a large set of points with associated values such as velocity or pressure. Due to the mesh-free nature of this method, their post-processing and visualization raise challenges to reconstruct the actual volume and extract significant features like interface surface or critical values.
This paper surveys the current and future post-processing methods of SPH simulations using ParaView [1], a reference tool to visualize and explore scientific data at scale. To visualize the millions of particles that SPH simulations can produce, ParaView can discretize particles and their density function over a regular grid, a surface or a line using a point interpolator. This enables using classic visualization techniques such as iso-contours or slicing. We also discuss other indirect rendering methods that can be used in ParaView, such as surface extraction and convex hulls, and introduce GPU representation methods [2], for direct and efficient rendering of particles over time, using gaussian points, volume rendering and ray tracing [3].
Finally, this paper provides an overview of particle rendering methods not yet available in Paraview, such as volume rendering using SPH kernels and data parallel processing algorithms on the GPU, for in situ rendering of large-scale SPH simulations.

1 INTRODUCTION - SPH & VISUALIZATION

Smoothed Particles Hydrodynamics (SPH) is a mesh-free fluid simulation method used in a variety of fields, such as hydrodynamics and astrophysics. SPH, unlike more traditional Computational Fluid Dynamics (CFD) simulations such as the Finite Volume Method, is a Lagrangian approach to simulating fluids, using particles to represent mass. Each particle has a constant mass, a position, a velocity, and discretizes continuous fields such as pressure in space using kernel functions around their position. Particles' properties change over time given the influence of their neighboring particles. This particle model and its relatively easy parallelization makes SPH simulations prime candidates for large-scale High-Performance Computing (HPC) and GPU execution [7]. HPC runs of SPH simulations often top a billion particles. Post-processing such a large number of particles creates new challenges in the context of Scientific Visualization: direct rendering of particles as color-mapped points does not yield a scientifically exploitable result because of the visual clutter it creates.

As a result, extensive literature has been published on visualizing and extracting information from SPH simulation results; some approaches target realistic fluid rendering [8] [9], for applications in movie animation and video games, and others aim for color-mapped indirect rendering for scientific analysis [2]. We will focus on the latter in this paper.

To showcase different post-processing methods, we will use ParaView [1], an industry-standard and open source visualization tool for large datasets. It can be deployed in high-performance compute clusters and controlled remotely using a desktop Qt client. The ParaView server can be run in a distributed environment leveraging MPI, natively distributing and processing data for efficient rendering. ParaView uses "filters" as building blocks for visualization pipelines, so data can be transformed and processed efficiently step by step. ParaView features a large number of filters for point cloud processing and visualization, which makes it an interesting platform for large-scale SPH post-processing.

We will apply post-processing methods on industry-proven open source SPH solvers DualSPHysics [7] and OpenRadioss. The former is a weakly compressible SPH Navier-Stokes solver often used for fluid dynamics, that can efficiently run on the GPU using the CUDA framework. Using DualSPHysics, we run a fluid simulation of a body of water flowing in an enclosed space simulating a dam break impacting on a structure, where particles represent fluid particles of constant mass. This simulation is available as part of the examples of the DualSPHysics package. The output of the simulation is a point cloud, with a pressure scalar field attached, as well as a velocity vector field.

OpenRadioss is Finite Element Analysis software, simulating impact, shock and highly dynamic events. OpenRadioss uses SPH as a complement to the Arbitrary Lagrangian Eulerian formulation [15], used to model fluid-like behaviors using moving points in space. The example simulation we are using models a close-in explosion on a concrete slab, resulting in structural damage of the slab. SPH particles represent the bits of the detonating charge, starting in a small area, and scattering all over after the impact. While these studies usually focus on the damage made to the concrete, we use this simulation to showcase visualization methods on the detonating charge [16], evaluating how the charge's particles scatter over time, and their concentration in space.

This leaves us two very different simulations using SPH particles, on which we will demonstrate different visualization techniques.

2 DIRECT POINT RENDERING TECHNIQUES

Scientific visualization of SPH simulation data should help scientists and decision-makers understand physical phenomena, by creating visually pleasing visualizations from a large number of points, representing particles in space. A first idea for straightforward visualization of SPH points would be to draw them as color-mapped points, but even for smaller datasets, the result in figure 1 does not help understand the particles organization and dynamics very much.

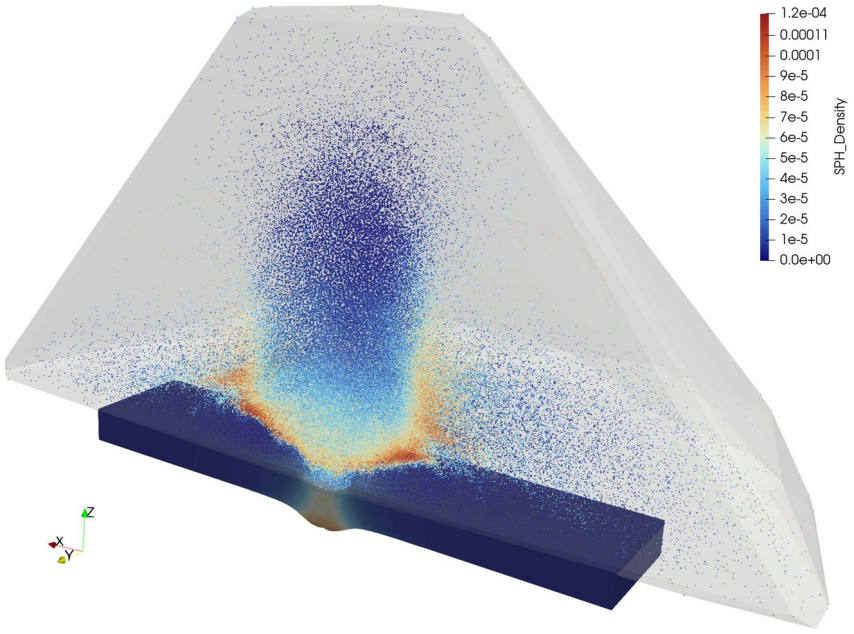

Figure 1: Direct point rendering of around 150k points of an explosion simulated using OpenRadioss. The convex hull containing all particles is depicted as a transparent grey surface.

To reduce the visual noise caused by the large number of points, we can subsample points and try to keep the most relevant ones, preserving the datasets' main features and dimensionality. A possible approach is to derive a histogram from the points scalars being sampled, and select with a higher probability points that fall into bins where the least other points are. When sampling SPH particles on pressure values using this histogram method, we keep extreme and rare values, while discarding many particles in uniform areas. ParaView provides a "Histogram Sampling" filter for this purpose, that is GPU-accelerated using the VTK-m (now Viskores) [6] framework, built for and tested on Exascale computers.

The output of this filter may be visualized directly using color-mapped spheres, or processed further, while keeping in mind that the point density has changed, which means that interpolations and density analysis will not be correct anymore. Rendering these spheres can be done using a GPU shader transforming points into spheres, mapping their color to the scalar field of interest, which is the particle diameter in our case. This visualization method has the advantage of being run exclusively on the GPU using a dedicated shader. While it can work in the order of magnitude of a few hundred thousand cells, it poorly scales with more cells, creating visual clutter that makes information harder to understand. ParaView supports this rendering mode as “Point Gaussian”.

This representation is used in figure 4, with a sampling ratio of 10%, and maps remaining points as spheres with a varying diameter, corresponding to the particle’s “Diameter” scalar field from the simulation output. The majority of particles before decimation had a small diameter, this sampling by histogram balances particle diameters highlighting the location of the whole range of particle diameters. Note that sampling using another scalar field would yield different results and show differences in-between points for this field.

When dealing with large particle simulations, we may also want to query and select subsets of particles from an area of interest. This can be performed in ParaView by a “Clip” operation, selecting particles inside of a box, a sphere or any implicitly defined surface. ParaView’s “Programmable filter” gives more control to the user, allowing to select points lying within a volume such as a sphere or a box, but also selecting points matching a scalar field or geometrical criterion. Figure 2 demonstrates the extraction of particles that lie within a tolerance distance off a vertical slicing plane with a programmable filter, using code from figure 3. This custom filter acts as a plane “probe”, where further particle filtering could easily be added given a condition on density, and integrated to get a particle flow scalar value over time.

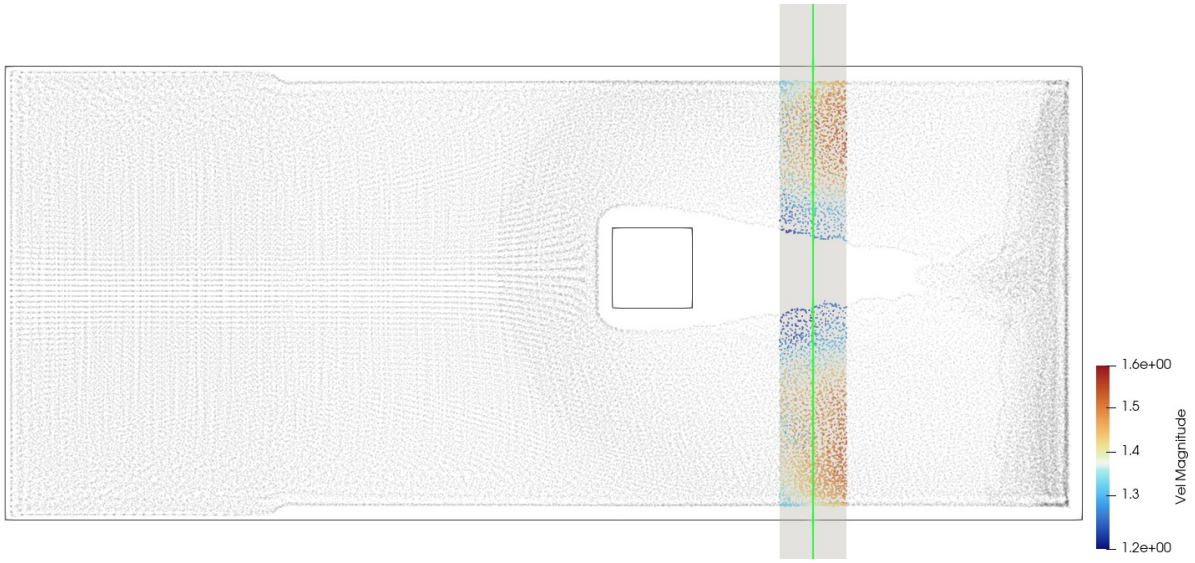

Figure 2: Extraction of particles within a tolerance distance of a plane, shown in green, as seen from the top. The extracted volume is shown as a grey box.

```
from vtkmodules.vtkFiltersPoints import vtkFitImplicitFunction
```

```
from vtkmodules.vtkCommonDataModel import vtkPlane

# Create a plane implicit function
plane = vtkPlane()
plane.SetOrigin([1.2,0.3,0.2])
plane.SetNormal([1, 0, 0])

# Extract points "near" the plane, within a tolerance
tolerance = 0.05
extract = vtkFitImplicitFunction()
extract.SetInputData(inputs[0].VTKObject)
extract.SetImplicitFunction(plane)
extract.SetThreshold(tolerance)
extract.Update()

output.ShallowCopy(extract.GetOutput())
```

Figure 3: ParaView Programmable filter Python code, extracting particles close to a plane, defined using an origin point and a normal.

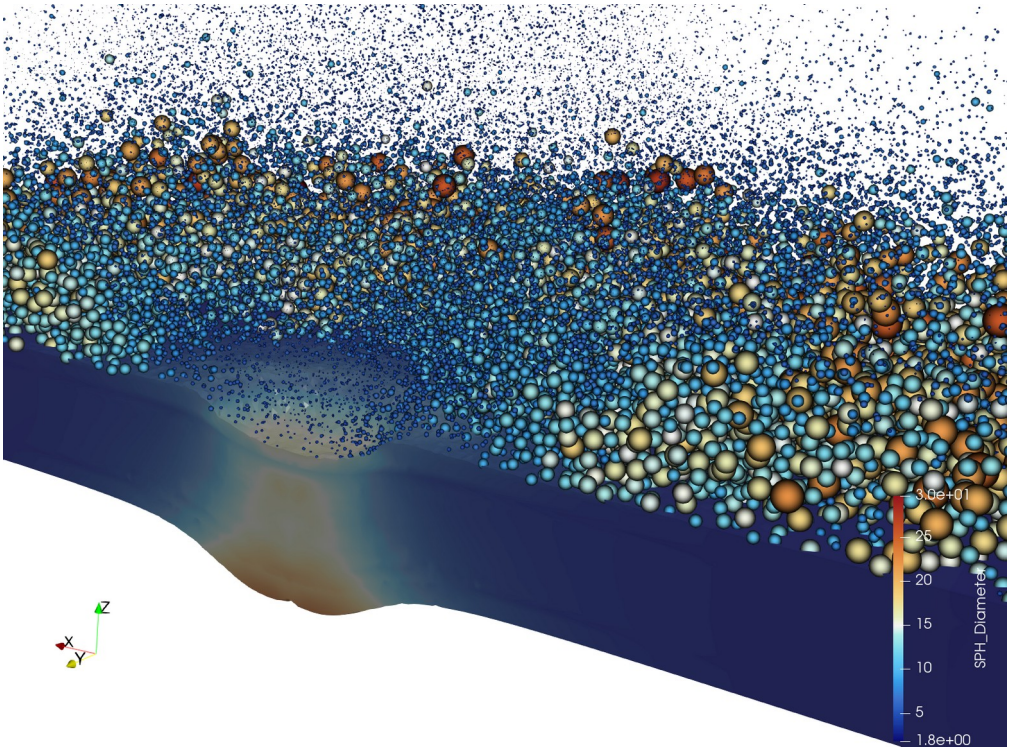

Figure 4: Close-up of the blast simulation, with particles decimated using the Histogram sampling filter. Each sphere represents a particle, diameter and color depending on the *SPH_Diameter* scalar field.

Another interesting algorithm operating directly on point clouds is convex hull computation. Convex hulls compute the smallest convex water-tight surface that contains all points in the point cloud. ParaView features 3D convex hulls through the VESPA [5] plugin wrapping CGAL geometry processing algorithms. Figure 1 shows in transparency the convex hull for the clipped explosion SPH point cloud. Convex hulls can help evaluate the overall volume where points have scattered, and is especially useful when viewed over multiple simulation time steps.

So far, we have only explored static visualization tools that do not take time into account. Both fluid and shock simulations are to be analyzed as dynamic events. Of course, one could create an animation from rendering every time step in a row, but creating a single image showing the changes over multiple steps is another option, better suited for research papers.

ParaView provides a “Temporal Particles To Pathlines” filter, that uses particle identifiers given by the simulation to track points over a time window, and draws the trail associated with changing point positions. The trail is colored given any scalar field changing over time, and the last particle position is represented as a white dot at the end of the trail. DualSPHysics provides unique identifiers for SPH points, so we can track points positions over time and follow their trajectory. Figure 5 represents the SPH fluid simulation as particle pathlines over 50 time steps, randomly decimated to 1/100th, and colored using particle velocity magnitude. This representation shows how a given particle changes position and velocity over a time window. We are also using realistic path tracing rendering, providing shadows that enhance 3D perception and contrast.

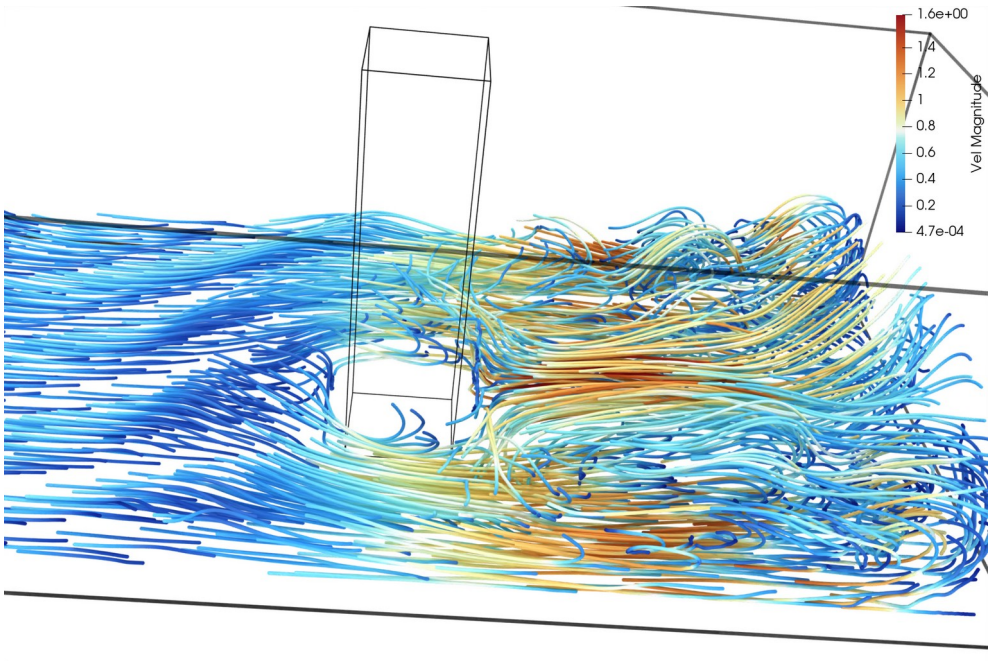

Figure 5: Particles trajectories over time represented as tubes colored by velocity vector magnitude, rendered using ParaView’s pathtracing module.

3 SPH GRID INTERPOLATION

As we demonstrated in the previous section, direct point rendering methods can be used to get a first look at the data, but fall short when scaling to millions of points. Methods involving point decimation lose crucial information about density and overlook SPH particle kernels.

Instead, many scientific visualization post-processing methods for SPH rely on particle density interpolation over a regular grid [10] [11] [12], as they are more suited for data extraction and rendering. ParaView's "SPH Interpolator" filter finds SPH particles around the regular grid's cells by dividing the 3D space into evenly spaced buckets, and keeping in memory a list of points that lie in each bucket. Point densities are then integrated over the cell volume. This interpolator implementation is multi-threaded for faster compute times. The "SPH interpolator" differs from classic interpolation algorithms as it evaluates points' impact on discretized regular cells using a particle kernel function, matching the one used by the simulator. ParaView supports the classic cubic, quartic, quintic and Wendland particle kernels. The interpolation can be done on a line, a 2D plane or a volume.

Once points are interpolated into a regular grid, we can apply more classic scientific visualization algorithms, such as volume rendering [17]. We use a custom color and opacity transfer function to represent the density in 3D space. For the explosion simulation (figure 6), we chose a dark-to-light color-map showing high-density areas as lighter and more opaque, and lower-density areas as dark and transparent. This representation shows the high-pressure areas forming a ring above the slab, where deformation is occurring.

For the fluid simulation, we run an iso-contour algorithm to highlight the waterfront (figure 7). The contour creates a surface at a given density iso-value, and color it using the velocity value.

Interpolating over a plane instead of a volume shows projected point density structures on a 2D surface. For the explosion simulation, we show figure 8 the particle density distribution over a lengthwise cut across the slab.

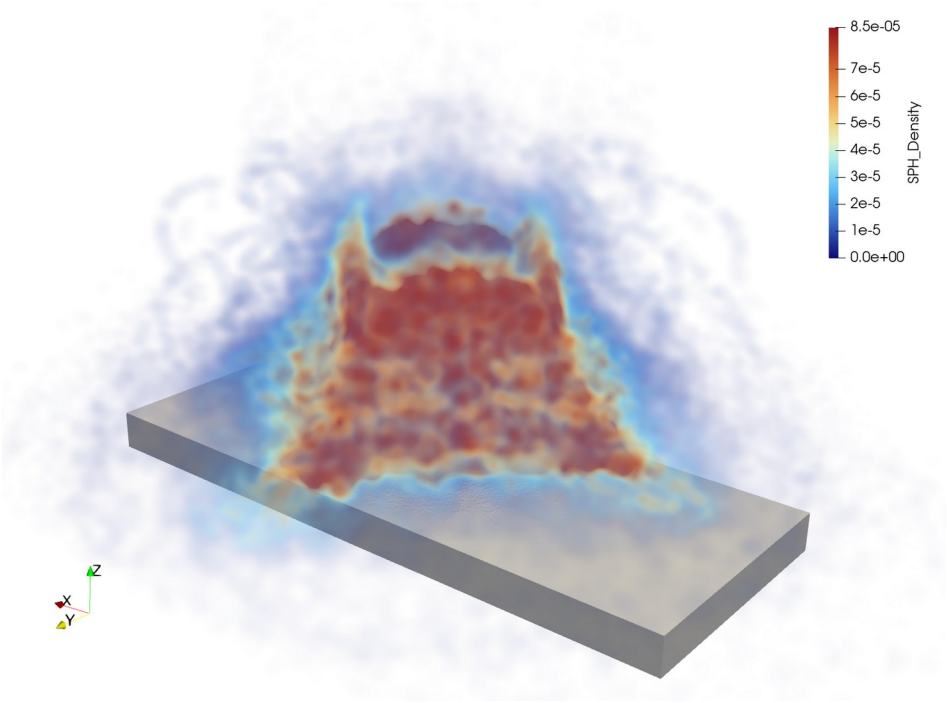

Figure 6: Volume rendering representation, computed from particle densities interpolated over a 3D grid.

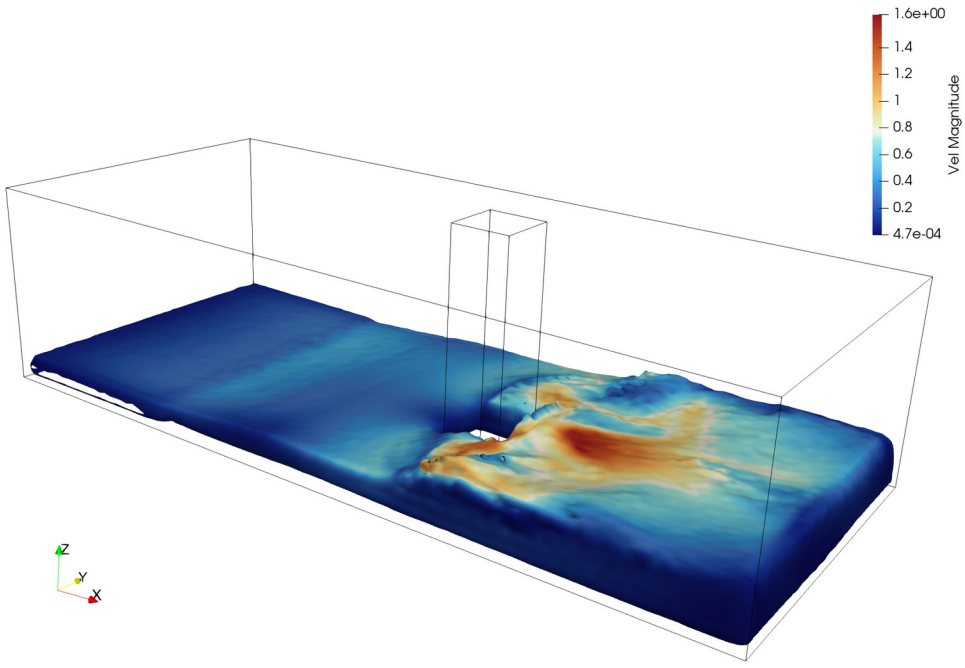

Figure 7. Pressure Isocontour on the fluid simulation, showing the waterfront, colored by the particle velocity field.

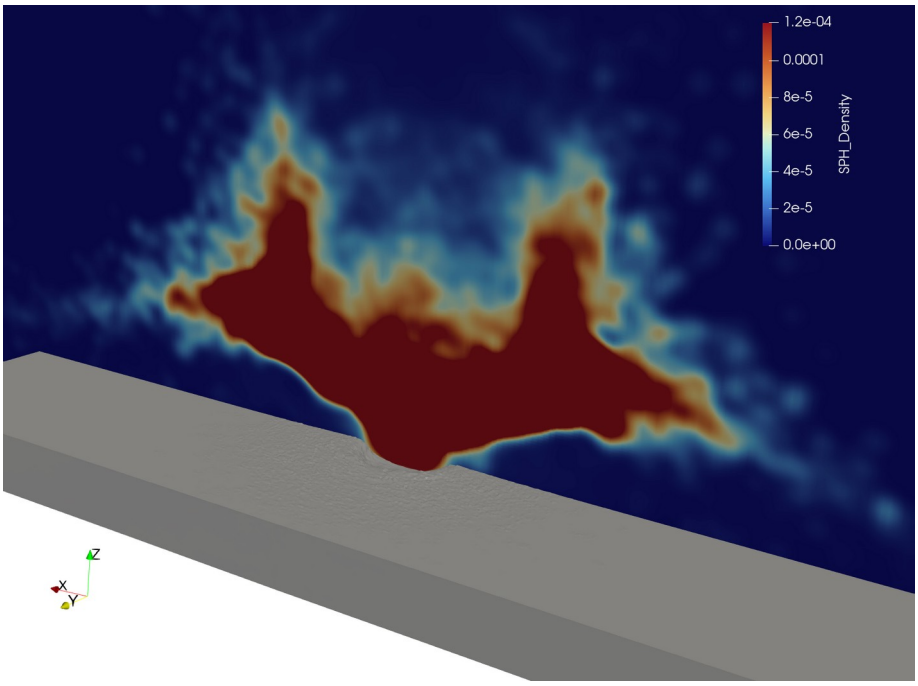

Figure 8: Particle density interpolated over a 2D plane, crossing the explosion simulation lengthwise.

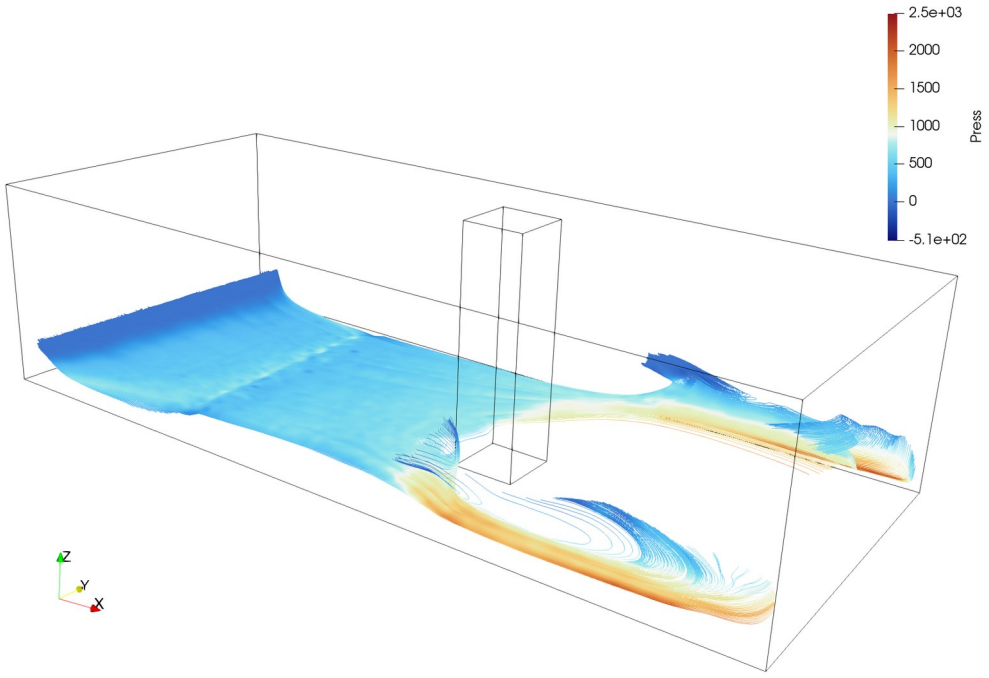

Figure 9: Streamlines, computed on the interpolated volume of the fluid simulation.

One last common visualization technique for CFD simulation is streamlines that show current lines for the vector velocity field. Once again, starting from the SPH particles interpolated over a volume, we set a starting line, and trace current lines through the regular grid, that preserves the velocity vector field set for each particle (figure 9). This reveals the direction of travel of a fluid element at a given point in time.

4 PERSPECTIVES

ParaView being a general-purpose tool for simulation post-processing, it provides many geometry filters and representation models that are not specific to SPH post-processing. The multi-purpose and configurable SPH Interpolator is an effort to make SPH simulation output fit the ParaView data model better, creating a regular grid from particles defined as points with properties, mapping a continuous field in space through a kernel function. However, ParaView is still not a specialized tool for SPH post-processing, and lacks many possibilities and performance improvement offered by ad hoc SPH scientific visualization tools.

For instance, octrees are a popular approach to efficiently store particles [13], evaluating their contributions in discretized cells using their Kernel function, and refining tree cells depending on the cell gradient. ParaView supports tree-based AMR models using 2D or 3D octrees as “HyperTreeGrid” [14], but there is not yet a filter interpolating SPH points to this efficient data model for rendering. ParaView’s pipeline and filter model limits coupling the different processing and rendering steps using data model and rendering mapper abstraction, so it is *de facto* slower than specialized SPH post-processing applications.

One last notable visualization technique is “*in situ*”, a way to view the simulation result as it is running, using the data in memory without copying it to create visualizations on the fly [4]. A large number of SPH simulations run on the GPU, but currently ParaView has limited capabilities for *in situ* processing of GPU memory, limiting the capacity for efficient visualization of SPH simulations as they are running. We can imagine running CUDA-enabled *in situ* visualization pipelines through ParaView, leveraging the VTK-m [6] framework to interpolate and visualize particles without needing to copy data over from the GPU VRAM to the main memory.

5 CONCLUSIONS

In this survey, we highlighted the characteristics of SPH simulation post-processing and visualization using ParaView. First, visualizing the particles only hides a large part of the simulation information because they are just calculation points, not the input surface or flow volume. So, the post-processing should take the kernel function and particle positions into account. ParaView offers several filters to extract meaningful information like the SPH Interpolators. Using real use cases simulated with DualSPHysics and OpenRadioss, we successfully explored the capabilities to retrieve relevant insights from SPH simulations, but we also demonstrated that some features are missing, like free surface extraction. Eventually, the user experience should be improved to ease the visualization process.

With the new computation paradigms using GPU intensively, SPH simulations are creating new challenges regarding post-processing like large numbers of particles and GPU memory management. ParaView should leverage its actual technologies like Viskores or GPU volume rendering to tackle these new challenges. These limitations are more pressing when analysing in situ large data, because zero-copy is mandatory.

We hope that this survey would stimulate the ParaView community regarding the challenges of post-processing modern SPH simulation. Discussions and contributions are very welcomed.

REFERENCES

[1] Ahrens, James, et al. "ParaView: an end-user tool for large-data visualization. Visualization handbook, pp 717–731." Amsterdam, Netherland: Elsevier. ISBN-13 (2005): 978-0123875822.

[2] Biddiscombe, John, David Graham, and Pierre Maruzewski. "Interactive visualization and exploration of SPH data." Proceedings of 2nd SPHERIC international workshop, Madrid (Spain). 2007.

[3] Jin, Zhefan, et al. "High-performance astrophysical visualization using Splotch." Procedia Computer Science 1.1 (2010): 1775-1784.

[4] Ayachit, Utkarsh, et al. "Paraview catalyst: Enabling in situ data analysis and visualization." Proceedings of the first workshop on in situ infrastructures for enabling extreme-scale analysis and visualization. 2015.

[5] Gueunet, Charles, and Tiffany L. Chhim. "VESPA: VTK Enhanced with Surface Processing Algorithms." *EuroVis (Posters)*. 2023.

[6] Moreland, Kenneth, et al. "Vtk-m: Accelerating the visualization toolkit for massively threaded architectures." *IEEE computer graphics and applications* 36.3 (2016): 48-58.

[7] Domínguez, Jose M., et al. "DualSPHysics: from fluid dynamics to multiphysics problems." *Computational Particle Mechanics* 9.5 (2022): 867-895.

[8] Goswami, Prashant, et al. "Interactive SPH simulation and rendering on the GPU." (2010): 55-64.

[9] Löschner, Fabian, et al. "Weighted Laplacian Smoothing for Surface Reconstruction of Particle-based Fluids." *VMV*. 2023.

[10] Price, Daniel J. "SPLASH: An interactive visualisation tool for Smoothed Particle Hydrodynamics simulations." *Publications of the Astronomical Society of Australia* 24.3 (2007): 159-173.

[11] Price, Daniel J. "Smoothed particle hydrodynamics and magnetohydrodynamics." *Journal of Computational Physics* 231.3 (2012): 759-794.

[12] Navratil, Paul, Jarrett Johnson, and Volker Bromm. "Visualization of cosmological particle-based datasets." *IEEE transactions on visualization and computer graphics* 13.6 (2007): 1712-1718.

[13] Chandler, Jennifer, Harald Obermaier, and Kenneth I. Joy. "WebGL-Enabled Remote Visualization of Smoothed Particle Hydrodynamics Simulations." EuroVis (Short Papers). 2015.

[14] Harel, Guénolé, Jacques-Bernard Lekien, and Philippe P. Pébaÿ. "Visualization and analysis of large-scale, tree-based, adaptive mesh refinement simulations with arbitrary

rectilinear geometry." *arXiv preprint arXiv:1702.04852* (2017).
[15] Barlow, Andrew J., et al. "Arbitrary Lagrangian–Eulerian methods for modeling high-speed compressible multimaterial flows." *Journal of Computational Physics* 322 (2016): 603-665.
[16] Loverini, Mathis “Radioss Validation Study: Close-In Explosion on Concrete Slab”, 2025, https://community.altair.com/discussion/62159
[17] Brebin, Robert A., Loren Carpenter, and Pat Hanrahan. "Volume rendering." *Seminal graphics: pioneering efforts that shaped the field*. 1998. 363-372.