On the boundary detection for particle-based methods: visibility, learning, interval analysis, metrics, and applications

Marcos Henrique Alves Sandim

Tese de Doutorado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

Marcos Henrique Alves Sandim

On the boundary detection for particle-based methods: visibility, learning, interval analysis, metrics, and applications

Thesis submitted to the Institute of Mathematics and Computer Sciences – ICMC-USP – in accordance with the requirements of the Computer and Mathematical Sciences Graduate Program, for the degree of Doctor in Science. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Profa. Dr. Afonso Paiva Neto

USP – São Carlos
July 2020

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

```
        Sandim, Marcos Henrique Alves
S213o      On the boundary detection for particle-based
        methods: visibility, learning, interval analysis,
        metrics, and applications / Marcos Henrique Alves
        Sandim; orientador Afonso Paiva Neto. -- São
        Carlos, 2020.
           83 p.


           Tese (Doutorado - Programa de Pós-Graduação em
        Ciências de Computação e Matemática Computacional) --
         Instituto de Ciências Matemáticas e de Computação,
        Universidade de São Paulo, 2020.


           1. Particle systems. 2. Boundary. 3.
        Computational geometry. 4. Machine learning. 5.
        Interval arithmetic. I. Neto, Afonso Paiva, orient.
        II. Título.
```

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

Marcos Henrique Alves Sandim

Sobre detecção de fronteira para métodos baseados em partículas: visibilidade, aprendizado, análise intervalar, métricas, e aplicações

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Doutor em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientadora: Profa. Dr. Afonso Paiva Neto

USP – São Carlos
Julho de 2020

ACKNOWLEDGEMENTS

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) -Finance Code 001.

RESUMO

SANDIM, M. **Sobre detecção de fronteira para métodos baseados em partículas: visibilidade, aprendizado, análise intervalar, métricas, e aplicações**. 2020. 83 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2020.

Esta tese é um estudo compreensivo sobre a definição, desenvolvimento, e avaliação de métodos de detecção de fronteira em sistemas de partículas. Um sistema de partículas é um tipo de representação de dados usada em diversos métodos de simulação de fluidos, como o *Smoothed Particle Hydrodynamics* (SPH) e o *Position Based Fluids* (PBF) usam sistemas de partículas como sua representação primária para o fluido. Outras técnicas como o *Fluid-Implicit-Particle* e o *Affine Particle-In-Cell* (APIC) usam sistemas de partículas como uma representação suplementar. Em ambos os casos a informação sobre a fronteira do sistema de partículas pode ser útil, uma vez que ela provê informação crucial para melhorar a precisão e qualidade da simulação, da geração de uma superfície-livre, ou para reamostrar ou redistribuir partículas em regiões críticas. Apesar disso esse é um problema mal definido e com soluções custosas e propensas a erros. Sendo assim, nós propomos uma definição matemática para o problema, e, a partir dessa definição, exploramos quatro soluções distintas. Baseamos nossas soluções em testes de visibilidade, aprendizado de máquina, e uma combinação de aritmética intervalar e geometria computacional. Nós testamos extensivamente nossas soluções usando diferentes classes de problemas e medimos a sua eficiência. Dados os resultados, nós podemos afirmar que cada uma das nossas soluções possuem características que as fazem adequadas para diversos casos de uso distintos.

Palavras-chave: Sistemas de partículas, Fronteira, Geometria computacional, Aprendizado de máquina, Aritmética intervalar.

ABSTRACT

SANDIM, M. **On the boundary detection for particle-based methods: visibility, learning, interval analysis, metrics, and applications**. 2020. 83 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2020.

This thesis is a comprehensive study of the definition, development, and evaluation of boundary detection methods for particle systems. A particle system is a variety of data-representation used in many fluid simulation methods, such as Smoothed Particle Hydrodynamics (SPH) and Position-Based Fluids (PBF) use particle systems as their primary representation for the fluid. Other techniques, such as Fluid-Implicit-Particle (FLIP) and Affine Particle-In-Cell (APIC), use particle systems as a supplemental representation. In both cases, the knowledge about the boundaries of the particle system can be useful, as it gives crucial information to improve the precision and quality of the simulation, of the generation of a free-surface, or to resample or redistribute particles in critical regions. Despite all that, this is still a poorly defined problem and with costly and error-prone solutions. In light of this, we introduce a mathematical definition for the problem, and, starting from this definition, we explore four distinct solutions. We based our solutions on visibility tests, machine learning, and a combination of interval arithmetic and computational geometry. We thoroughly tested our solutions using different classes of problems and measured their efficiency. Given the results, we can affirm that each of our solutions has characteristics that make them well suited for several distinct use cases.

Keywords: Particle systems, Boundary, Computational geometry, Machine learning, Interval arithmetic.

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF TABLES

CONTENTS

CHAPTER

1 INTRODUCTION

Particle-based methods – also called meshfree methods – are one of the most widely used tools in numerical simulation of physical phenomena. They have been applied in the simulation of fluids for decades since the introduction of the *Smoothed Particle Hydrodynamics* method by Gingold and Monaghan (1977), and many others improved and extended the idea of particle-based fluid simulation. They consist of using a finite set of discrete elements to represent a continuous medium that would otherwise be impracticable to represent and manage using the limited resources of a computer. In a particle-based method, each particle carries information about its position and any other relevant information. In fluid simulation, the pressure, density, and velocity are some of the data needed to compute approximate solutions for the Navier-Stokes equations or Euler equations that describe fluid motion. However, the use of particle-based methods goes beyond the extensions and improvements of the SPH method (LIU; LIU, 2003; GOSWAMI *et al.*, 2010; AKINCI; AKINCI; TESCHNER, 2013; ORTHMANN *et al.*, 2013), with methods such as Position-Based Fluids (PBF) (MACKLIN; MÜLLER, 2013), Fluid-Implicit-Particle (FLIP) (ZHU; BRIDSON, 2005), and Moving-Particle Semi-implicit (MPS) (KOSHIZUKA *et al.*, 2018) taking widely different ways on the use of particles.

The use of meshfree methods has some intrinsic advantages over mesh-based methods. Particles can naturally track changes in topology and adapt to high-detail boundaries. Nevertheless, it is not easy to obtain accurate information about the boundaries themselves in particle-based methods, and authors have proposed many solutions to this problem. One can refer to this problem as the classification of boundary and internal particles, or just boundary detection.

Even though the boundary of a set of particles is not trivial to compute, it has many applications. Incompressibility is a property that is hard to ensure in a fluid simulation using the SPH method. We can solve the incompressibility problem through the Poisson

Pressure Equation, usually referred to as the PPE, which needs information about the boundaries of the particle system to impose boundary conditions (HOSSEINI; FENG, 2011). Without boundary conditions, the PPE can have infinite solutions. Another use of boundary information is on the evaluation of surface tension. Lin, Liu and Wang (2019) attested that information about the boundary of the system could help to calculate the local curvature at the position of the particles and thus help estimate the surface tension. Two-way coupling between fluids and solids can also take advantage of this information, as shown by He *et al.* (2012) and Lin, Liu and Wang (2019).

1.1 Contribution

Given the importance of the boundary computation problem, we explored four different solutions for it. Each one with a different objective in mind, as it is clear that one single solution cannot fit all requirements of each possible application.

Focusing on speed but keeping the consistency and the reliability of the method in mind, we developed a method that reuses traditional data structures present on meshfree methods and applies visibility tests to identify boundary particles. Every particle-based method needs a tool to accelerate the search for neighbors for each particle, and regular grids are one of the most common. We use the grids to distribute a comparatively low number of viewpoints around the boundaries of the particle system in a way that we guarantee that they are close to the boundary. From each viewpoint, we use a local version of the Hidden Point Removal (HPR) operator proposed by Katz et al. By using an appropriate flipping function, we increase the separation between layers before computing the convex hull. This method produces reliable and consistent results; it is simple to implement; and adds low overhead.

One alternative to reduce the runtime overhead of a method is offloading part of the cost to a pre-processing step. Machine learning methods often do this by using a training phase that is considerably more expensive than the classification phase. By reframing the boundary detection problem as a classification problem, we can also use training and classification phases. The use of machine learning has the potential to significantly improve the runtime performance of the boundary detection at the cost of reliability and accuracy. This tradeoff is an option for cases when performance is more important than accuracy.

The use of visibility tests and machine learning have an inherent flaw: they have weak assurances about the overall correctness of the method. Interval analysis, on the other hand, is a tool with theoretical guarantees that empower us to approximate and refine a solution as long as we are capable and willing to reduce the size of the intervals. By using interval analysis, we developed a simple method that adaptatively refines the solution while preserving the guarantee that it will never label a boundary particle as internal. This kind of error is usually referred to as Type I error, or a False Positive. Type II errors – or False Negative – can still occur in our method, but the user can choose to reduce the rate of false negatives in exchange for a slightly higher computational cost.

1.2 Outline

This organization of this text follows our main contributions, where each chapter corresponds to a paper of our authorship:

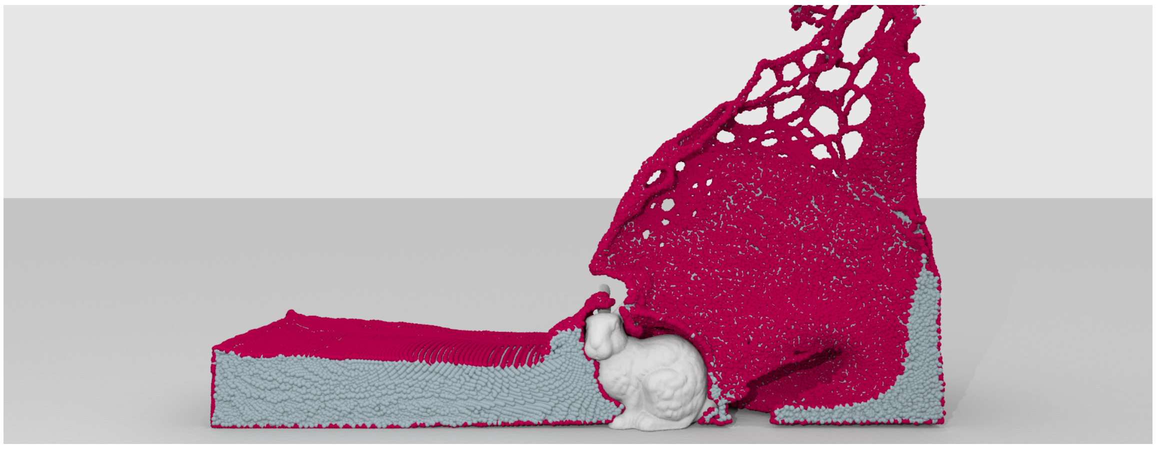
- Chapter 2 presents our visibility-based method and other contributions, such as a mathematical definition for the boundary detection problem, and a metric that can give an overall score to a boundary detection method (SANDIM *et al.*, 2016);
  - Section 2.6 exhibits the application of the visibility-based boundary detection method on a resampling method that can improve the particle distribution in critical areas (SANDIM *et al.*, 2019).
- Chapter 3 shows how we can use the mathematical definition of the boundary detection method to train a Support Vector Machine (SVM) that can classify particles as boundary and internal (SANDIM; PAIVA, 2020);
- Chapter 4 discusses the use of interval analysis to solve the boundary detection problem while preserving some theoretical guarantees. It is an adaptive method that is simple to implement and simple to use (SANDIM; PAIVA; FIGUEIREDO, 2020);
- Chapter 5 concludes the thesis and presents possibilities for future work.

CHAPTER

2

BOUNDARY DETECTION IN PARTICLE-BASED FLUIDS

Figure 1 – The boundary particles (red) detected by our method in a cutaway view of a dam break with Bunny as obstacle.



Source: Sandim *et al.* (2016).

This chapter presents a novel method to detect free-surfaces on particle-based volume representation. In contrast to most particle-based free-surface detection methods, which perform the surface identification based on physical and geometrical properties derived from the underlying fluid flow simulation, the proposed approach only demands the spatial location of the particles to properly recognize surface particles, avoiding even the use of kernels. Boundary particles are identified through a Hidden Point Removal (HPR) operator used for visibility test. Our method is very simple, fast, easy to implement and robust to changes in the distribution of particles, even when facing large deformation of the free-surface. A set of comparisons against state-of-the-art boundary detection methods show the effectiveness of our approach. The good performance of our method is also attested in the context of fluid flow simulation involving free-surface, mainly when using level-sets for rendering purposes.

2.1 Introduction

Particle-based fluid representation has an important role in computer animation, enabling realistic physical simulations of water and blood flows, honey pouring, and smoke dispersion. In fact, meshfree methods such as Smoothed Particle Hydrodynamics (SPH) (IHMSEN *et al.*, 2014), Moving-Particle Semi-implicit (MPS) (PREMZOE *et al.*, 2003), Fluid-Implicit-Particle (FLIP) (ZHU; BRIDSON, 2005), and Position Based Fluids (PBF) (MACKLIN; MÜLLER, 2013) strongly rely on particle representation to simulate topologically intricate fluid flows in complex geometries.

A main issue for meshfree methods is the imposition of boundary conditions, a task that demands accurate identification of *boundary particles*, that is, particles that belong to the free-surface. Inaccuracies in the detection of boundary particles can lead to undesirable results such as unrealistic artifacts during simulation. Moreover, rendering mechanisms can greatly benefit from the information of boundary particles, reinforcing the need of an effective detection of boundary particle.

Despite its importance, the problem of accurately detecting boundary particles has not been extensively addressed in the literature. One of the first attempt to properly tackle the problem is the work by Dilts (DILTS, 2000), where a geometric approach is proposed to identify boundary particles in 2D meshfree simulations. However, Dilts' method is computationally involved, making its extension to 3D not so straightforward (HAQUE; DILTS, 2007). Müller et al. (MÜLLER; CHARYPAR; GROSS, 2003) rely on a simple approach based on the SPH gradient of a color field defined on the particles. A main drawback of using color field gradients is its instability when facing non-uniform particle distribution, making hard the task of finding a global gradient magnitude threshold to properly classify particles as boundary or non-boundary. Zhang et al. (ZHANG; SOLENTHALER; PAJAROLA, 2008) proposed to identify boundary particles based on the distance between a particle and the center of mass of its neighbors. He et al. (HE *et al.*, 2012) improved Zhang's method by adding the SPH particle density information in the boundary/non-boundary test. However, the use of a kernel function to estimate densities and center of masses makes the method sensitive to uneven particle distribution. In order to simulate fluid surface transport, Orthmann et al. (ORTHMANN *et al.*, 2013) compute surface particles using an SPH approximation of the surface area. Their method classifies surface particles using an area threshold, resulting in layers of particles around the surface. In computational physics, Marrone et al. (MARRONE *et al.*, 2010) performs boundary/non-boundary particle classification based on the spectrum of the SPH kernel correction matrix combined with a conical region scan defined from normal vector information. However, Marrone's methodology is restricted to SPH-based numerical simulation with fairly uniform distribution of particles.

In this work we propose a novel method to detect boundary particles in meshfree fluid flow simulations. Our approach converts the boundary particle identification problem in a visibility test problem, employing the well-known *Hidden Point Removal* (HPR) operator proposed by Katz et al. (KATZ; TAL; BASRI, 2007) to identify the particles that belong to the free-surface of the fluid. Figure 1 shows our method in action.

Contributions. In contrast to previous techniques, the proposed method is purely geometric, requiring only particle positions (without normal) to perform the boundary/non-boundary particle classification. In other words, our methodology does not demand any kernel-based interpolation of physical and geometrical properties. Moreover, besides being computationally efficient and easy to implement, our approach is able to deal with non-uniform particle distributions and it is robust to large free-surface deformations.

The effectiveness of our approach is shown through a set of comparisons against state-of-the-art of boundary particle detection techniques. The good performance of the proposed method is also confirmed in experiments involving free-surfaces generated from level-sets defined from boundary particles.

2.2 Boundary and Visibility

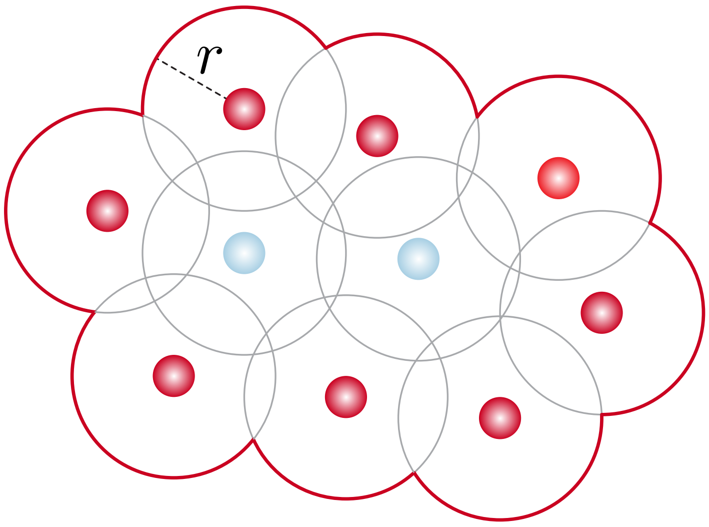
Let $\mathscr{P}$ be a set of points (particles) sampling a compact region $\Omega \subset \mathbb{R}^d$ and $S = \partial\Omega$ be the boundary surface (free-surface in the case of a fluid) of Ω. Our main goal is to identify the subset of points $\mathbf{p} \in \mathscr{P}$ that lie on S, called *boundary points*. Boundary points can be characterized by their visibility, that is, a point $\mathbf{p}$ is a boundary point if there is a viewpoint $\mathbf{V} \notin \Omega$ such that the line segment connecting $\mathbf{p}$ and $\mathbf{V}$ intersects Ω only in $\mathbf{p}$. Intuitively, the visibility criterion says that points $\mathbf{p} \in \mathrm{int}(\Omega)$ are "hidden" from $\mathbf{V}$ by S, therefore, only points on S are visible.

Although intuitive, the visibility test as stated above is not useful in practice, as the region defined by Ω is unknown (only information about the particles are available). In order to precisely define boundary particles in a meshfree context we rely on the concept of *r-sampling* (KATZ; TAL; BASRI, 2007). A point set $\mathscr{P}$ is a r-sampling of a domain Ω if for any $\mathbf{x} \in \Omega$ there exist $\mathbf{p} \in \mathscr{P}$, such that $\|\mathbf{x} - \mathbf{p}\| < r$. Assuming a r-sampling $\mathscr{P}$, consider the set

$$M_r = \{\mathbf{x} \in \mathbb{R}^d \mid \mathrm{dist}(\mathbf{x}, \mathscr{P}) < r\,,\, \mathscr{P} \subset \Omega\}\,. \tag{2.1}$$

Let $B_r(\mathbf{p})$ denotes an open ball of radius r centered at $\mathbf{p}$. A point $\mathbf{p} \in \mathscr{P}$ is called *r-interior* if $\partial B_r(\mathbf{p}) \subset M_r$, where $\partial B_r(\mathbf{p})$ is the boundary of $B_r(\mathbf{p})$. Otherwise, $\mathbf{p}$ is called *r-boundary* and we denote by $\mathscr{B}_r \subset \mathscr{P}$ the set of all r-boundary particles (see Figure 2).

Figure 2 – The r-boundary particles (red) and r-interior particles (blue) from $\mathscr{P}$.



Source: Adapted from Sandim *et al.* (2016).

Computing the r-boundary set $\mathscr{B}_r$ is a laborious, intricate, and computationally intensive task (HAQUE; DILTS, 2007). An alternative is to approximate $\mathscr{B}_r$ rather than compute it directly. A tenable option is to approximate $\mathscr{B}_r$ using the hidden point removal (HPR) operator (KATZ; TAL; BASRI, 2007), which aims at identifying visible points on point-set surfaces.

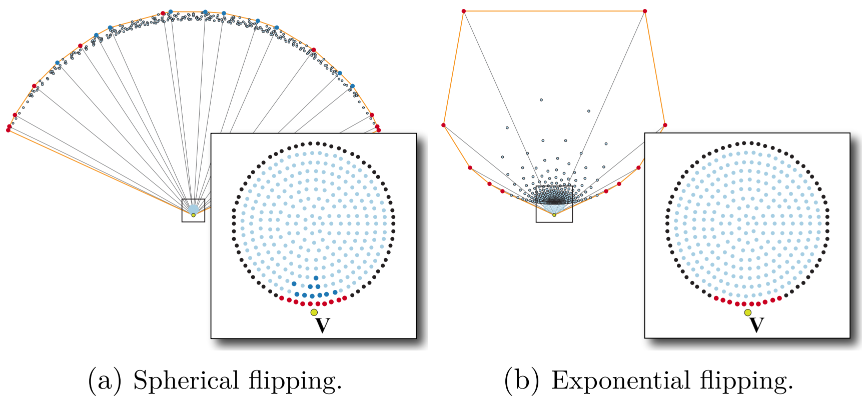
The HPR operator comprises two main steps: inversion and convex hull computation. In the inversion step each point $\mathbf{p} \in \mathscr{P}$ is mapped to an "inverted" domain as to a viewpoint $\mathbf{V}$, that is, points closer to $\mathbf{V}$ are mapped far away while points distant from $\mathbf{V}$ tend to be placed closer to $\mathbf{V}$. Denoting the set of inverted points as $\hat{\mathscr{P}}$, a point $\mathbf{p}$ is labeled as visible if its inverted image $\hat{\mathbf{p}}$ is a vertex of the convex hull of $\hat{\mathscr{P}} \cup \{\mathbf{V}\}$. The second step of the HPR algorithm computes the convex hull of inverted points, returning the vertices of the resulting polyhedron as the visible points.

Spherical flipping (KATZ; TAL; BASRI, 2007) is possibly the most widely used inversion mapping in the context of point-set surfaces. However, when dealing with volumetric particle distributions, the spherical flipping does not present satisfactory results, as illustrated in Figure 3a, where r-interior points end up being labeled as visible when the spherical flipping mapping is employed.

In order to better handle volumetrically distributed particles we opt to the *exponential flipping* inversion mapping. Assuming a viewpoint placed at origin $\mathbf{0}$ and that $\|\mathbf{p}\| < 1$, the exponential flipping is defined as (KATZ; TAL; BASRI, 2007):

$$f(\mathbf{p}) = \begin{cases} \dfrac{\mathbf{p}}{\|\mathbf{p}\|^{\gamma}} & \text{if } \|\mathbf{p}\| \neq 0 \\ \mathbf{0} & \text{otherwise} \end{cases}, \tag{2.2}$$

Figure 3 – Boundary particles (red) detected from a viewpoint $\mathbf{V}$ (green): (a) the effect of HPR operator using spherical flipping, (b) HPR operator using exponential flipping and their correspondent convex hull of $\hat{\mathscr{P}} \cup \{\mathbf{V}\}$ (orange) on top left. Note that, the spherical flipping can produce misclassified points (dark blue).



(a) Spherical flipping. (b) Exponential flipping.

Source: Sandim *et al.* (2016).

where $\gamma > 1$ is a parameter ($\gamma = 1.3$ in our implementation). Figure 3b shows particles labeled as visible when using the exponential flipping inversion mapping, a result clearly better than in Figure 3a. As depicted in Figure 3b, the exponential flipping performs better because it generates larger and more "tangential" displacements, tending to place visible points on the border of the convex hull of inverted points.

An important aspect to be analyzed is whether the set of visible points obtained from the HPR algorithm is indeed a good approximation to the r-boundary set $\mathscr{B}_r$. The following lemma provides an answer to this question by establishing a relation between visible and r-boundary points (see proof in Appendix A):

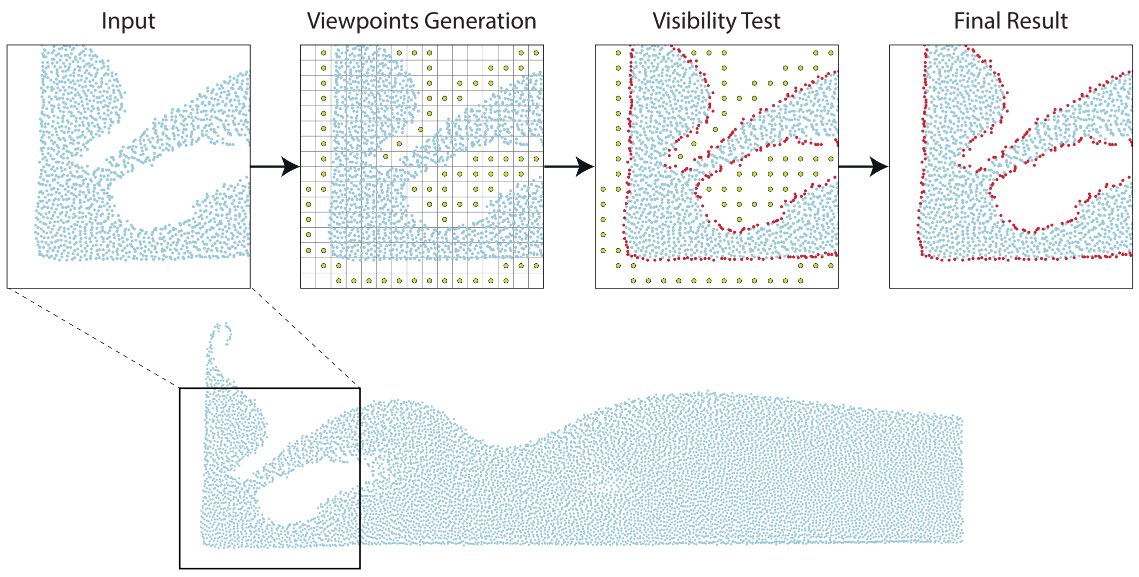
Lemma 2.2.1. Let $\mathscr{P}$ be a r-sampling of a domain $\mathbf{\Omega}$ and $\mathbf{p}$ be a r-boundary point, that is, $\mathbf{p} \in \mathscr{B}_r$. There exists a viewpoint $\mathbf{V}$ such that $\mathbf{p}$ is in the set of visible points $\mathscr{H}_{\mathbf{V}}(\mathscr{P})$ resulting from HPR algorithm.

Lemma 2.2.1 provides theoretical guarantees on the existence of viewpoints from which one can obtain approximations to $\mathscr{B}_r$, however, its implementation is cumbersome and computationally intensive. Next section tackles the problem of creating viewpoints that produce satisfactory results in a computationally efficient manner.

2.3 The Proposed Method

Given a r-sampling set of points $\mathscr{P}$, we have to choose viewpoint positions from which the HPR algorithm can properly identify boundary points. Since our approach rely on HPR to perform the boundary/non-boundary classification, it only requires the

Figure 4 – Overview of our pipeline in a dam break simulation.



Source: Sandim *et al.* (2016).

coordinates of the points as input, disregarding physical attributes typically used by other approaches.

As illustrated in Figure 4, two main steps are involved in our boundary identification process: viewpoint placement (green dots in Figure 4) and the visibility test (red dots in Figure 4). The following subsections detail both steps of our methodology.

2.3.1 Viewpoint Placement

In order to place viewpoints in appropriate locations, we first define a Cartesian uniform grid $\mathscr{G}$ covering the bounding box of $\mathscr{P}$, setting the edge length of each cubic (square in 2D) cell equal to $2r$. A grid cell C is labeled *empty* if it contains no particles in its interior, otherwise, C is labeled *full*.

Viewpoints must be placed outside the domain Ω (the fluid domain in our case). For this reason, empty cells adjacent to at least one full cell become natural candidates to host viewpoints. Therefore, we place a viewpoint in the centroid of each empty cell adjacent to a full cell. However, cavities with a diameter at least $2r$ may be present inside or between full cells and the points on the boundary of those cavities must also be identified. We mark a full cell C as a *cavity cell* if it satisfies the two conditions below:

(a) all cells adjacent to C are full,

(b) there is a ball with center in C and radius r that contains no point of $\mathscr{P}$ in its interior.

Condition (a) is straightforward to check. In order to verify condition (b), for each point

$\mathbf{p}_i$ inside C a candidate viewpoint $\mathbf{V}_i$ is located as follows:

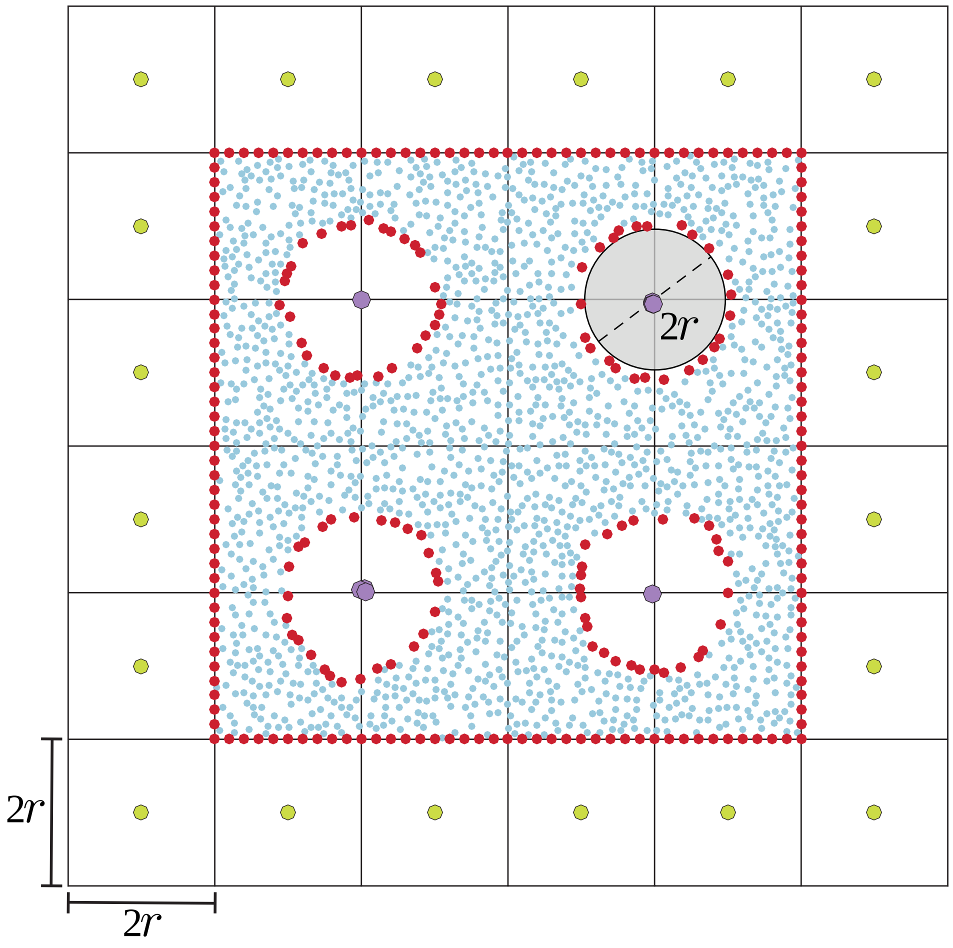
$$\mathbf{V}_i = \begin{cases} \mathbf{p}_i + r\dfrac{\boldsymbol{\delta}_i}{\|\boldsymbol{\delta}_i\|} & \text{if } \|\boldsymbol{\delta}_i\| \neq 0 \\ \mathbf{p}_i & \text{otherwise} \end{cases} \tag{2.3}$$

with

$$\boldsymbol{\delta}_i = \mathbf{p}_i - \frac{1}{|\mathscr{N}_i|}\sum_{j\in\mathscr{N}_i}\mathbf{p}_j\,,$$

where j indexes the points $\mathbf{p}_j$ lying in the neighborhood $\mathscr{N}_i$ of radius $2r$ from $\mathbf{p}_i$. Then, we check the emptiness of each ball $B_r(\mathbf{V}_i)$. If $B_r(\mathbf{V}_i)$ is empty, its center is chosen as a viewpoint. In terms of computational aspects, we decrease the diameter of $B_r(\mathbf{V}_i)$ by multiplying a scale factor of 0.95 due to the non-uniform particle distribution. Figure 5 shows the viewpoints generated in the cavity cells, note that a cavity cell can admit multiple viewpoints.

Figure 5 – Viewpoints generated in empty cells (green) are not enough to detect boundary particles (red) in internal cavities with diameter at least $2r$ (gray). While, viewpoints generated in cavity cells (purple) are able to detect the remaining boundary particles.



Source: Adapted from Sandim *et al.* (2016).

2.3.2 Visibility Test

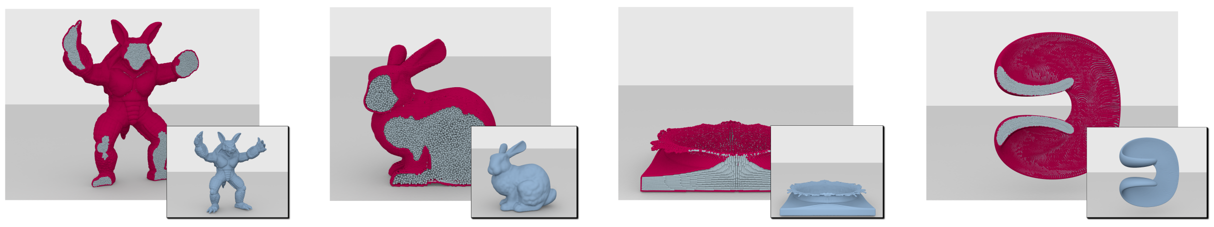
Once the viewpoints have been settled, the HPR operator should be locally applied as discussed in Section 2.2. For each viewpoint $\mathbf{V}_i$, the set of points $\mathscr{P}_i$ within the ball centered in $\mathbf{V}_i$ with radius $4r$ are picked out for inversion. In order to ensure the HPR requirements, apply an affine transformation to $\mathscr{P}_i \cup \{\mathbf{V}_i\}$ such that $\mathbf{V}_i$ is moved to the origin and $\mathscr{P}_i$ is scaled by a factor of $1/4r$. After inversion, the convex hull of the inverted points $\hat{\mathscr{P}}_i \cup \{\mathbf{V}_i\}$ is computed using the QuickHull algorithm (BARBER; DOBKIN; HUHDANPAA, 1996). The union of all points marked as visible after applying the HPR operator from each $\mathbf{V}_i$ gives rise to the set of the boundary points $\mathscr{B}_r$.

The proposed method can be summarized by the Algorithm 1.

Algorithm 1 – Boundary Particle Detection

```
function DETECTION($\mathscr{P}_r$)
    build a uniform grid $\mathscr{G}$ of cell size $2r$
    insert the points of $\mathscr{P}$ in $\mathscr{G}$
    for all cell $C \in \mathscr{G}$ do
        if $C$ is empty and has a adjacent cell that is full then
            place a new viewpoint at the centroid of $C$
        else if $C$ is a cavity cell then
            for all point $\mathbf{p}_i \in C$ do
                place a viewpoint $\mathbf{V}_i$ according to Eq. (2.3) which satisfies the condition
(b)
            end for
        end if
    end for
    for all viewpoint $\mathbf{V}_i$ do
        run the visibility test with all points within radius $4r$ of $\mathbf{V}_i$
        insert the visible points in $\mathscr{B}_r$
    end for
    return $\mathscr{B}_r$
end function
```

Figure 6 – Boundary particles (red) detected by our method. The interior points (gray) are shown in a cutaway view. At bottom-right, their reconstructed surface.


(a) Armadillo (b) Bunny (c) Water Crown (d) Enright Test

Source: Sandim *et al.* (2016).

2.4 Results

We implemented our technique in C++ using `OpenMP` (OpenMP Architecture Review Board, 2011) under the Microsoft Visual Studio 2013 for Windows. All experiments have been performed on an Intel Xeon processor E5620 with four 2.4 GHz cores and 12 GB RAM. Table 1 shows statistics as to the number of particles ($|\mathscr{P}|$), the number of viewpoints (ϑ) generated by our method, the ratio between the number of boundary particles (Bd) and interior particles (Int), and computational times for all 3D experiments presented in this and the following section.

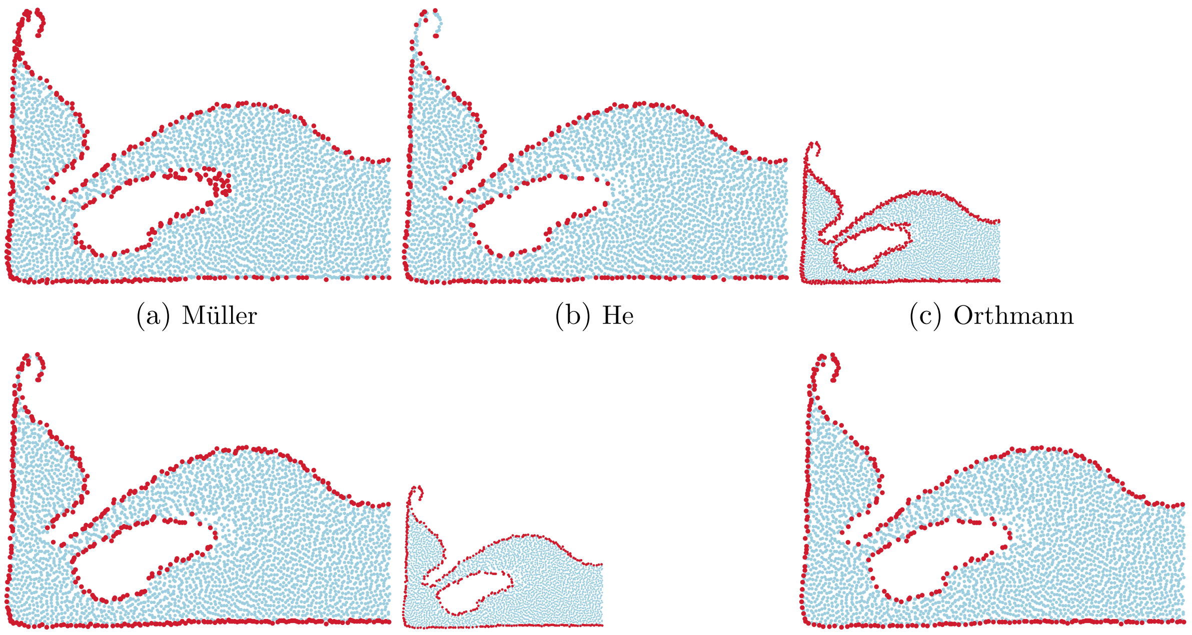
Table 1 – Average statistics and computational times (in seconds) per frame.

Dataset	$\lvert\mathscr{P}\rvert$	ϑ	**Ratio (Bd:Int)**	**Exec. Time**
Bunny	114K	4K	1:2.3	0.9
Double Dam Break	275K	8K	1:9.3	1.6
Water Crown	297K	8K	1:5.7	1.8
Armadillo	797K	18K	1:3.7	7.5
Enright Test	1904K	13K	1:16.8	44.7

The parameter r used to define the underlying grid (see subsection 2.3.1) depends on numerical and spatial resolution of the problem. In SPH simulations a good choice for r is the *smoothing length*, which defines the radius of influence of the kernel function (HAQUE; DILTS, 2007). In our experiments, all particle-based fluid simulations were produced using a weakly compressible SPH formulation implemented in `SPHysics` (GOMEZ-GESTEIRA *et al.*, 2012). The volumetric particle models for Armadillo and Bunny were created by first generating tetrahedral meshes from the corresponding triangular surface meshes using `Gmsh` (GEUZAINE; REMACLE, 2009), getting the vertices of the tetrahedra as particles. Tetrahedron element size and the r parameter were chosen as the average edge length of the input triangular surface meshes (see Figure 6).

The effectiveness of our approach (denoted by "Ours") is assessed through a set of qualitative and quantitative comparisons against four existing surface detection techniques. More precisely, we compare our approach against the methods proposed by Müller et al. (MÜLLER; CHARYPAR; GROSS, 2003) (Müller) , He et al. (HE *et al.*, 2012) (He) and Orthmann et al. (ORTHMANN *et al.*, 2013) (Orthmann), which are well known by the computer graphics community. We also compare against the method proposed by (MARRONE *et al.*, 2010) (Marrone), which is the state-of-art in computational physics. All those four techniques were applied using parameters as suggested in the corresponding papers. Figures 7 and 8 enable qualitative comparisons by showing boundary points resulting from each technique. Notice that our approach is able to capture sharp and thin features better than other methods while producing a reduced number of misclassified interior points.

Figure 7 – Comparison between different boundary detection methods in a 2D dam break simulation.



(a) Müller (b) He (c) Orthmann

(d) Marrone (e) Ours (f) Ground Truth

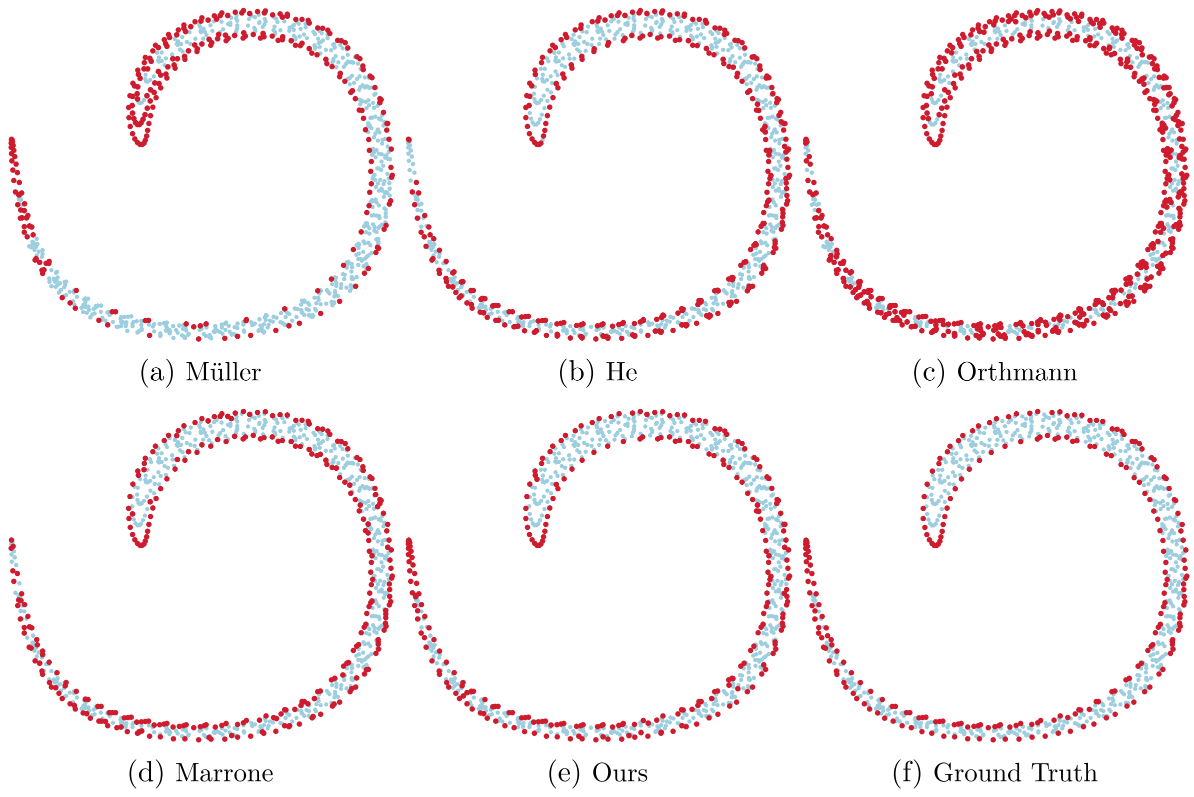
Source: Sandim *et al.* (2016).

For the sake of quantitative comparisons, we apply the method proposed by Haque and Dilts (HAQUE; DILTS, 2007) (and its 2D version (DILTS, 2000)) to label boundary/non-boundary particle, which we will consider as ground truth. The classification provided by Haque and Dilts is fairly reliable, as the method computes, for a given r, the "exact" intersection of spheres with radius r centered at each particle, identifying r-interior and r-boundary particles as defined in Section 2.2. Although very accurate, Haque and Dilts' method is not scalable for practical applications, as its computational complexity is $O(|\mathscr{P}|n^{\varepsilon})$, where n is the average number of particles inside spheres with radius $2r$ centered at the particles and $\varepsilon \in [1.6, 2.0]$ is a problem-dependent constant. Our approach, in contrast, inherits the complexity of the local convex hull computations, resulting in a complexity $O(\vartheta k \log k)$, where ϑ is the number of viewpoints and k is the maximum number of particles used as input the convex hull. Although in the worst case (a full convex hull for each point) our approach is also costly, that worst case is very rare, specially in real applications involving a large number of particles. Given a reference classification, we can assess the accuracy of each method using quantitative metrics commonly employed in data classification (POWERS, 2011).

Specifically, we assign each particle to one of four categories:

- *True Positive* (TP): a r-boundary point correctly classified;
- *True Negative* (TN): a r-interior point correctly classified;

Figure 8 – Comparison between different boundary detection methods in a 2D *single vortex* experiment, as detailed in (ENRIGHT *et al.*, 2002).



Source: Sandim *et al.* (2016).

- *False Positive* (FP): a r-interior point classified as r-boundary;
- *False Negative* (FN): a r-boundary point classified as r-interior.

The *Recall*, is given by

$$Rec = \frac{TP}{TP+FN}.$$

measures how well a technique performs when detecting boundary particles among the actual set of r-boundary particles. The best possible result is $Rec = 1$, meaning that all boundary particles were detected correctly, although false positives can be included. The main issue caused by false positives is that they thicken the boundary surface, thus hampering tasks such as surface reconstruction. On the other hand, the *False Positive Rate* (FPR) quantifies how many interior particles were classified as boundary. Mathematically, FPR is defined as

$$FPR = \frac{FP}{FP+TN}.$$

The best scenario occurs when $FPR = 0$, i.e., no interior particle has been classified as boundary. When employed separately those two metrics may lead to a wrong analysis. For instance, the maximum recall can be achieved by classifying every particle in $\mathscr{P}$ as boundary, regardless whether this is true or not. FPR can be minimized when the whole set of particles is classified as interior. Therefore, we combined both metrics as follows:

$$M_C = Rec * (1 - FPR).$$

The combined metric M_C reaches its maximum value 1 when the Recall is maximum and FPR is minimum simultaneously. Table 2 shows M_C scores resulting from each of the five techniques. Notice that our approach outperforms the four other methods in all experiments.

Table 2 – Quantitative analysis between different boundary detection techniques (best results are shown in bold).

	Dataset	**Method**	M_C
2D cases	Dam Break (Fig. 7)	Müller	0.796
		He	0.795
		Orthmann	0.885
		Marrone	0.949
		Ours	**0.978**
	Single Vortex (Fig. 8)	Müller	0.256
		He	0.708
		Orthmann	0.596
		Marrone	0.873
		Ours	**0.944**
3D cases	Armadillo (Fig. 6a)	Müller	0.000
		He	0.078
		Orthmann	0.822
		Marrone	0.947
		Ours	**0.995**
	Bunny (Fig. 6b)	Müller	0.017
		He	0.027
		Orthmann	0.740
		Marrone	0.921
		Ours	**0.994**
	Water Crown (Fig. 6c)	Müller	0.058
		He	0.848
		Orthmann	0.931
		Marrone	0.957
		Ours	**0.971**
	Enright Test (Fig. 6d)	Müller	0.926
		He	0.938
		Orthmann	0.904
		Marrone	0.968
		Ours	**0.989**
	Double Dam Break (Fig. 11)	Müller	0.589
		He	0.074
		Orthmann	0.919
		Marrone	0.892
		Ours	**0.974**

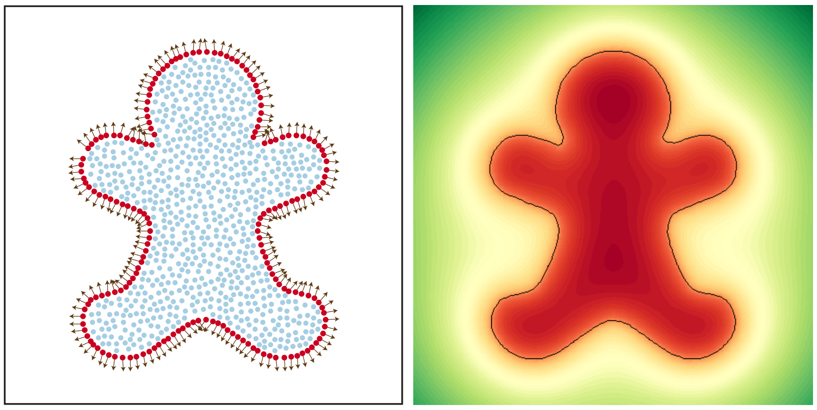
2.5 Applications

One of the main advantages of correctly classifying boundary particles is the possibility of using point-set surface reconstruction algorithms (BERGER *et al.*, 2014) to generate the free-surface of particle-based fluids, instead of SPH-based surface reconstruction (MÜLLER; CHARYPAR; GROSS, 2003; SOLENTHALER; SCHLÄFLI; PAJAROLA, 2007; YU; TURK, 2013) which requires a weighted summation over all neighboring particles. The free-surface S is given implicitly by the zero level-set of a signed distance field $\phi : \mathbb{R}^d \to \mathbb{R}$. Besides the identified boundary particles, the construction of ϕ demands oriented surface normals. Many algorithms have been proposed to compute oriented normals. In our case, the traditional SPH approximation proposed by Müller et al. (MÜLLER; CHARYPAR; GROSS, 2003) is used for estimating oriented normals. Thus, the normal at a boundary particle i is computed as

$$\mathbf{n}_i = \frac{\mathbf{u}_i}{\|\mathbf{u}_i\|} \quad \text{with} \quad \mathbf{u}_i = \sum_{j \in \mathcal{N}_i} \nabla W_h \left(\|\mathbf{p}_i - \mathbf{p}_j\| \right) ,$$

where W_h is the Gaussian kernel function with smoothing length h. Figure 9 shows a 2D example of our boundary detection (BD) method and the computed surface normals (Figure 9a). The smooth signed distance field ϕ defined using *Radial Basis Functions* (RBF) (CARR *et al.*, 2001) and its corresponding surface $S = \phi^{-1}(0)$ is depicted in Figure 9b.

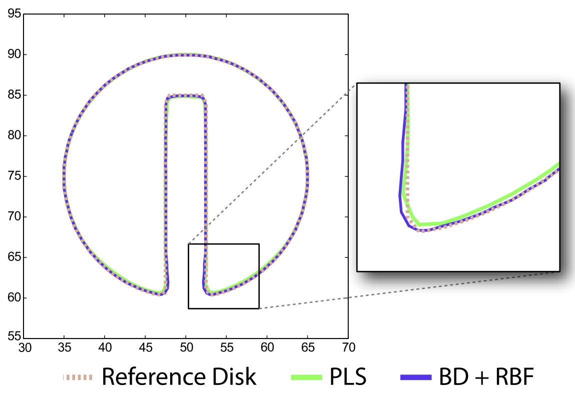
Figure 9 – Surface reconstruction of gingerbread man: (a) boundary particles (red) with oriented normals (brown) and (b) the signed distance field ϕ using RBF, varying from negative (red) to positive (green) values and its zero level-set (black).


(a) (b)

Source: Sandim *et al.* (2016).

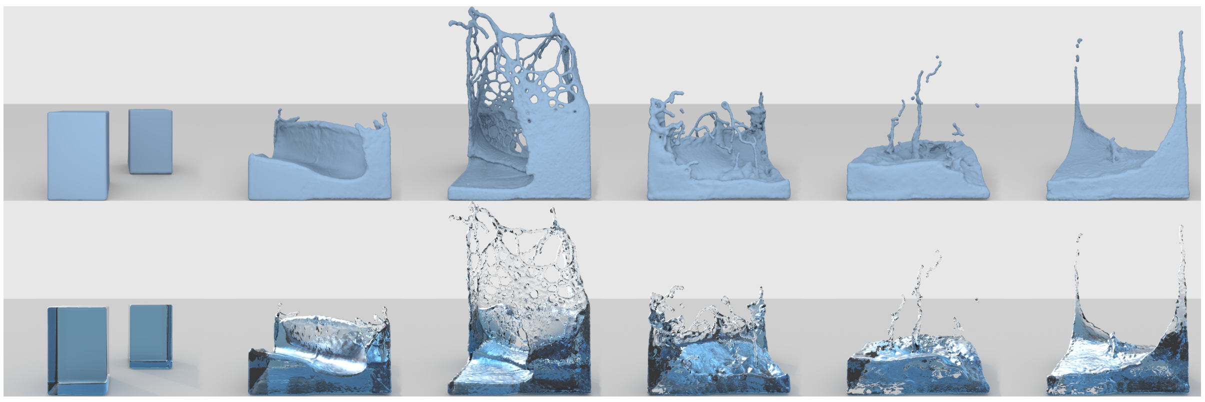
Figure 10 compares level-sets generated by our BD with RBF implicit (CARR *et al.*, 2001) against the *Particle Level Set* (PLS) method (ENRIGHT *et al.*, 2002), a well known level-set method employed in Eulerian grid-based simulations. The experiment shows the resulting boundary surface after a 360° degree rotation of the classical *Zalesak's disc* with fourth-order Runge-Kutta for time integration (ZALESAK, 1979). Notice that the surface resulting from our method is quite close to that generated by PLS, preserving the sharp features and avoiding numerical diffusion (mass loss) at the interface. Moreover, in contrast to PLS, our approach does not demand any level-set re-initialization nor outer marker particles, making it a good asset for any point-set surface reconstruction method.

Figure 10 – Zalesak's disk after one full rotation: our method with 800 particles and PLS on a 100×100 resolution grid.



Source: Sandim *et al.* (2016).

Figure 11 – The free-surface reconstruction of a double dam break simulation with opaque (top) and transparent (bottom) renderings.

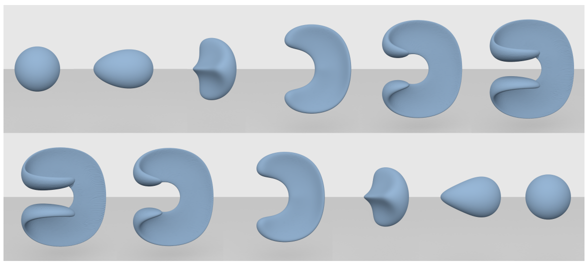


Source: Sandim *et al.* (2016).

Figure 11 shows our BD method combined with *Screened Poisson surface reconstruction* (KAZHDAN; HOPPE, 2013) to compute the level-set surface that represents the liquid interface. Notice that due to the good performance of our BD method, the level-set obtained by the Screened Poisson method nicely captures droplets and thin sheets of water (usually formed by a thin layer of particles) in a double dam break simulation. In all 3D experiments shown in this section we used the Screened Poisson as surface reconstruction method (notice that Screened Poisson can not operate in volumes).

Figure 12 shows the effectiveness of our approach when facing the *Enright test* (ENRIGHT *et al.*, 2002), where a sphere is advected by a velocity field which induces large shape deformation. Notice that the sphere is preserved even after a stretching followed by a compression. In this example, we used fourth-order Runge-Kutta to compute particles trajectories.

Figure 12 – Enright test: the shape of sphere is preserved by our approach despite large deformations (start: top left, end: bottom right).

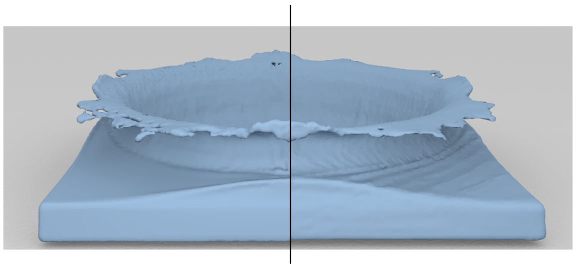


Source: Sandim *et al.* (2016).

Figure 13 shows a comparison between our approach and the surface reconstruction method proposed by Zhu and Bridson (ZHU; BRIDSON, 2005), which also operates on the particle volume. One can easily see that Zhu and Bridson's method introduces surface bumps due to the irregular particle distribution. Those artifacts are not present in the reconstruction generated from our approach. We use an implementation of Marching Cubes (MC) algorithm (LEWINER *et al.*, 2003) to extract the isosurface from the output of Zhu and Bridson's method, defining the MC grid size from the particle spacing (AKINCI *et al.*, 2012). Moreover, the surface reconstruction with Screened Poisson took a similar time of Zhu and Bridson's method, approximately 5 seconds per frame.

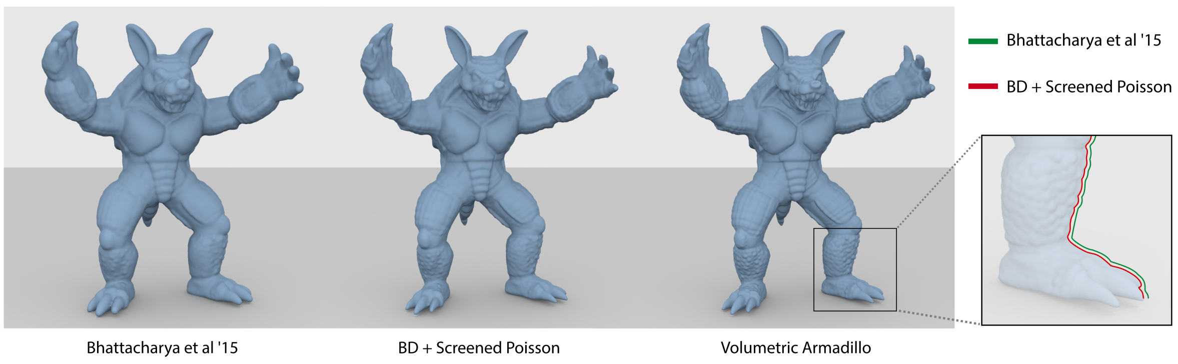
A common strategy when dealing with particle volumes is to perform surface reconstruction directly from the volume defined by the particles. Bhattacharya et al. (BHATTACHARYA; GAO; BARGTEIL, 2015) proposed a technique for skinning particle data that relies on a biharmonic-smoothed distance field constrained between two-offset surfaces obtained from the particles. Figure 14 presents surfaces reconstructed from a volumetric Armadillo model using our approach and the particle skinning proposed by Bhattacharya. Notice that the reconstruction generated by Screened Poisson from the boundary surface captured by our method contains most of the details and sharp features present in the original model. Bhattacharya et al. technique, in contrast, produced a smoother reconstruction. In this experiment, our BD method with Screened Poisson took approximately 16 seconds (BD corresponds to $\sim 47\%$ of the time) against 105 seconds of Bhattacharya's method.

Figure 13 – Comparison with our method (left) and the surface reconstruction proposed by Zhu and Bridson (ZHU; BRIDSON, 2005) (right) on the water crown formed by the impact of a drop on a water layer.



Source: Sandim *et al.* (2016).

Figure 14 – Comparison of our strategy (middle) against Particle Skinner (left) proposed by Bhattacharya et al. (BHATTACHARYA; GAO; BARGTEIL, 2015). At the right-most, the silhouette profiles of each reconstruction strategy is shown in detail against the reference volumetric Armadillo.



Source: Sandim *et al.* (2016).

2.6 Boundary particle resampling for surface reconstruction in liquid animation

The renderings shown in Section 2.5 can suffer from poor particle distribution. Three main feature types can hinder the quality of the generated mesh:

- Droplets: single particles disconnected from the rest of the fluid;
- Ligaments: thin fluid features in the form of strings;
- Thin sheets: overstretched sheets of fluid that contain a single layer of particles.

These features can cause surface-fitting algorithms (KAZHDAN; HOPPE, 2013) to miss some portions of the fluid. To overcome this problem, we developed a simple scheme to identify them and to replace them with better-distributed particles (SANDIM *et al.*, 2019).

We explored different ways to identify particles in critical areas. We based our approach in the spectral decomposition of the covariance matrix of a particle:

$$\mathbf{C}_i = \frac{1}{|\mathcal{N}_i|}\sum_{j\in\mathcal{N}_i}(\mathbf{x}_j - \bar{\mathbf{x}}_i)(\mathbf{x}_j - \bar{\mathbf{x}}_i)^\top \quad \text{with} \quad \bar{\mathbf{x}}_i = \frac{1}{|\mathcal{N}_i|}\sum_{j\in\mathcal{N}_i}\mathbf{x}_j\,. \tag{2.4}$$

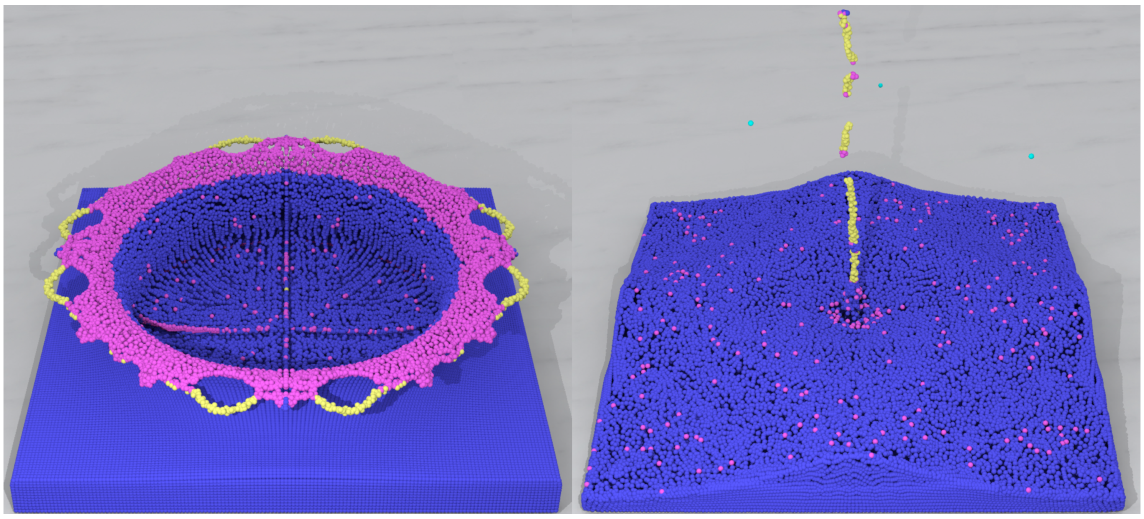
From the matrix Ci, we computed the eigendecomposition $\mathbf{C}_i = \mathbf{U}_i\Sigma_i\mathbf{U}_i^\top$ using the Singular Value Decomposition (SVD) present in the Eigen library[1] (GUENNEBAUD; JACOB *et al.*, 2010). The matrix Σ_i gives us the three positive eigenvalues in ascending order: $\Sigma_i = \operatorname{diag}(\sigma_1, \sigma_2, \sigma_3)$.

Along with these values, we employ two parameters K and α, where K is a lower bound on the number of neighbors of a particle, and α is a threshold that allows us to regulate the desired degree of deformation of our features. In our experiments, we used $K = 4$ and $\alpha = 0.2$. We can write the final labeling scheme as:

$$\ell_i = \begin{cases} 0\,, & |\mathcal{N}_i| < K \\ 1\,, & |\mathcal{N}_i| \geq K \text{ and } \sigma_2 \leq \alpha\,\sigma_3 \\ 2\,, & |\mathcal{N}_i| \geq K \text{ and } \sigma_1 \leq \alpha\,\sigma_3 \text{ and } \sigma_2 > \alpha\,\sigma_3 \\ 3\,, & |\mathcal{N}_i| \geq K \text{ and } \sigma_1 > \alpha\,\sigma_3 \end{cases}. \tag{2.5}$$

Figure 15 show an example of the result of our labeling.

Figure 15 – Feature classification color encoding: the color encodes the boundary particles (blue) and the features particles in a water crown splash: small droplets (cyan), ligaments (yellow) and thin sheets (magenta).



Source: Adapted from Sandim *et al.* (2019).

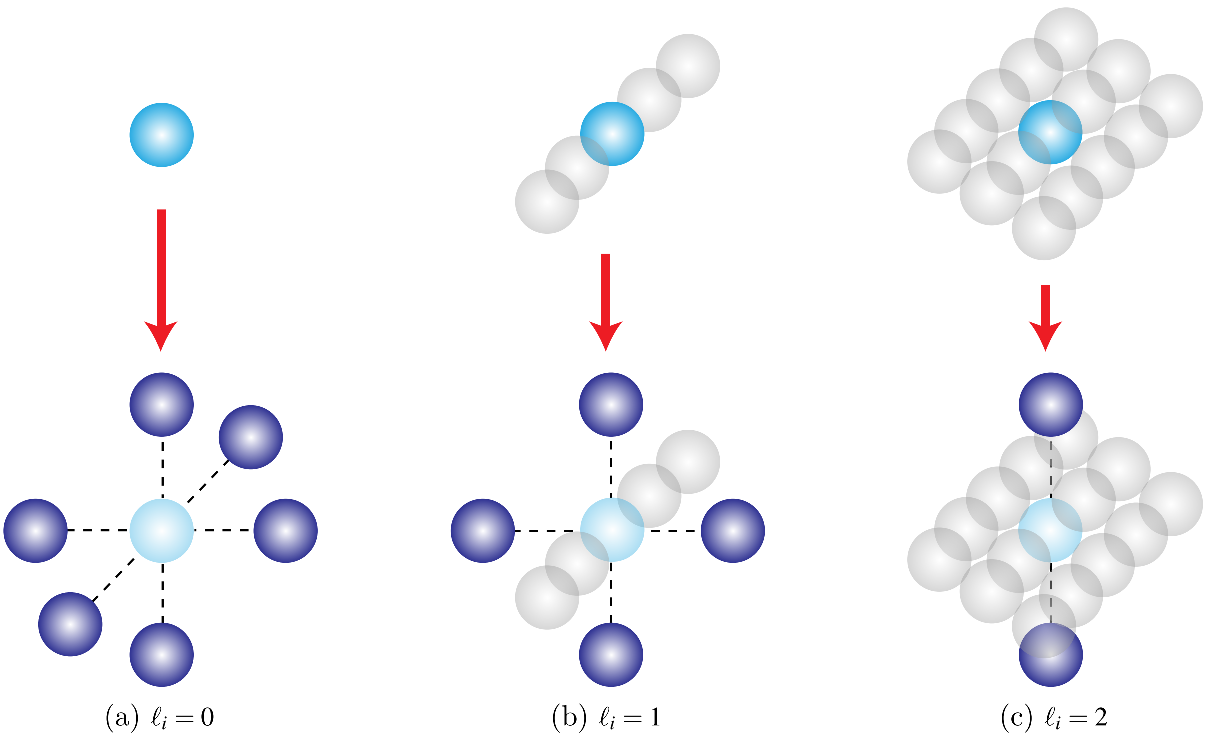
After labeling the particles, we resample the particles where $\ell_i \neq 3$. For each label:

[1] <http://eigen.tuxfamily.org>

- $\ell = 0$: we replace the particle with six new particles;
- $\ell = 1$: we replace the particle with four new particles;
- $\ell = 2$: we replace the particle with two new particles.

We define the positions of the new particles as shifts from the position of the original particles: eigenvectors extracted from U_i give the direction of the shift, and its size is chosen as a factor of $0.5r$. When $\ell = 0$ we can't use the eigenvectors, so we use the vectors that compose the standard basis as the directions. Figure 16 illustrate the resampling scheme for the three types of features.

Figure 16 – Particle refinement: a feature particle i (light blue) is replaced by new particles (dark blue) according to its label ℓ_i.

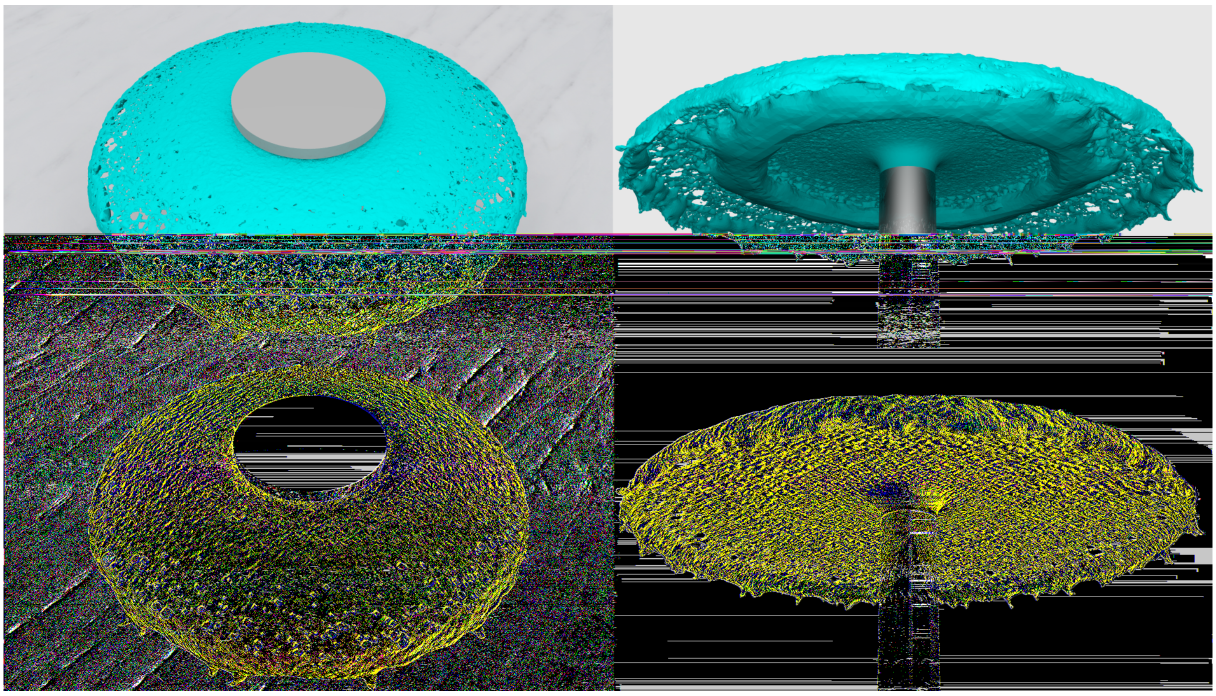

(a) $\ell_i = 0$ (b) $\ell_i = 1$ (c) $\ell_i = 2$

Source: Sandim *et al.* (2019).

This resampling scheme can introduce discontinuities on the boundary and, consequently, on the generated surfaces. We work around this problem by shifting the non-resampled boundary particles by a factor of $0.5r$ in the direction of their normal vectors.

The resulting surfaces are more consistent with the underlying particle system, as it does not lose features and does not create spurious features caused by incorrect normal alignment. Figure 17 gives an interesting example of the results; our method produces a smoother and more consistent surface on the top and eliminates a spurious feature from the bottom. The original paper (SANDIM *et al.*, 2019) contains more details of the method, results, and analysis.

Figure 17 – Mushroom jet simulation in a top (left column) and a bottom view (right column) at $t = 0.38$ sec. Surface reconstruction without (top row) and with (bottom row) our resampling in a simulation involving 2M particles.



Source: Sandim *et al.* (2019).

2.7 Discussion and Limitations

We have experimentally found that good results are reached when setting $\gamma = 1.3$ in the exponential flipping. We use this parameter value in all experiments presented in this thesis. However, misclassification can show up in concave regions with high curvature, resulting in false negatives. A theoretical upper-bound for the curvature (cf. Lemma 3.1 in (KATZ; TAL, 2013)) can be established in terms of γ and surface normals, i.e., HPR is guaranteed to produce correct results if the curvature is not larger than an upper-bound.

The parameter r defining the underlying grid size employed our BD method can also be tricky to be specified when information about particle spacing is not available. If particles are provided with no additional information, it is very hard to make any assumption about the topology of the surface and thus proper value to set r. We believe that is possible to establish a relation between r and the *local feature size* (AMENTA; BERN, 1998) of the points, but this is not a straightforward task.

2.8 Conclusion and Future Work

We presented a novel method to detect free-surface on particle-base volumes. The method relies on HPR visibility test and in the concept of r-sampling to identify particles laying on the boundary of a volume sampled by particles. Comparisons against state-of-the-art methods show the effectiveness of our methodology, which can be used together with most point set-surface reconstruction method to produce accurate surface models.

As future work, we would like to build our boundary detection method into mesh-free solvers, especially in simulations of physical phenomena involving complex free-surface phenomena. In particular, we believe that particle-based simulations involving surface tension (AKINCI; AKINCI; TESCHNER, 2013) and surface turbulence (MERCIER *et al.*, 2015) can greatly benefit from our methodology. In terms of rendering, we want to investigate the surface reconstruction in screen space using an image-based implementation of the HPR operator (SILVA *et al.*, 2014). Finally, since our method is built upon local visibility tests, adapting it to GPU architecture is feasible, being another direction for future work.

CHAPTER
3

SUPERVISED LEARNING FOR BOUNDARY DETECTION ON PARTICLE SYSTEMS

In particle-based physics simulations, the information about which particles belong to the boundary of the system and which are considered internal is, in general, an information that is useful but hard to obtain in antabu efficient way. This information can be applied to generate the free surface of the fluid or to compute the surface tension, among other applications. Techniques found in the literature may present satisfactory results, but in general they are sensible to the problem scale, particle distribution and involve computationally expensive operations such as matrix inversion. The goal of this study is to present an alternative with lower computational cost at runtime by transferring some of the work to a preprocessing step using offline learning.

3.1 Introduction

The area of Computational Fluid Dynamics (CFD) has received attention for years due to its application on many fields, ranging from industry to entertainment. Meshless methods such as Smoothed Particle Hydrodynamics (SPH) (GINGOLD; MONAGHAN, 1977), although very popular, still have a series of problems with suboptimal solutions. One of these problems is the detection of boundary particles (DILTS, 2000; SANDIM *et al.*, 2016; ORTHMANN *et al.*, 2013) This information is useful on other steps of a simulation such as the pressure computation on incompressible simulations, which needs a boundary condition to make its solution unique. Other uses are the application of surface tension, and visualization of the free-surface of the fluid.

An interesting approach to this problem that hasn't been explored on the Computer Graphics or Computational Physics literature, is the use of Machine Learning (ML) to solve a classification problem that separates the boundary particles from the internal ones. During a simulation step using the SPH method, many intermediary attributes are computed for each particle of the system, but in most cases they are discarded at the end of the step. The main idea of this article is to use these attributes along with a few others that can be easily computed to build a feature vector that represents the particle and ML to classify the particles. A recent study about this problem, using a different approach, was made recently by Sandim *et al.* (2016) and it can be used as a primer on the subject.

To successfully apply an supervised learning method a ground truth is needed, since supervised learning methods need labels in its training phase. This ground truth is computed trough a more complex and expensive method proposed by Dilts (2000) which uses a notion that can be interpreted as an r-boundary definition that will be discussed ahead. With the labels provided by Dilts' method and feature vectors for a representative subset of the data, we can train a classifier that can be applied to other particle systems with similar characteristics.

3.2 Supervised learning

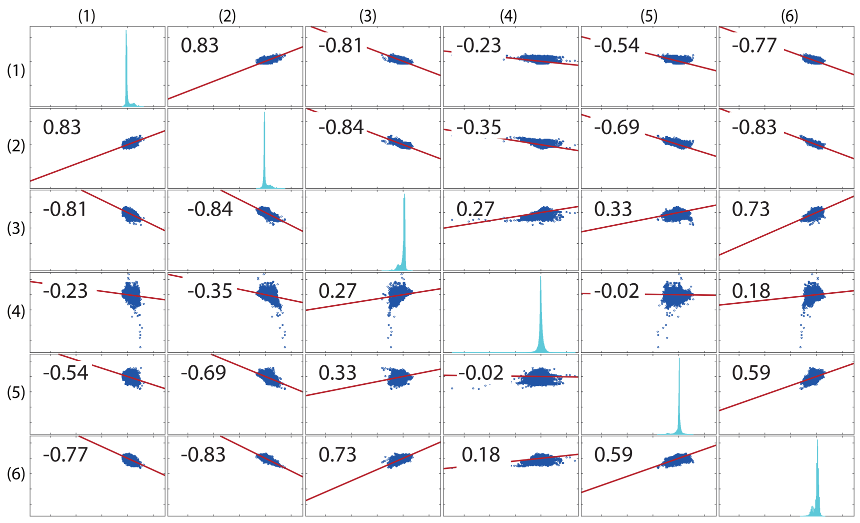
In a classification problem, supervised learning consists in feeding data with known labels to an algorithm so it can learn a mapping between the data and the labels. This mapping can then be used to assign labels to new data samples. In our case, the data comes from a set of features computed for each particle and the labels are, at first, unknown. In the next sections we'll discuss in detail how we obtain both the data and the labels used in the training step.

3.2.1 Feature vectors

During the simulation process of a particle system, most methods – like SPH – produce a lot of data for each particle of the system. In many cases a lot of this data is discarded at the end of the simulation step. This data is composed by some simple variables, like the number of neighbors inside the influence radius of a particle, some are more complex: the particle's velocity, density or pressure, and derivatives (gradient, divergent, curl) of some of them. Since the behavior of each particle is a result of these variables, we can use them to create a feature vector that represent the particle.

Two points arise from the idea of using this data on a feature vector: which variables are useful to separate the boundary from the interior, and how are they related to each other. Variables that don't carry any information about the surroundings of a particle aren't really useful when trying to separate the boundary of the system. The particle's position in $\mathbb{R}^n$, for instance, describes its location in space but doesn't say much about its position relative to other particles or the system itself. On the other hand, information that is derived from the surroundings of a particle are more useful for us. Particles on

Figure 18 – Plot of the correlation matrix of the chosen features: (1) modulus of the gradient of the density function, (2) divergence of the density function, (3) smallest eigenvalue of the covariance matrix, (4) largest eigenvalue of the covariance matrix, (5) density function, (6) number of neighbors inside influence radius.



Source: Sandim and Paiva (2020).

the boundary, or near it, tend to have fewer neighbors, lower density, it's density have a larger gradient, and so on.

The problem with some of these variables is that they are highly correlated, which makes them redundant. On Fig. 18 we show a correlation matrix created from a small sample of the data that can be extracted from particles. This is very useful to help us better understand this problem as it helps to identify and eliminate variables that add little to no information.

For instance, the density of a particle in the SPH method uses a density estimator based on the Parzen window (PARZEN, 1962) with an additional scale by the particle's mass: $\rho_i = \sum_{j\in\mathscr{N}_i} m_j W_{ij}$. Here ρ_i represents the density of the particle i, $\mathscr{N}_i$ the set of the neighbors (particles inside the influence radius) of i, m_j is the mass of j and W_{ij} is the value at the position of j of a kernel function, usually the gaussian kernel, centered on i. If all the particles have the same mass, it can be moved outside the summation and becomes just a scale: $\rho_i = m\sum_{j\in\mathscr{N}_i} W_{ij}$. Since the chosen features will be normalized in a further step, this scale by the mass is useless and can be removed. This leaves us with a simple density estimator based on the Parzen window, which is simpler and carries similar information.

The density function in the context of the SPH method is a bit noisy thanks to the discretization used and the sum of kernel functions. This variation can be useful to identify regions near the boundary of the system. The gradient of the density function is greater near the boundaries of the system, so the modulus of the gradient of density function is greater at the position of particles near or at the boundary of the system.

By using Principal Component Analysis (PCA) on the neighborhood of a particle, we can differentiate between particles near the boundary as they have very different eigenvalues. Particles inside the system have roughly the same eigenvalues as their neighborhood is evenly distributed around them.

The final set of features used to train a classifier capable of extracting the boundary of a system are: the density function, the modulus of its gradient and divergence, the number of neighboring particles inside the influence radius of the particle, and the smallest and largest eigenvalues of the covariance matrix of the positions of the particles on the neighborhood of each particle.

3.2.2 Ground truth and labels

A solid ground truth that can be used as labels for training is essential in training and testing a classifier. Here we discuss how we use the method proposed by Dilts (DILTS, 2000) to generate the labels that we use.

Consider a point set $\mathscr{P}$ that represent the positions of particles in a system, this set can be seen as a sampling of a closed region Ω that contains the system and has a surface $S = \partial\Omega$. As discussed by Sandim *et al.* (2016), a boundary point of $\mathscr{P}$ is a point that lies on S. Since it is quite hard to know if a point lies exactly on S, a more useful definition takes into account the sampling density of the set, as proposed by Sandim et al. By taking into account the sampling density, Sandim et al. give the definition of r-boundary points. These definitions match the method proposed by Dilts, so we can use this method to compute our labels. Albeit faster, the method proposed by Sandim et al. can't be applied here since it gives an approximation of the set of boundary particles of $\mathscr{P}$.

By applying directly the *arcs* method from Dilts (2000), we obtain a reliable set of labels that follow the definitions given by Sandim *et al.* (2016). These labels can then be used by us to train our classifier in order for it to identify r-boundary particles, which will be called only boundary particles for simplicity.

3.2.3 Training and testing

We chose to use a Support Vector Machine (SVM) classifier was used. This choice was based on the fact that its training phase may be costly but the classifying phase has linear complexity on the number of support vectors. This means that the majority of the computational effort is moved to the training phase of the method, and allowing the already trained classifier to be used inside the pipeline of a particle-based simulation.

The literature of Support Vector Machines dates back to 1964, with Vapnik and

Chervonenkis (1964) first introducing the idea of statistical learning, which is the basis for the SVM method. The SVM method itself was introduced by Boser, Guyon and Vapnik (1992) and ,from then on, many others extended the idea to adapt and use the SVM in different applications.

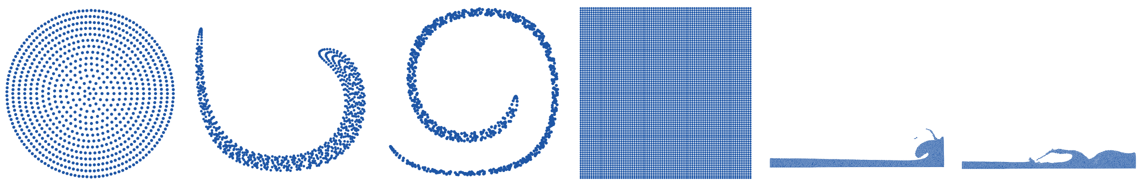
The training data is extracted from preprocessed SPH simulations, which contains the necessary information to compute the feature vector for each particle. All steps of the simulations used in training and testing have their labels computed according to Section 3.2.2. To build a training set that contains different cases from different configurations, we do the sampling in two phases: sampling of the time steps of the simulation and sampling of particles inside the sampled steps.

To sample the time steps of the simulation we use a random sampling of 30% of the available steps to be used as training, leaving the other 70% to testing. From the training steps we do a hold-out validation in which we chose again 30% of the particles from the step to be used as training and leaving the rest to testing. Note that the 30% of the particles chosen from a step are composed by roughly the same amount of internal and boundary particles to ensure that both cases are properly represented in the training data.

With this partitioning scheme we use only 9% of the available data as training, but if we consider that the total size of the datasets available from our SPH simulations are often around 100 steps with 500 thousand particles each, adding up to a total of 50 million different instances in a single simulation, the training set with 9% of this still has a total of 4.5 million samples. This training set is quite large and can be problematic to process in a conventional workstation with 8GB of RAM. The use of Sequential Minimal Optimization (SMO) to solve the optimization step of the SVM, proposed by Platt *et al.* (1999), reduces the memory needed during the training phase, since its memory requirements are linear to the training set size. An extended version of the SMO was made by Fan, Chen and Lin (2005).

The training is made using the SVM implementation from LIBSVM (CHANG; LIN, 2011), with an extended version of the SMO algorithm made by Fan, Chen and Lin (2005) and a Radial Basis Function (RBF) kernel. The choice of the RBF kernel over a linear was based on the fact that the groups are not linearly separable using the chosen features, preventing the convergence of the SMO algorithm in an acceptable number of iterations. After the training phase, the remaining data is used as testing and the confusion matrix is computed. Along with the confusion matrix we plot the particle system with the labels from the ground truth and the classifier's results, this plot shows not just the number of correct or wrong cases but also where they occur. The information of where the particles are classified on the wrong group is useful to understand the effect of the chosen features on the classifier.

Figure 19 – Sample steps from the training data.



Source: Sandim and Paiva (2020).

3.3 Results

The method was implemented using MATLAB, which uses LIBSVM as the back-end for its SVM tools. The tests were run in a workstation equipped with an Intel® Core™ i5-3570 processor and 8GB of RAM. The data used was obtained from SPH simulations made with DualSPHysics, which is an open source SPH simulation tool built and maintained by Crespo *et al.* (2015) and others. Another dataset is the single vortex spin by Bell, Colella and Glaz (1989), which is a common benchmark in the computational physics literature.

We trained a classifier using the information from the single vortex spin and a dam break simulation. Both start with a well behaved distribution but evolve to configurations with thin layers, sharp features, drops and holes. These features are fundamental to the training since the feature vectors of the particles in these regions carry the information needed to identify these structures. Fig. 19 shows a few steps of these datasets with the mentioned features. It is important to include extreme cases in the training data, with bad particle distribution, such as the tail of the system in the single vortex dataset. These cases are the ones that push the classifier to its limit.

The results obtained with the training data are consistent, even between runs with different partitioning of the data. This means that the partitioning scheme is successfully including representative samples into the training set. The confusion matrix on Table 3 shows a very low False Negative Rate (FNR) meaning that a low amount of boundary particles were wrongly classified as internal. A low FNR also indicates that the boundary has few or no holes. On the other hand, the False Positive Rate (FPR) is somewhat high, indicating that there is a considerable amount of internal particles being classified as boundary particles. This higher FPR may be a problem to some applications, but this can be solved by using a more expensive method such as Dilts' method (DILTS, 2000) to filter this cases and reduce the FPR.

After training and testing, we used a different simulation made with DualSPHysics to analyze the use of a classifier built with one simulation to classify another one and test the generalization of the model. In this case we used a double dam break scene, with a higher number of particles and finer discretization. The results were similar to the testing

Table 3 – Confusion matrix from the classification of the testing data.

		Predicted class	
		Internal	**Boundary**
Actual class	**Internal**	562,246	14,813
	Boundary	1,600	86,251

Table 4 – Confusion matrix from the classification of the double dam break data.

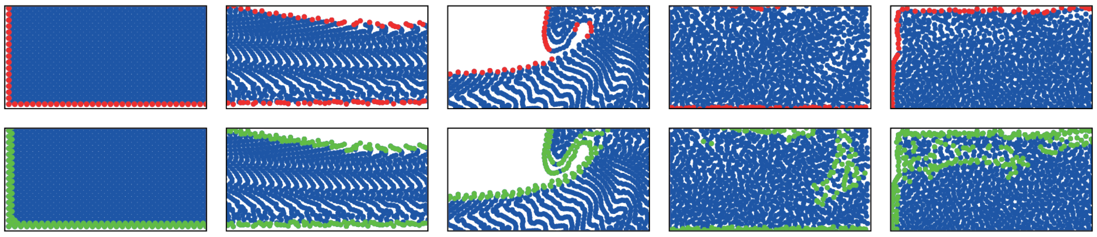
		Predicted class	
		Internal	**Boundary**
Actual class	**Internal**	3,917,101	285,354
	Boundary	11	204,090

Table 5 – Time taken in seconds by the classification of the double dam break data.

Method	**Total time (sec)**	**Avg time/step (sec)**
SVM	317.80	1.26
Reference	4,979.28	19.83

data but with a higher FPR as seen on Table 4, this is a grave issue that needs to be solved. On the other hand, the time to classify all the particle of each time step of the simulation using the SVM model is lower than the time needed to compute the reference method used as ground truth. After all 250 time steps were processed, the total time spent by the ML approach is considerably lower than the reference method. Table 5 shows the time in seconds spent by each technique to process the entire dataset. The geometric approach consumes more than 15 times more time to label the same data, this shows that while the approach using SVM may yield a high FPR it can be much faster than the geometric approach. Fig. 20 shows a few steps comparing the reference method with our proposal.

Figure 20 – Results of the classification of the double dam break simulation. The top row contains the results from the reference method, and the bottom row the results obtained by the proposed method. It is clear that the proposed method gives a closed boundary but has a high FPR.



Source: Sandim and Paiva (2020).

3.4 Conclusion

With this article we can conclude that it is possible to employ ML techniques to solve the boundary detection problem with low time overhead on the moment of classification. The running time of the proposed approach is more than 15 times lower than the reference method, this was one of the objectives and was successfully met. However the high FPR may turn the method unusable in some contexts where lower error rates are a requirement.

A third possibility, apart from a pure ML or pure geometric solution, is to combine both approaches in a two phase method: coarse phase using SVM and a fine phase using Dilts' method (DILTS, 2000). The second phase can be run only on the particles labeled as "boundary candidates" by the classifier used in the first phase. With this we may be able to achieve smaller running time than the reference method, but with FPR in acceptable levels.

CHAPTER

4

SIMPLE AND RELIABLE BOUNDARY DETECTION FOR MESHFREE PARTICLE METHODS USING INTERVAL ANALYSIS

We present two novel algorithms for detecting boundary particles in 2D and 3D domains that are suitable for meshfree particle methods in Computational Fluid Dynamics. We combine a robust purely geometric sphere covering test based on interval analysis with an adaptive spatial subdivision of the sphere associated with a given particle. The methods are simple, fast, and easy to code. We report comparisons against state-of-the-art boundary detection methods in free-surface flow problems to demonstrate the effectiveness and accuracy of our approaches.

4.1 Introduction

Meshfree methods, such as Smoothed Particle Hydrodynamics (SPH) (LIU; LIU, 2003), Moving Least-Squares Particle Hydrodynamics (MLSPH) (DILTS, 2000), and Moving Particle Semi-implicit (MPS) (KOSHIZUKA *et al.*, 2018), provide attractive numerical discretizations for a wide range of Computational Fluid Dynamics applications. In these methods, particles typically carry a kernel function for a local approximation of attributes, such as density and pressure, instead of keeping track of the connectivity between particles during the simulation. Besides, meshfree interpolations can also be applied to local subdomains to perform numerical integration like Galerkin type methods (FASSHAUER, 2007; NGUYEN *et al.*, 2008) or to achieve generalized finite difference stencils from scattered nodes (FLYER; BARNETT; WICKER, 2016).

An intrinsic advantage of meshfree methods over mesh-based methods is related to problems involving complex free-surface flows, efficiently capturing the topological changes (splitting and merging regions) that occur in the free surface, such as fragmentation, fracture, waves, splashing, and air bubbles creation. A delicate task for meshfree methods in handling boundary conditions at solid walls (e.g., no-slip and pressure Neumann condition) or across the free surface (e.g., constant pressure and kinematic condition), because such tasks demand accurate identification of *boundary particles*, that is, the particles that comprise the boundary of the fluid. This drawback is especially severe when solving problems where incompressibility must be satisfied (HOSSEINI; FENG, 2011) or surface tension has a meaningful effect on the flow behavior (LIN; LIU; WANG, 2019). Beyond boundary conditions, detecting boundary particles is essential also in applications comprising level-set definition (MARRONE *et al.*, 2010; SANDIM *et al.*, 2016) and particle shifting technology (PST) (WANG *et al.*, 2019).

Despite its importance, the problem of accurately detecting boundary particles has not been extensively addressed in the literature. To detect the "exact" boundary, Dilts (DILTS, 2000) provided a robust and reliable two-dimensional (2D) algorithm, where complex geometric predicates determine whether a particle is covered by its neighbor particles. Haque and Dilts (HAQUE; DILTS, 2007) extended that algorithm to three-dimensional (3D) particles. However, their extension is neither straightforward to implement nor efficient, since computing mutual covering of spheres is expensive. Lo and Shao (LO; SHAO, 2002) proposed criteria where an SPH particle is classified as boundary particle if its density is less than a certain threshold with respect to a reference density. He *et al.* (2012) and Liu, Lin and Shao (2014) presented a similar approach considering the SPH weighting of the particle distances. Although these methods involving SPH kernels are simple, they are inaccurate for free-surface flows with large deformations. Marrone *et al.* (2010) performed boundary/non-boundary particle classification using a pre-processing step based on the spectrum of the SPH correction matrix (RANDLES; LIBERSKY, 1996) combined with a geometrical test that verifies whether a neighbor particle lies in a conical region determined by a SPH approximation of normal vector. Barecasco, Terissa and Naa (2013) simplified that method by replacing the normal vector by a cover vector defined by a weighted average of the particle positions. However, both approaches (MARRONE *et al.*, 2010; BARECASCO; TERISSA; NAA, 2013) are sensitive to the non-uniformity of particle distribution due to the estimative of surface normals. Recently, Wang *et al.* (2019) presented an optimization of the algorithm by Marrone *et al.* (2010) just by changing the pre-processing step by the criteria regarding the SPH divergence of the particle positions as stated by Lee *et al.* (2008). Sandim *et al.* (2016) introduced an accurate and efficient geometrical test reinterpreting the original problem as a point-cloud visibility problem. Lin, Liu and Wang (2019) proposed an algorithm to detect boundary particles in 2D. Their method first computes a Delaunay triangulation of the neighbor particles projected on a unit circle, and then classifies a particle as boundary if any circumcircle radius exceeds a certain threshold. Since Lin, Liu and Wang (2019) provided no hints on extending their method to 3D, we believe that extension is not straightforward.

In this chapter, we present two novel algorithms for detecting boundary particles in 2D and 3D. Our approach is similar to the ones by Dilts and Haque (DILTS, 2000; HAQUE; DILTS, 2007), in that we perform a purely geometric sphere covering test. The key difference is that we use interval analysis methods to ensure robustness. The accuracy of our methods relies on an adaptive spatial subdivision of the sphere associated with a given particle. The first method performs an interval evaluation of the implicit function defining the sphere for each particle. The second method uses geometric predicates on the enclosures of the boundary of the sphere. Both methods are robust: they do not produce false negatives; that is, no boundary particles are classified as interior. Moreover, the methods are simple, easy to code, and with competitive computational times. We perform a set of comparisons against state-of-the-art boundary detection methods in free-surface flow problems to demonstrate the effectiveness and accuracy of our approaches.

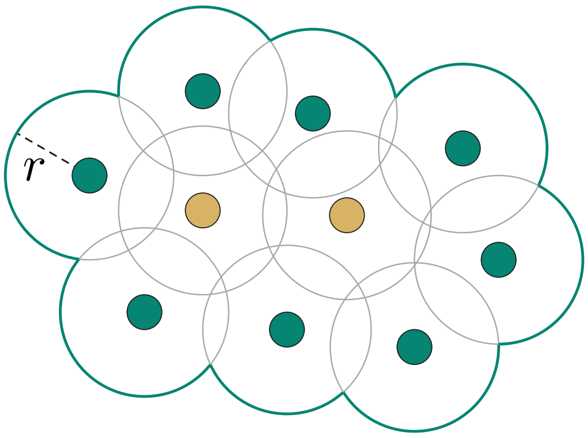
4.2 Boundary particles

Let $\mathscr{P}$ be a set of scattered particles sampling a compact region $\Omega \subset \mathbb{R}^d$ and let $\partial\Omega$ be the boundary surface of Ω. We index the particles in $\mathscr{P}$ by $i \in \mathbb{N}$. Particle i is located at the point $\mathbf{p}_i \in \mathbb{R}^d$. Our main goal is to identify the boundary particles in $\mathscr{P}$, that is, the set of particles in $\mathscr{P}$ that lie on $\partial\Omega$. Ideally, we would like to identify all boundary particles precisely, but we shall aim for a large subset of particles that are guaranteed to be on the boundary.

To define boundary particles precisely, we assume that $\mathscr{P}$ is an *r-sampling* of Ω (KATZ; TAL; BASRI, 2007): for every point $\mathbf{x} \in \Omega$, there is a particle i in $\mathscr{P}$ such that $\|\mathbf{x} - \mathbf{p}_i\| < r$. The parameter r corresponds to the numerical resolution of the problem. For instance, in SPH solvers the radius r coincides with the SPH smoothing length (HAQUE; DILTS, 2007). Thus, the radius r dictates the accuracy of the method: the boundary detection method should be able to capture small-scale details of the fluid (like cavities, thin-sheets, ligaments, and drops) of diameter or thickness at least $2r$.

Let B_i be the ball of radius r centered at the point $\mathbf{p}_i$ and let $S_i = \partial B_i$ be the boundary sphere of B_i. A particle i is called an *interior particle* when its sphere S_i is completely covered by the neighboring balls; more precisely, when $S_i \subset \cup_{j \in \mathscr{N}_i} B_j$ where $\mathscr{N}_i = \{j \in \mathbb{N} : \|\mathbf{p}_j - \mathbf{p}_i\| \leq 2r\}$. Note that it is the boundary S_i of B_i that is completely covered other balls, not necessarily the whole ball B_i (Figure 21). A particle i is called a *boundary particle* when it is not an interior particle.

Figure 21 – Interior particles (brown dots) and boundary particles (green dots).



Source: Sandim, Paiva and Figueiredo (2020).

4.3 Boundary detection using interval analysis

To determine whether a particle is on the boundary, we perform a purely geometric covering test, following the definition above. This approach is similar to the ones by Dilts and Haque (DILTS, 2000; HAQUE; DILTS, 2007). The contribution of this chapter is two robust and efficient solutions for this geometric problem. They rely on adaptive spatial subdivision and robust geometric tests. There is a single user parameter, the subdivision depth, which controls the tradeoff between speed of processing and accuracy of the results.

Our methods use tools from interval analysis: interval arithmetic and geometric enclosures. The first method is easier to understand and to implement. It also serves as an introduction to the second method, which deals with slightly more complicated geometry. Recall that our main task is deciding whether particle i is an interior particle. By definition, we have to check whether the boundary S_i of B_i is completely covered by neighboring balls B_j for $j \in \mathcal{N}_j$.

4.3.1 *Using interval arithmetic*

In our first method, we perform an adaptive spatial subdivision of the bounding box of B_i into *query boxes*. We test the query boxes that intersect S_i against the neighboring balls B_j. The geometric tests in this method rely on the implicit formulation of balls:

$$B_j = \{\mathbf{x} \in \mathbb{R}^d : f_j(\mathbf{x}) \leq 0\}, \quad \text{where} \quad f_j(\mathbf{x}) = \|\mathbf{x} - \mathbf{p}_j\|^2 - r^2$$

We test a query box Q against a ball B_j by computing the interval $I = f_j(Q)$. We compute this interval exactly using *interval arithmetic* (MOORE R. BAKER KEARFOTT, 2009)[1].

[1] Interval arithmetic is a numerical technique that provides estimates for the whole range of values taken by a function in a box in $\mathbb{R}^d$. For the functions f_j defining spheres, it happens that the estimates computed by interval arithmetic are exact.

There are three possible outcomes of this computation, according to the position of the bounds of I, given by $\min I$ and $\max I$, with respect to 0:

- $\max I \leq 0$: Then $I \subseteq [-\infty, 0]$ and Q is *completely inside* B_j.
- $\min I > 0$: Then $I \subseteq (0, \infty)$ and Q is *completely outside* B_j.
- $\min I \leq 0 \leq \max I$: Then Q *straddles the boundary* S_j of B_j.

The query boxes that straddle the boundary S_i of B_i are called *boundary boxes*. They are found by testing the interval $f_i(Q)$. Boundary boxes are the interesting query boxes; the other ones are discarded. We test whether a boundary box Q is completely covered by neighboring balls B_j using the same test on the interval $f_j(Q)$. There are three possible outcomes:

- Q is completely inside B_j for *some* j: Then the part of S_i inside Q is completely **covered** by B_j.
- Q is completely outside B_j for *all* j: Then the part of S_i inside Q is completely **uncovered** by the neighboring balls.
- Otherwise, Q is **partially covered**. Then, Q is subdivided by bisection at its center into 2^d children boxes, the test is applied recursively to the children, and the results are combined.

If all boundary boxes are completely covered, then particle i is an interior particle. Otherwise, particle i is a boundary particle.

The function `is_interior` provided by Algorithm 2 implements these ideas: it performs an adaptive spatial subdivision of the sphere associated to particle i, starting the recursion with the bounding box of B_i, that is, the cube centered at $\mathbf{p}_i$ with side $2r$. The algorithm follows a subdivision tree (a quadtree in 2D and an octree in 3D), trying to classify query boxes according to the criteria above, up to a maximum depth chosen by the user to control the tradeoff between speed and accuracy.

Figure 22 illustrates this algorithm in 2D using a quadtree with maximum depth 3. In this case, the algorithm classifies particle i as a boundary particle without having to process the whole quadtree. For this, we assume that the conjuction in `subdivide` is short-circuited, that is, stops at the first term that is false. (The algorithm is correct even if this assumption does not hold; it just processes more query boxes.)

We shall now discuss some important implementation details.

Algorithm 2 – Recursive classification of a particle i

```
 1: function ISINTERIOR($i$)
 2:     $Q \leftarrow$ bounding box of $B_i$
 3:     return QUERY($Q, i, 0$)
 4: end function
 5: function QUERY($Q, i, depth$)
 6:     if $Q$ is completely inside $B_i$ or $Q$ is completely outside $B_i$ then
 7:         return true                                        ▷ uninteresting boxes
 8:     end if
 9:     $uncovered \leftarrow true$
10:     for all $j \in \mathscr{N}_i$ do
11:         if $Q$ is completely inside $B_j$ then
12:             return true
13:         end if
14:         if $Q$ straddles the boundary $S_j$ of $B_j$ then
15:             $uncovered \leftarrow false$
16:         end if
17:     end for
18:     if uncovered then
19:         return false                                       ▷ definitely boundary
20:     end if
21:     if depth = maximum depth then
22:         return false                                       ▷ probably boundary
23:     end if
24:     return SUBDIVIDE($Q, i, depth$)
25: end function
26: function SUBDIVIDE($Q, i, depth$)
27:     $Q = Q_1 \cup \cdots \cup Q_{2^d}$                                 ▷ subdivision
28:     return QUERY($Q_1, i, depth+1$) and ... and QUERY($Q_{2^d}, i, depth+1$)
29: end function
```

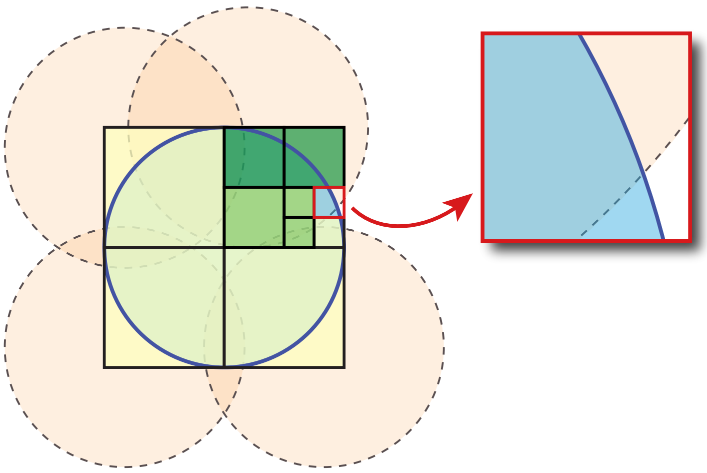
Finding neighboring particles

Crucial to the performance of the algorithm is a fast way to identify the set $\mathscr{N}_i$ of neighboring particles. For this task, we use the *linked-list* algorithm (LIU; LIU, 2003), where the cells of the search grid have a size of $2r$.

Faster tests

To test whether a query box Q is a boundary box, we skip interval arithmetic and test the signs of f_i on the vertices of Q. Then Q is a boundary box iff the signs are not the same. This change does not affect correctness because query boxes are subdivisions of the bounding box of B_i and so are in special positions with respect to B_i. In particular, a boundary box never has all of its vertices outside B_i. The one exception is the bounding box of B_i, which is the initial query box. We avoid this case by starting the recursion at

Figure 22 – The 2D version of our boundary detection method using interval arithmetic. The ball B_i around particle i is shown in light blue. We check whether the circle S_i (dark blue) is covered by its neighboring balls (orange). Some boundary boxes are not processed (yellow), and some are covered by a neighbor ball (dark green), while the non-boundary boxes (light green) are discarded by our method. A query box (red) at depth 3 reveals an uncovered region of S_i. Thus, particle i is a boundary particle.



Source: Sandim, Paiva and Figueiredo (2020).

depth 1, by changing `is_interior` slightly to call `subdivide` instead of `query`, as shown in Algorithm 3:

Algorithm 3 – Modified recursive classification of a particle i

```
1: function ISINTERIOR($i$)
2:     $Q \leftarrow$ bounding box of $B_i$
3:     return SUBDIVIDE($Q, i, 0$)
4: end function
```

This change improves overall performance by about 16%. On the other hand, we cannot use the signs of f_j on the vertices of Q to test Q against neighboring balls B_j because their relative positions are arbitrary. We must rely on interval arithmetic for those tests.

Interval arithmetic libraries

For computations with intervals, we used PyInterval[2] in 2D and Boost C++[3] in 3D, both easy-to-use libraries. For the simple case of a sphere equation, we could avoid a full interval arithmetic library and use the simple ad-hoc code given in the Appendix. We could also probably avoid using outward rounding, because the geometric resolution is much lower than the numerical resolution of the floating point system.

[2] <https://pypi.org/project/pyinterval/>
[3] <https://www.boost.org/doc/libs/1_66_0/libs/numeric/interval/doc/interval.htm>

Avoiding recursion

While Algorithm 2 is easy to understand and to check correctness, our actual implementation in Algorithm 4 simulates recursion by keeping query boxes that need to be checked in a stack. This makes it easier to terminate the process earlier without changing the final result, by following the simple rules below:

R1: If Q is uncovered, we can safely say that S_i has an uncovered region and thus the particle i is a boundary particle;

R2: If Q is covered, we stop the subdivision at Q;

R3: If Q is partially covered and the maximum depth has not been reached, we subdivide Q and continue recursively;

R4: If the maximum depth has been reached, we declare that the particle i is a boundary particle. These are the potential false positives, i.e., interior particles misclassified as boundary particles.

Rules **R1** and **R2** have precedence over **R3** and **R4**. If **R2** is satisfied, we must keep evaluating the sibling nodes until we find an uncovered node or determine that all nodes are covered, in which case that the particle i is interior. In **R4**, although Q is partially covered, we classify the particle i as boundary particle for two reasons: (i) we ensure that the algorithm will never produce a false negative and (ii) it allows us to a short circuit the evaluation process and thus speeds up the detection.

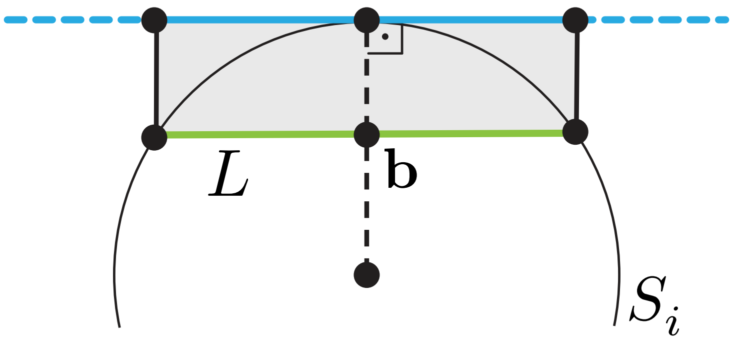
Whenever we need a box to evaluate, we get the one on the top of the stack; when we subdivide a box, we push its children ton top of the stack. The simulated depth-first traversal is beneficial since we can reach rules **R1** and **R4** quicker and terminate the process earlier. If we do not arrive at **R1** or **R4**, and we reach the bottom of the stack, then all boxes fell on rule **R2**, and thus they are all covered; consequently, the particle i in an interior particle.

4.3.2 *Using geometric enclosures*

Our first method is simple to understand and to implement, but it spends effort classifying query boxes against a ball B_i: it finds and discards boxes that are completely inside or completely outside B_i. However, we care only about the boundary S_i of B_i and the only interesting boxes are the boundary boxes.

Our second method performs an adaptive spatial subdivision of an enclosure of S_i, a process that takes place in a lower dimension. We decompose S_i into a coarse mesh of linear elements and enclose S_i around that mesh with a union of thin convex polytopes which we shall call *slabs*. This cover is refined as needed by adaptively refining the underlying mesh.

Figure 23 – Slab (gray) enclosing a segment L (green).



Source: Sandim, Paiva and Figueiredo (2020).

In 2D, the mesh is initially given by the sides of an equilateral triangle inscribed in S_i. The mesh is then adaptively refined into an inscribed polygon. The slab around a mesh segment L is the rectangle having one side on L and the opposite side tangent to the circle S_i at the projection of the midpoint $\mathbf{b}$ of L onto S_i (see Figure 23). To test a segment L against a neighboring ball B_j, we test the corresponding slab Q against B_j by testing the signs of f_j at the vertices of Q, as follows:

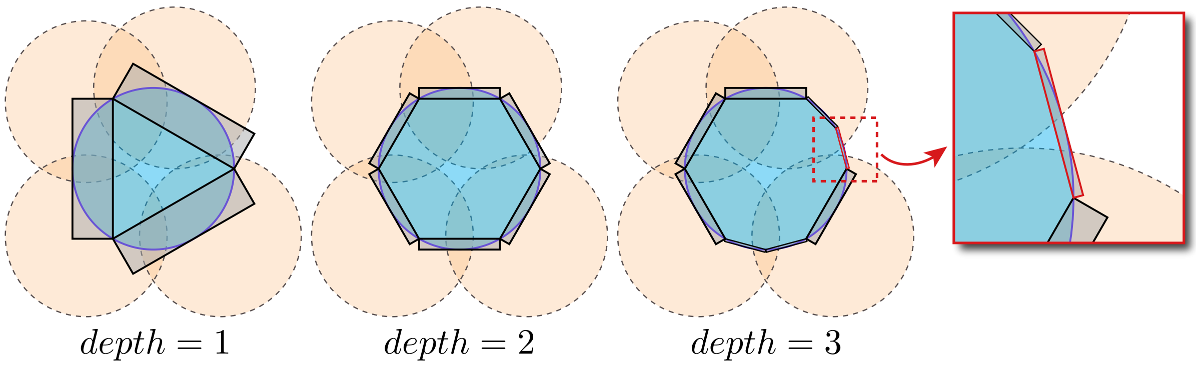
- If $f_j(\mathbf{v}) < 0$ for all vertices $\mathbf{v}$ of Q: Then Q is *completely inside* B_j.
- If $f_j(\mathbf{v}) > 0$ for all vertices $\mathbf{v}$ of Q: Then Q is *completely outside* B_k.
- Otherwise, Q *straddles the boundary* S_j of B_j.

As in the first method, there are three possible outcomes:

- Q is completely inside B_j for *some* j: Then L is completely **covered** by B_j.
- Q is completely outside B_j for *all* j: Then L is completely **uncovered** by the neighboring balls.
- Otherwise, L is **partially covered**. Then, L is subdivided at its midpoint into two children segments, the test is applied recursively to the children, and the results are combined.

If all segments are completely covered, then particle i is an interior particle. Otherwise, particle i is a boundary particle. Figure 24 illustrates our second method in 2D.

Figure 24 – The 2D version of our method based on geometric enclosures. At each level, we check whether the slabs (gray) are covered by its neighboring balls (orange). At maximum depth 3, a slab (red) shows an uncovered region of S_i. Thus, the particle i is a boundary particle.



Source: Sandim, Paiva and Figueiredo (2020).

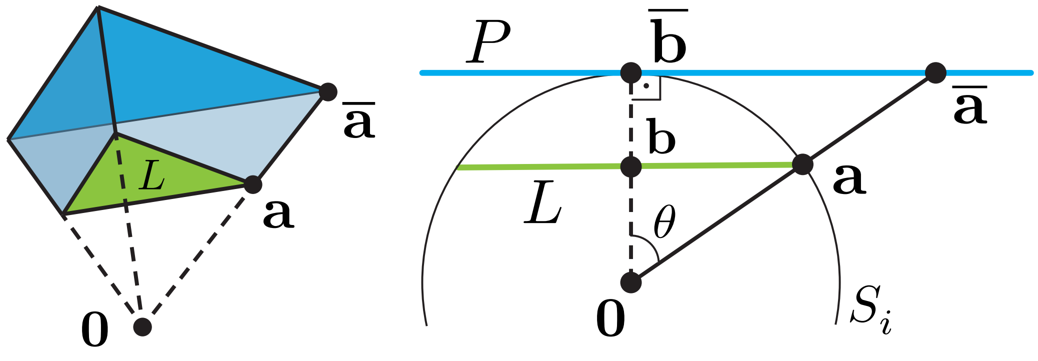
Computing slabs in 3D

Firstly, we apply the affine transformation $T_i(\mathbf{x}) = r^{-1}(\mathbf{x} - \mathbf{p}_i)$ in the sphere S_i and its neighboring balls. Note that the center of S_i is translated for the origin $\mathbf{0}$ and all balls become unit balls.

In 3D, the linear elements are triangles; the mesh is initially a regular tetraheadron inscribed in S_i. Let L be a mesh triangle and $\mathbf{b}$ its barycenter. The slab around L is the frustum obtained by projecting L onto the plane P that is tangent to S_i at the projection $\overline{\mathbf{b}} = \mathbf{b}/\|\mathbf{b}\|$ of $\mathbf{b}$ onto S_i (see Figure 25).

Each vertex $\mathbf{a} \in L$ is projected onto P to mimic a perspective projection from the viewpoint $\mathbf{0}$. Considering $\overline{\mathbf{a}}$ as the projection of $\mathbf{a}$ onto P, by construction as shown in Figure 25, we have that $\mathbf{a} \cdot \overline{\mathbf{b}} = \cos\theta = \|\overline{\mathbf{a}}\|^{-1}$. Thus, the projected vertex is given by $\overline{\mathbf{a}} = \mathbf{a}/(\mathbf{a} \cdot \overline{\mathbf{b}})$.

Figure 25 – Perspective projection of a triangle L (green) onto the tangent plane P (blue) of the unit sphere S_i: the frustum formed by L and its projection (left) and the projection of a single vertex $\mathbf{a} \in L$ on the plane P (right).

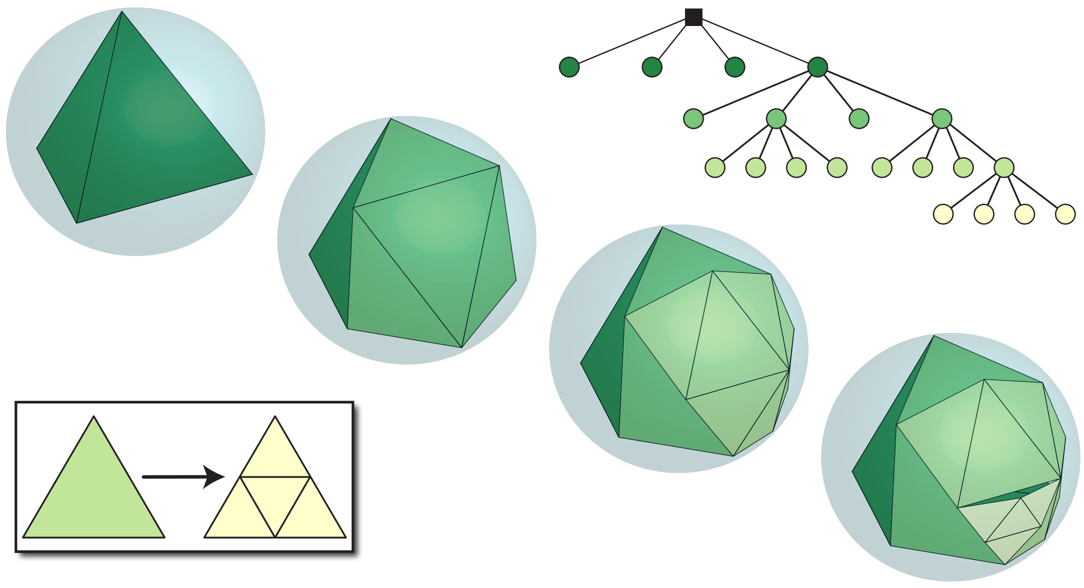


Source: Sandim, Paiva and Figueiredo (2020).

The mesh triangle L is refined when necessary by using the standard midpoint subdivision scheme (see Figure 26). Then, we update the location of these new vertices by projecting them on S_i. Although the refined mesh provides a better approximation of S_i, this process can produce hanging nodes and so a non-conforming mesh. However, this is not a problem because we are not interested in the resulting mesh itself, only in the slabs that enclosure the boundary.

The mesh refinement process rapidly reduces the total volume of the enclosure because the combined volumes of the slabs associated with the four child nodes will be a fraction of the volume associated with the original parent node. So each subdivision step improves the accuracy of the covering test, consequently its convergence.

Figure 26 – Mesh refinement in 3D (center) and its underlying quadtree (top-right): a regular tetrahedron inscribed in S_i is refined using midpoint subdivision scheme (bottom-left).



Source: Sandim, Paiva and Figueiredo (2020).

We apply the rules **R1**–**R4** of the previous method in each slab. Assuming that a slab is a boundary box, the labeling of a generic element (box or slab) regarding its covering is summarized in Algorithm 4.

4.4 Results

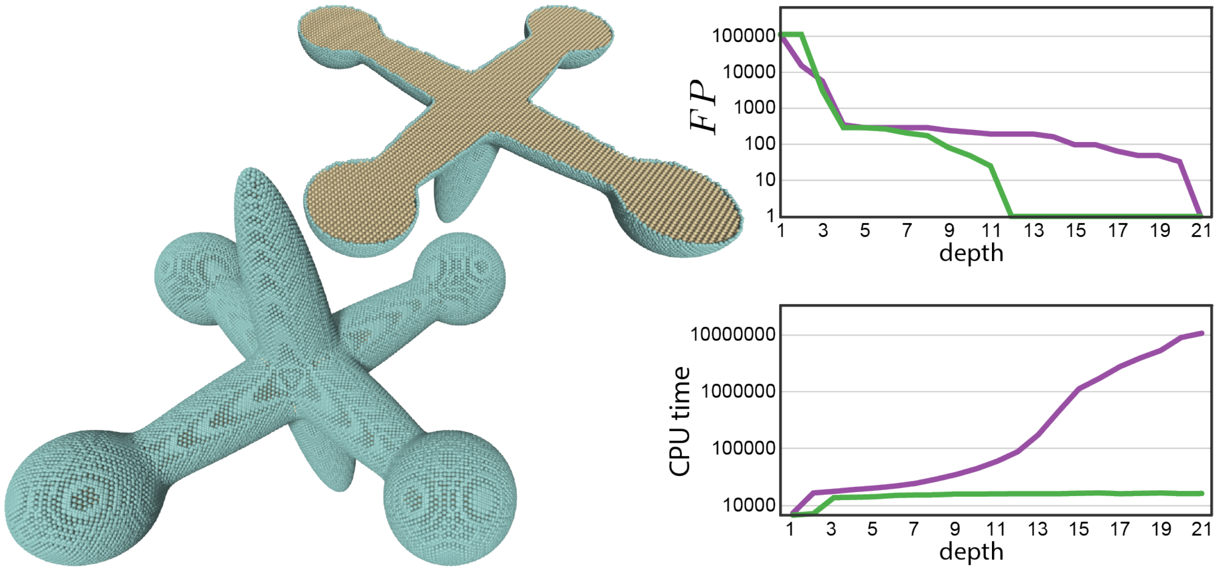
We first analyze the performance of our novel techniques based on interval arithmetic (IA) and geometric enclosure (GE). Assuming the classification provided by Dilts and Haque (DILTS, 2000; HAQUE; DILTS, 2007) as ground truth, we count the number of particles according to their assignment:

Algorithm 4 – Classification of a particle i

```
function ISINTERIOR($i$)
    initialize the stack $\mathscr{T}$ with the nodes from the first tree-level
    while $\mathscr{T} \neq \emptyset$ do
        $Q \leftarrow$ the topmost element from $\mathscr{T}$
        remove $Q$ from $\mathscr{T}$
        $q \leftarrow$ LABEL($Q, i$)
        if $q$ is uncovered then
            return false
        else if $q$ is covered then
            continue                                  ▷ proceed to the next element of $\mathscr{T}$
        else                                          ▷ $Q$ is partially covered
            $\ell \leftarrow$ the depth of $Q$
            if $\ell <$ maximum depth then
                $Q = Q_1 \cup \cdots \cup Q_m$        ▷ subdivision
                for all child node $Q_k$ do
                    if $Q_k$ is boundary box then
                        add $Q_k$ to the top of $\mathscr{T}$ with depth $\ell + 1$
                    end if
                end for
            else
                return false
            end if
        end if
    end while
    return true
end function
function LABEL($Q, i$)
    $q \leftarrow$ uncovered
    for all $j \in \mathscr{N}_i$ do
        if $Q$ is completely outside $B_j$ then
            continue                                  ▷ keep the current label
        else                                          ▷ either covered or partial
            if $Q$ is completely inside $B_j$ then
                return covered
            end if$q \leftarrow$ partial
        end if
    end for
    return q
end function
```

- *True Positive* (TP): a boundary particle correctly classified;
- *True Negative* (TN): an interior particle correctly classified;
- *False Positive* (FP): an interior particle classified as boundary;
- *False Negative* (FN): a boundary particle classified as interior.

Figure 27 – Comparison between IA (🟪) and GE (🟩) methods. On the left, the classification of the boundary particles (green) and interior particles (brown) in a model with 170k particles, and a cutaway view thereof. On the right, the symmetric log (symlog) plots of the FP and the computational times (in milliseconds) w.r.t. the subdivision depth.



Source: Sandim, Paiva and Figueiredo (2020).

Since our interval approaches do not produce false negatives (i.e., $FN = 0$), the comparison between IA and GE approaches is performed by analyzing the number of false positives. Figure 27 shows an error analysis based on FP in a static model and a comparison of the computational performance between both approaches as well. We observe that GE approach converges to the exact classification faster than IA with less computational effort.

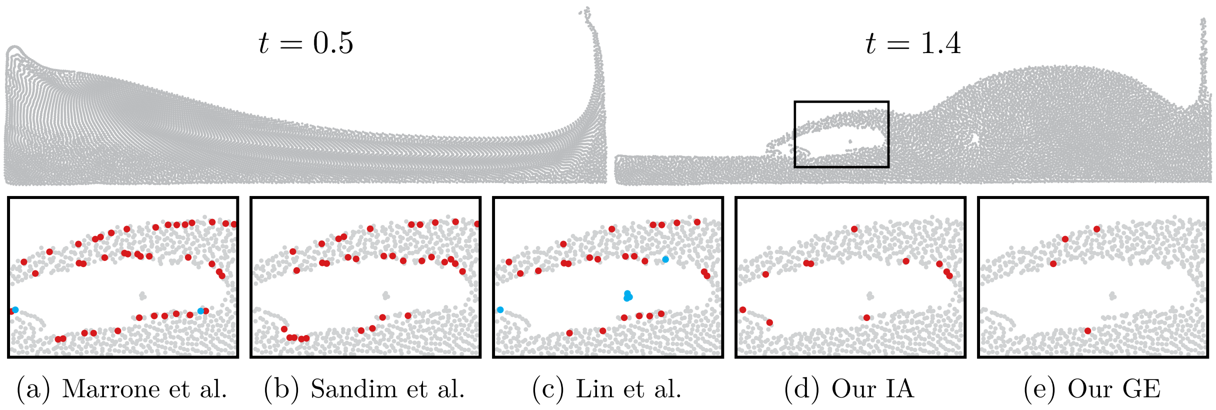
In search of a balance between accuracy and efficiency in IA and GE methods, we choose the maximum depth of 6 in all experiments carried out below.

Figure 28 provides a qualitative comparison by showing the misclassified particles (FP and FN) resulting from each technique in an SPH simulation of a 2D dam-break problem, as described in (MARRONE *et al.*, 2010). The zoomed rectangle in this figure shows that our geometric approaches can capture cavities and thin sheets of fluid better than any other detection method due to the reduced number of misclassified particles.

Figure 29 shows the behavior of the particles in the classical *single vortex* experiment, as detailed by Enright *et al.* (2002). Our IA method detects the boundary particles gracefully, even when the particles are compressed due to a considerable stretching caused by the vortex flow field.

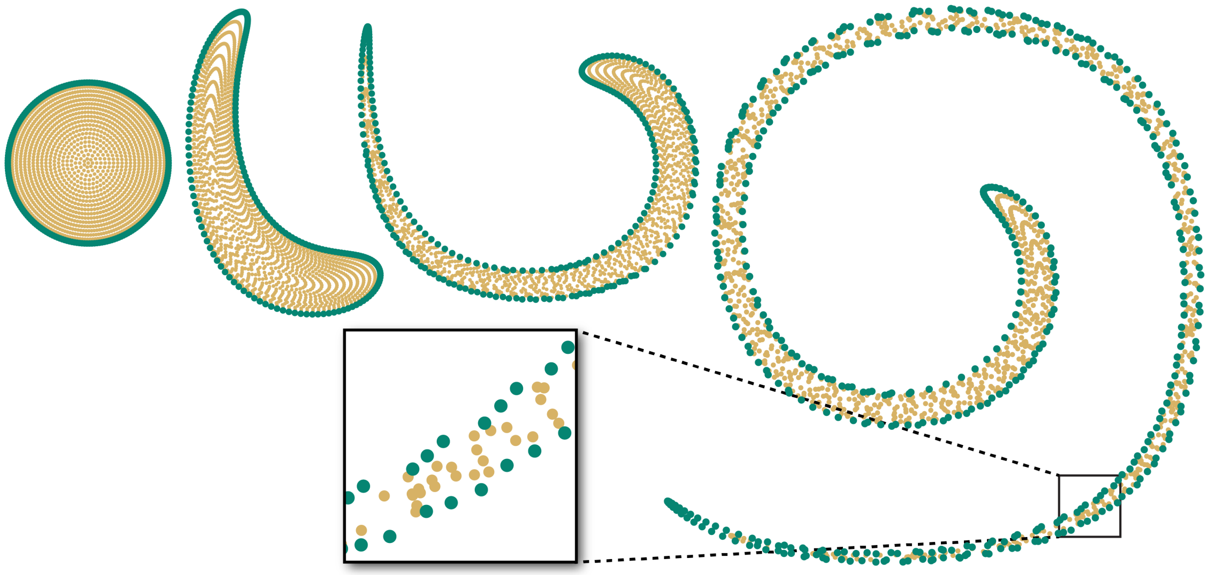
Figure 30 shows an SPH flow simulation of a liquid been injected in a domain with an obstacle. Our GE method captures the bubbles formed by the impact of the liquid

Figure 28 – Comparison between different boundary detection methods in a 2D dam-break simulation using SPH (top) at $t = 1.4$. The misclassified particles are highlighted (bottom): FP (red) and FN (blue).



Source: Sandim, Paiva and Figueiredo (2020).

Figure 29 – Boundary detection using IA in a single vortex at different times $t = 0.0, 0.25, 0.5$ and 1.0 (from left to right): boundary particles (green) and interior particles (brown).



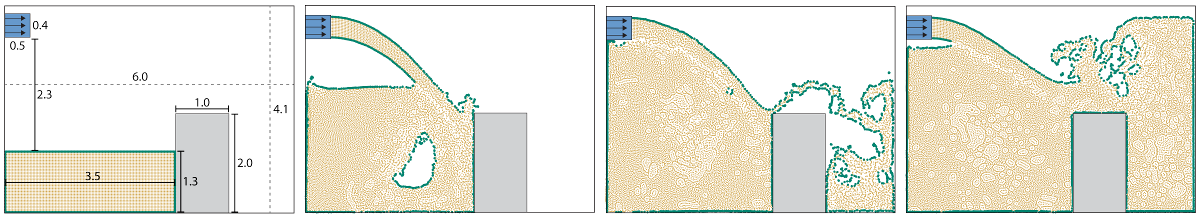
Source: Sandim, Paiva and Figueiredo (2020).

against a rigid obstacle.

Figures 31 and 32 depict the boundary detection using IA and GE methods in complex free-surface flows in 3D resulting from the impact of a double dam-break and against a rigid tall obstacle after a single dam-break, respectively. Our geometric methods are resilient to the fragmentation of the interface and the thin layers of particles created by the impact.

All SPH flow simulations presented in this section were performed using the com-

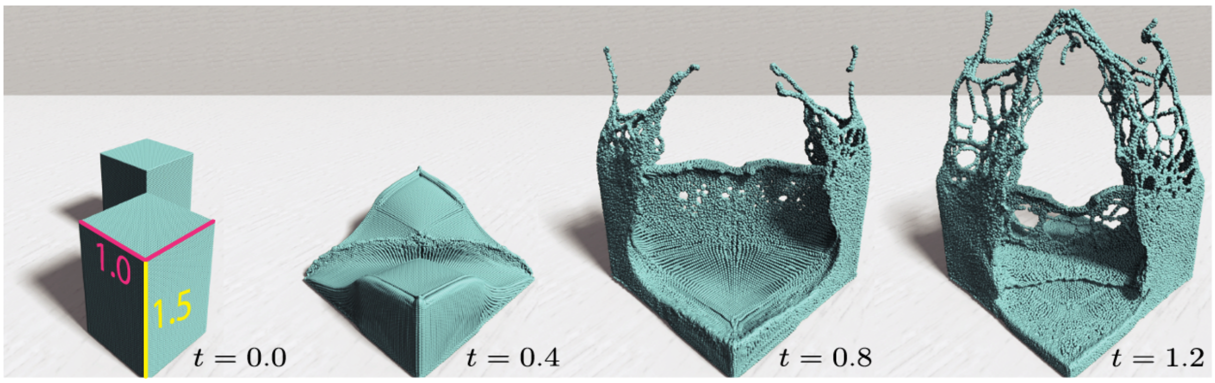
Figure 30 – Boundary detection using GE in an SPH flow simulation of an injector (blue) with inlet velocity of $4m/s$ in a domain with an obstacle (gray) at different times (t): boundary particles (green) and interior particles (brown).



(a) $t = 0.00$ (b) $t = 2.48$ (c) $t = 4.95$ (d) $t = 7.43$

Source: Sandim, Paiva and Figueiredo (2020).

Figure 31 – Boundary particles detected by our IA method in an SPH flow simulation of a dam-break problem of two liquid columns at opposite corners of a container with a square base of size 2.5 and height 3.5 at different times (t).



Source: Sandim, Paiva and Figueiredo (2020).

putational platform SPHysics (GOMEZ-GESTEIRA *et al.*, 2012).

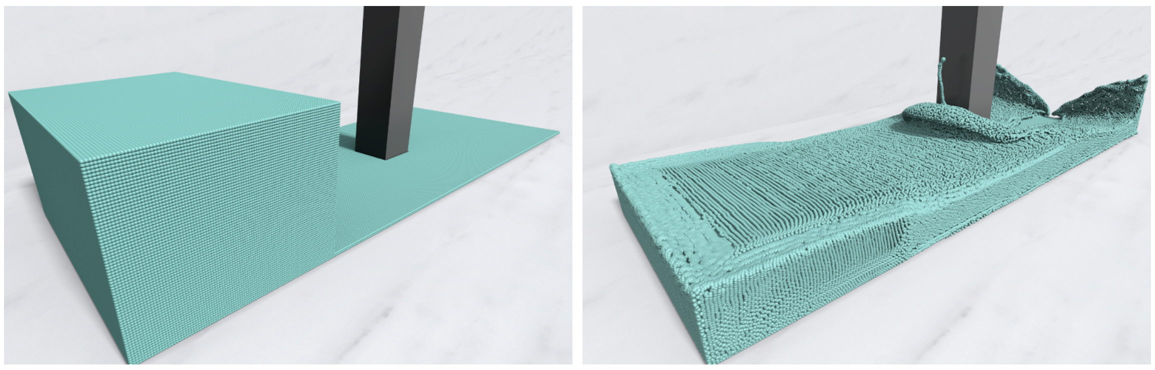
Beyond SPH, Figure 33 shows the versatility of our GE method when applied to detect the boundary particles in an *Affine Particle-in-Cell* (APIC) (JIANG; SCHROEDER; TERAN, 2017) flow simulation of a liquid sloshing in a spherical tank. In this experiment, we use the APIC implementation provided by Kim (2016).

Similar to the level-set definition from the boundary particles introduced by Marrone *et al.* (2010), Figure 34 demonstrates the effectiveness of our geometric approach when the free-surface is reconstructed from the level-set of the *Enright test* (ENRIGHT *et al.*, 2002) generated by Sandim et al.'s algorithm (SANDIM *et al.*, 2019).

4.4.1 Quantitative analysis

To perform quantitative comparisons between different boundary detection methods, we measure the accuracy of each method using the metric proposed by Sandim *et al.*

Figure 32 – Boundary particles detected by our GE method in an SPH flow simulation of the impact against a rigid obstacle after a dam-break, as reported in (MARRONE *et al.*, 2010).



(a) $t = 0.0$ (b) $t = 0.2$

Source: Sandim, Paiva and Figueiredo (2020).

(2016):

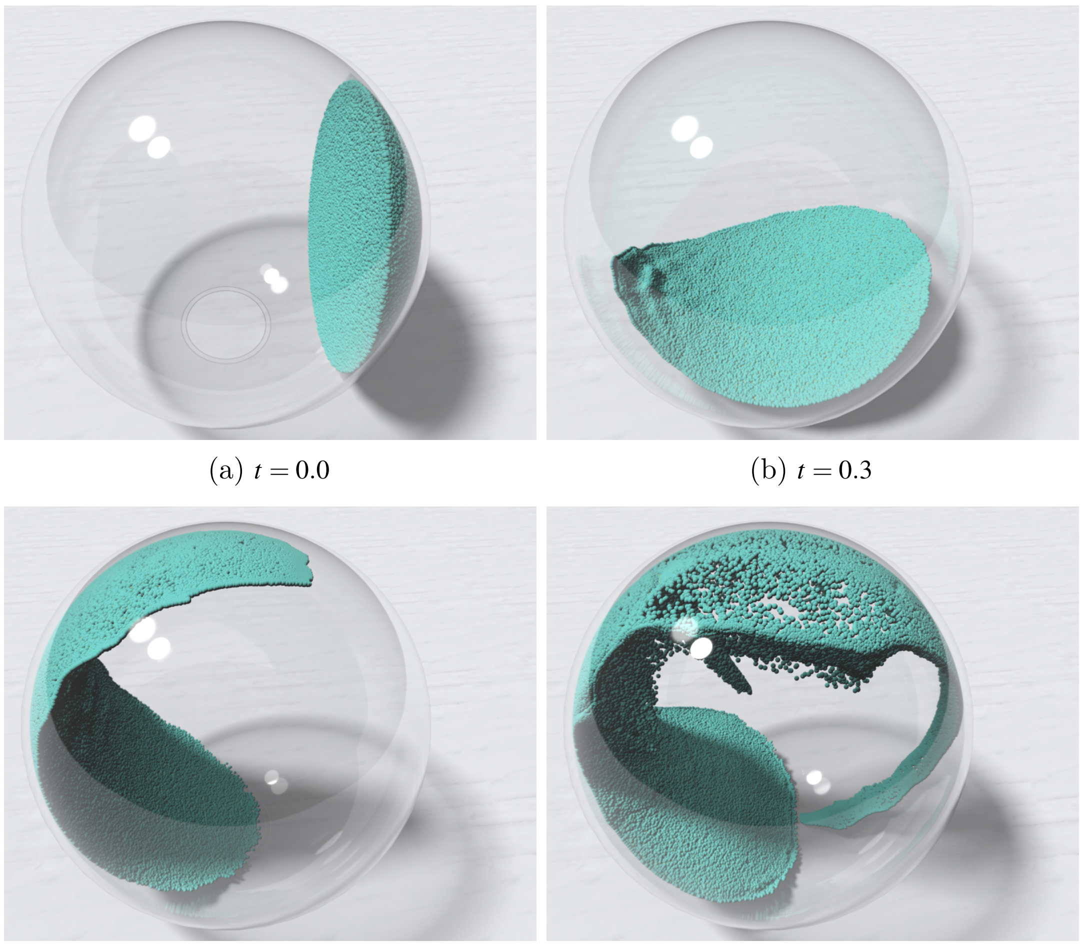
$$M = Rec \cdot (1 - FPR),$$

where $Rec = TP/(TP+FN)$ and $FPR = FP/(P+TN)$ are well-known metrics in data analysis (FAWCETT, 2006) called *Recall* and *False Positive Rate* (FPR), respectively. Recall measures the accuracy of a method in detecting boundary particles precisely among the actual set of boundary particles. The best result occurs when $Rec = 1$, meaning that all boundary particles were classified correctly, although of the possible presence of false positives. While FPR quantifies how many interior particles were misclassified as boundary. The best case occurs when $FPR = 0$, when no interior particles have been classified as boundary. Thus, the best classification occurs when the combined metric M reaches its maximum value of 1, i.e. when the Recall is maximum, and FPR is minimum simultaneously.

Table 6 and Table 7 show the number of particles ($|\mathscr{P}|$), the parameter r, and the average scores resulting from each technique in 2D and 3D, respectively. We apply the same parameters as suggested in the corresponding paper of each technique. Remembering, for the assessments of IA and GE, we choose the maximum depth of 6. Notice that our interval approaches reliably detect all boundary particles, providing certified results with $Rec = 1$. Moreover, the GE approach outperforms the other methods in all experiments.

We implemented the 2D version of our techniques in Python and the 3D version in C++. All boundary detection methods in C++ were parallelized using OpenMP, except the ground truth (HAQUE; DILTS, 2007) due to its complexity. All experiments have been performed on a computer equipped with processor Intel i7-8750H with six 2.2GHz cores and 16GB RAM. Table 8 shows the computational times regarding the number of processor cores for the experiments in 3D. The column *speedup* is the relation 1-core/6-

Figure 33 – APIC flow simulation of liquid sloshing starting in a section of a sphere of diameter 0.8: boundary particles detected by our GE method at different time instants.


(a) $t = 0.0$ (b) $t = 0.3$ (c) $t = 0.6$ (d) $t = 0.9$

Source: Sandim, Paiva and Figueiredo (2020).

core, and the last column is how much faster is a detection method than the exact method, i.e., the *rate* of CPU time of a detection method over the exact method using a single core. As can be seen, the GE method is almost twice as fast as the exact method in the worst case.

The scalability of the detection methods is measured on a mushroom jet simulation using SPH, as illustrated by Figure 35. In this simulation, more than 3 million particles are inserted in the system along the time. The computational time of GE method is located between the exact and the fastest methods.

4.5 Conclusion

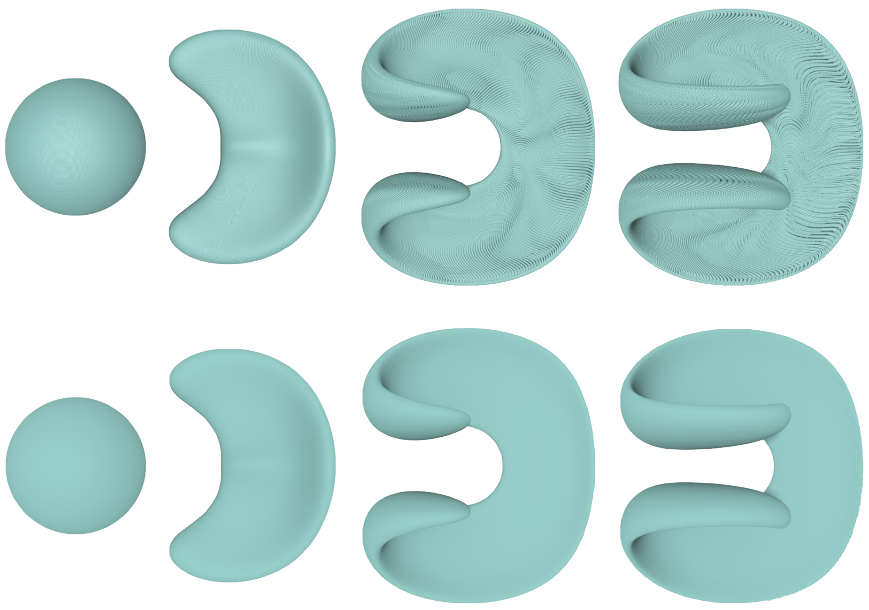
We have presented two novel methods for detecting boundary particles in both 2D and 3D domains. These methods are tailored to particle-based methods in free-surface flow

Table 6 – Quantitative analysis between different boundary detection methods in 2D (best results are shown in bold).

Experiment	$\|\mathscr{P}\|$	r	Methods	Rec	FPR	M
Figure 28	11.2k	0.009	Marrone et al.	0.9767	0.0245	0.9523
			Sandim et al.	0.9979	0.0184	0.9796
			Lin et al.	0.9762	0.0189	0.9577
			Our IA	**1.0000**	0.0036	0.9964
			Our GE	**1.0000**	**0.0016**	**0.9984**
Figure 29	837	0.022	Marrone et al.	0.9779	0.0701	0.9093
			Sandim et al.	**1.0000**	0.0776	0.9224
			Lin et al.	0.9982	0.0642	0.9341
			Our IA	**1.0000**	0.0569	0.9431
			Our GE	**1.0000**	**0.0472**	**0.9528**
Figure 30	[5k, 23.4k]	0.051	Marrone et al.	0.9735	0.0136	0.9602
			Sandim et al.	0.9540	0.0040	0.9502
			Lin et al.	0.9307	0.0106	0.9209
			Our IA	**1.0000**	0.0052	0.9948
			Our GE	**1.0000**	**0.0014**	**0.9986**

simulations. We combine a robust purely geometric sphere covering tests based on interval analysis with an adaptive spatial subdivision of the sphere associated with a given particle. Our approaches outperform the state-of-the-art boundary detection methods as attested by the set of experiments and comparisons carried out in the chapter. As an application, we show that the proposed methods can be applied to define a level-set function from

Figure 34 – The Enright test at different times $t = 0.0, 0.5, 1.0$ and 1.5 (from left to right): the boundary particles detected by our GE method (at top) and its level-set (at bottom).



Source: Sandim, Paiva and Figueiredo (2020).

Table 7 – Quantitative analysis between different boundary detection methods in 3D (best results are shown in bold).

Experiment	$\lvert\mathscr{P}\rvert$	r	**Methods**	*Rec*	*FPR*	*M*
Figure 31	275.4k	0.033	Marrone et al.	0.9127	0.0304	0.8850
			Sandim et al.	0.9992	0.0269	0.9723
			Our IA	**1.0000**	0.0094	0.9906
			Our GE	**1.0000**	**0.0065**	**0.9935**
Figure 32	1.1M	0.007	Marrone et al.	0.9708	0.0376	0.9343
			Sandim et al.	0.9998	0.0298	0.9700
			Our IA	**1.0000**	0.0122	0.9878
			Our GE	**1.0000**	**0.0070**	**0.9930**
Figure 33	550k	0.007	Marrone et al.	0.6928	0.0269	0.6742
			Sandim et al.	0.9938	0.0338	0.9602
			Our IA	**1.0000**	0.0169	0.9831
			Our GE	**1.0000**	**0.0089**	**0.9911**
Figure 34	1.9M	0.005	Marrone et al.	0.9708	0.0376	0.9343
			Sandim et al.	0.9998	0.0298	0.9700
			Our IA	**1.0000**	0.0122	0.9878
			Our GE	**1.0000**	**0.0070**	**0.9930**
Figure 35	[22k, 3.2M]	0.010	Marrone et al.	0.6727	0.1729	0.5564
			Sandim et al.	0.99908	0.2027	0.7965
			Our IA	**1.0000**	0.2219	0.7781
			Our GE	**1.0000**	**0.0787**	**0.9213**

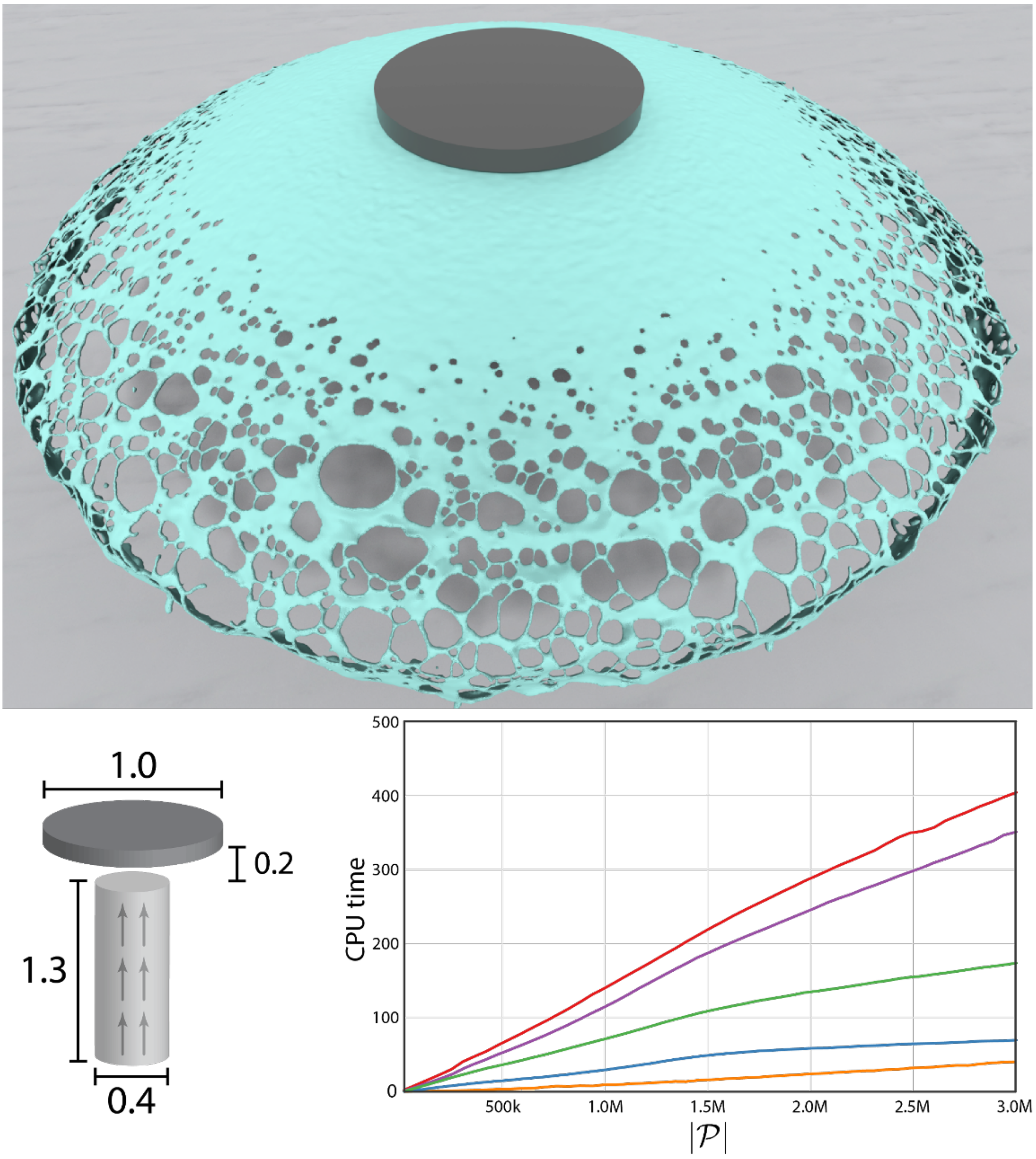
the boundary particles by using the strategy provided by Sandim et al. (SANDIM *et al.*, 2019).

As future work, we intend to port our geometric methods to GPU architecture, since the particle classification is performed locally and independently for each particle. Another direction is to apply our boundary detection to adapt tree-based grids dynamically around the liquid interface in fluid simulations (OLSHANSKII; TEREKHOV; VASSILEVSKI, 2013) and in particle remeshing applications (OBEIDAT; BORDAS, 2019) as well.

Table 8 – Average computational times (in seconds) per time-step.

Experiment	$\|\mathscr{P}\|$	**Methods**	1-core	6-core	speedup	rate
Figure 31	275.4k	Haque and Dilts	19.70	✗	✗	1.00
		Marrone et al.	4.77	0.97	4.91	4.13
		Sandim et al.	4.68	1.02	4.59	4.21
		Our IA	15.26	2.85	5.35	1.29
		Our GE	10.60	2.38	4.46	1.86
Figure 32	1.1M	Haque and Dilts	113.00	✗	✗	1.00
		Marrone et al.	31.09	6.52	4.77	3.63
		Sandim et al.	23.37	5.37	4.35	4.84
		Our IA	81.32	15.07	5.39	1.39
		Our GE	61.45	13.68	4.49	1.84
Figure 33	550k	Haque and Dilts	112.89	✗	✗	1.00
		Marrone et al.	15.51	3.60	4.30	7.28
		Sandim et al.	17.03	5.73	2.97	6.63
		Our IA	48.36	9.95	4.86	2.33
		Our GE	37.10	8.48	4.38	3.04
Figure 34	1.9M	Haque and Dilts	649.05	✗	✗	1.00
		Marrone et al.	149.16	35.53	4.20	4.35
		Sandim et al.	74.85	21.21	3.53	8.67
		Our IA	362.61	70.32	5.16	1.79
		Our GE	300.71	71.73	4.19	2.19

Figure 35 – SPH flow simulation of a mushroom jet with inlet velocity of $5 m/s$. On the top, the level-set from the boundary particles detected by our GE method at $t = 0.5$. On the bottom-left, the sketch of the problem geometry. On the bottom-right, the computational times (in seconds) of each detection method: Haque and Dilts (HAQUE; DILTS, 2007) (🟥), Marrone *et al.* (2010) (🟦), Sandim *et al.* (2016) (🟧), our IA (🟪), and our GE (🟩).



Source: Sandim, Paiva and Figueiredo (2020).

CHAPTER

5

CONCLUSION

In this thesis, we successfully explored the boundary detection problem, its definition, and possible solutions. We analyzed the methods present in the literature of both computer graphics and computational physics and proposed new techniques that contribute to both areas.

As shown in Chapter 2, we have successfully presented a mathematical definition of the problem. This formal definition allowed us to establish metrics to evaluate the effectiveness of a boundary solution method and develop methods that satisfy diverse requirements. Moreover, we presented a method that is simple to use, has a single parameter, works for most cases with its default configuration, and produces excellent results when compared to other popular and state-of-the-art methods. We also showed successful cases of the application of the method with the generation of free-surface meshes through surface-fitting algorithms.

The method presented in Chapter 3 builds upon the definition of the problem presented in the previous chapter while exploring an innovative way to tackle the problem. It also employed tools that have great potential to accelerate boundary detection and many other problems in computer graphics.

For applications where the user needs a robust boundary detection method, we presented methods based on interval analysis in Chapter 4. They outperform both state-of-the-art methods in boundary detection, including our previous solutions, while being straightforward and keeping a reasonable runtime.

We heavily tested all of our solutions using SPH simulations, but we have also shown cases have we successfully applied our solution to other methods, such as APIC. These results reinforce the flexibility of the proposed solutions and widen the range of possible use cases.

We also briefly discussed an application of one of the boundary detection methods to particle resampling. The method in Section 2.6 applies to different particle methods, improving the particle distribution significantly in critical areas, and it is fast and easy to implement. We showed how resampling improves the generation of free-surface meshes for rendering, without creating new particles unnecessarily.

5.1 Future work

All of the solutions proposed in this thesis can take advantage of parallel processing hardware and tools. The algorithms presented in Chapters 2 and 4 have low data dependency, low memory usage, rely on local computations, and simple operations. These aspects make them simple to adapt to a parallel processing model. The method proposed in Chapter 3 can use common machine learning frameworks that are ready to explore parallel processing hardware like TensorFlow(ABADI *et al.*, 2015) and ThunderSVM (WEN *et al.*, 2018). This type of change can significantly improve the performance of the methods.

Our machine learning method still suffers heavily from memory management issues. A more sophisticated sample selection scheme is crucial to allow its application to training-sets with a higher number of particles. Another alternative is the application of incremental training or methods suited to streams of data.

Explore the application of the proposed solutions integrated into a simulation pipeline is an exciting path to take. By inserting them into a pipeline, other steps of the simulation can take advantage of the information, improving results related to incompressibility, surface tension, interaction with other objects of a scene, and others. The particle resampling method presented in Section 2.6 has the potential to improve the simulation by keeping a more regular distribution in small feature areas during simulation, avoiding possible numerical errors due to the lack of neighbor particles to interpolate information.

Finally, it is nothing new to use computational geometry to solve other classes of problems. Some applications in pattern recognition, image processing, statistics, and data mining use convex-hull algorithms to extract information from multi-dimensional data. Given this, utilize our boundary detection methods with multi-dimensional data is an intriguing possibility.

BIBLIOGRAPHY

ABADI, M.; AGARWAL, A.; BARHAM, P.; BREVDO, E.; CHEN, Z.; CITRO, C.; CORRADO, G. S.; DAVIS, A.; DEAN, J.; DEVIN, M.; GHEMAWAT, S.; GOODFELLOW, I.; HARP, A.; IRVING, G.; ISARD, M.; JIA, Y.; JOZEFOWICZ, R.; KAISER, L.; KUDLUR, M.; LEVENBERG, J.; MANé, D.; MONGA, R.; MOORE, S.; MURRAY, D.; OLAH, C.; SCHUSTER, M.; SHLENS, J.; STEINER, B.; SUTSKEVER, I.; TALWAR, K.; TUCKER, P.; VANHOUCKE, V.; VASUDEVAN, V.; VIéGAS, F.; VINYALS, O.; WARDEN, P.; WATTENBERG, M.; WICKE, M.; YU, Y.; ZHENG, X. **TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems**. 2015. Software available from tensorflow.org. Available: <https://www.tensorflow.org/>. Citation on page 74.

AKINCI, G.; IHMSEN, M.; AKINCI, N.; TESCHNER, M. Parallel surface reconstruction for particle-based fluids. **Computer Graphics Forum**, v. 31, n. 6, p. 1797–1809, 2012. Citation on page 37.

AKINCI, N.; AKINCI, G.; TESCHNER, M. Versatile surface tension and adhesion for sph fluids. **ACM Trans. Graph.**, v. 32, n. 6, p. 182:1–182:8, 2013. Citations on pages 19 and 42.

AMENTA, N.; BERN, M. Surface reconstruction by voronoi filtering. In: **Proceedings of Symposium on Computational Geometry (SCG '98)**. [S.l.: s.n.], 1998. p. 39–48. Citation on page 41.

BARBER, C. B.; DOBKIN, D. P.; HUHDANPAA, H. The quickhull algorithm for convex hulls. **ACM Trans. Math. Softw.**, v. 22, n. 4, p. 469–483, 1996. Citation on page 30.

BARECASCO, A.; TERISSA, H.; NAA, C. Simple free-surface detection in two and three-dimensional SPH solver. **arXiv**, p. 1–10, 2013. Citation on page 52.

BELL, J. B.; COLELLA, P.; GLAZ, H. M. A second-order projection method for the incompressible Navier-Stokes equations. **J. Comput. Phys.**, v. 85, p. 257–283, 1989. ISSN 00219991. Available: <http://dx.doi.org/10.1016/0021-9991(89)90151-4>. Citation on page 48.

BERGER, M.; TAGLIASACCHI, A.; SEVERSKY, L. M.; ALLIEZ, P.; LEVINE, J. A.; SHARF, A.; SILVA, C. State of the art in surface reconstruction from point clouds. In: **Eurographics STAR (Proc. of EG'14)**. [S.l.: s.n.], 2014. Citation on page 35.

BHATTACHARYA, H.; GAO, Y.; BARGTEIL, A. W. A level-set method for skinning animated particle data. **IEEE Trans. Vis. Comput. Graph.**, v. 21, p. 315–327, 2015. Citations on pages 11, 37, and 38.

BOSER, B. E.; GUYON, I. M.; VAPNIK, V. N. A training algorithm for optimal margin classifiers. In: **COLT '92**. [s.n.], 1992. p. 144–152. ISBN 0-89791-497-X. Available: <http://doi.acm.org/10.1145/130385.130401>. Citation on page 47.

CARR, J. C.; BEATSON, R. K.; CHERRIE, J. B.; MITCHELL, T. J.; FRIGHT, W. R.; MCCALLUM, B. C.; EVANS, T. R. Reconstruction and representation of 3d objects with radial basis functions. In: **Proceedings of SIGGRAPH '01**. [S.l.: s.n.], 2001. p. 67–76. Citation on page 35.

CHANG, C.-C.; LIN, C.-J. LIBSVM: A library for support vector machines. **ACM TIST**, v. 2, p. 27:1–27:27, 2011. Code at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>. Citation on page 47.

CRESPO, A.; DOMíNGUEZ, J.; ROGERS, B.; GóMEZ-GESTEIRA, M.; LONGSHAW, S.; CANELAS, R.; VACONDIO, R.; BARREIRO, A.; GARCíA-FEAL, O. DualSPHysics: Open-source parallel CFD solver based on Smoothed Particle Hydrodynamics (SPH). **Comput. Phys. Commun.**, v. 187, n. 0, p. 204 – 216, 2015. ISSN 0010-4655. Available: <http://www.sciencedirect.com/science/article/pii/S0010465514003397>. Citation on page 48.

DILTS, G. A. Moving least-squares particle hydrodynamics II: conservation and boundaries. **Int. J. Numer. Meth. Eng.**, v. 48, n. 10, p. 1503–1524, 2000. Citations on pages 24, 32, 43, 44, 46, 48, 50, 51, 52, 53, 54, and 61.

ENRIGHT, D.; FEDKIW, R.; FERZIGER, J.; MITCHELL, I. A hybrid particle level set method for improved interface capturing. **J. Comput. Phys.**, v. 183, n. 1, p. 83–116, 2002. Citations on pages 33, 35, 37, 63, and 65.

FAN, R.-E.; CHEN, P.-H.; LIN, C.-J. Working set selection using second order information for training support vector machines. **The Journal of Machine Learning Research**, JMLR. org, v. 6, p. 1889–1918, 2005. Citation on page 47.

FASSHAUER, G. F. **Meshfree Approximation Methods with MATLAB**. [S.l.]: World Scientific, 2007. Citation on page 51.

FAWCETT, T. An introduction to ROC analysis. **Pattern Recogn. Lett.**, v. 27, n. 8, p. 861 – 874, 2006. Citation on page 66.

FLYER, N.; BARNETT, G. A.; WICKER, L. J. Enhancing finite differences with radial basis functions: experiments on the Navier-Stokes equations. **J. Comput. Phys.**, v. 316, p. 39–62, 2016. Citation on page 51.

GEUZAINE, C.; REMACLE, J.-F. A. Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities. **Int. J. Numer. Meth. Eng.**, v. 79, n. 11, p. 1309–1331, 2009. Citation on page 31.

GINGOLD, R. A.; MONAGHAN, J. J. Smoothed particle hydrodynamics: theory and application to non-spherical stars. **Mon. Not. R. Astron. Soc.**, v. 181, p. 375–389, 1977. Citations on pages 19 and 43.

GOMEZ-GESTEIRA, M.; ROGERS, B. D.; CRESPO, A. J. C.; DALRYMPLE, R. A.; NARAYANASWAMY, M.; DOMINGUEZ, J. M. SPHysics - development of a free-surface fluid solver - part 1: theory and formulations. **Comput. Geosci.**, v. 48, p. 289–299, 2012. Citations on pages 31 and 65.

GOSWAMI, P.; SCHLEGEL, P.; SOLENTHALER, B.; PAJAROLA, R. Interactive SPH simulation and rendering on the GPU. In: **Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA'10)**. [S.l.: s.n.], 2010. p. 55–64. Citation on page 19.

GUENNEBAUD, G.; JACOB, B. *et al.* **Eigen v3**. 2010. <http://eigen.tuxfamily.org>. Citation on page 39.

HAQUE, A.; DILTS, G. A. Three-dimensional boundary detection for particle methods. **J. Comput. Phys.**, v. 226, n. 2, p. 1710 – 1730, 2007. Citations on pages 24, 26, 31, 32, 52, 53, 54, 61, 66, and 71.

HE, X.; LIU, N.; WANG, G.; ZHANG, F.; LI, S.; SHAO, S.; WANG, H. Staggered meshless solid-fluid coupling. **ACM Trans. Graph.**, v. 31, n. 6, p. 149:1–149:12, 2012. Citations on pages 20, 24, 31, and 52.

HOSSEINI, S. M.; FENG, J. J. Pressure boundary conditions for computing incompressible flows with SPH. **J. Comput. Phys.**, v. 230, n. 19, p. 7473–7487, 2011. Citations on pages 20 and 52.

IHMSEN, M.; ORTHMANN, J.; SOLENTHALER, B.; KOLB, A.; TESCHNER, M. SPH Fluids in Computer Graphics. In: **Eurographics STAR (Proc. of EG'14)**. [S.l.: s.n.], 2014. Citation on page 24.

JIANG, C.; SCHROEDER, C.; TERAN, J. An angular momentum conserving affine-particle-in-cell method. **J. Comput. Phys.**, USA, v. 338, p. 137–164, 2017. Citation on page 65.

KATZ, S.; TAL, A. Improving the visual comprehension of point sets. In: **Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on**. [S.l.: s.n.], 2013. p. 121–128. Citation on page 41.

KATZ, S.; TAL, A.; BASRI, R. Direct visibility of point sets. **ACM Trans. Graph.**, v. 26, n. 3, 2007. Citations on pages 25, 26, and 53.

KAZHDAN, M.; HOPPE, H. Screened poisson surface reconstruction. **ACM Trans. Graph.**, v. 32, n. 3, p. 29:1–29:13, 2013. Citations on pages 36 and 38.

KIM, D. **Fluid Engine Development**. [S.l.]: CRC Press, 2016. Citation on page 65.

KOSHIZUKA, S.; SHIBATA, K.; KONDO, M.; MATSUNAGA, T. **Moving Particle Semi-Implicit Method**. [S.l.]: Academic Press, 2018. Citations on pages 19 and 51.

LEE, E.-S.; MOULINEC, C.; XU, R.; VIOLEAU, D.; LAURENCE, D.; STANSBY, P. Comparisons of weakly compressible and truly incompressible algorithms for the SPH mesh free particle method. **J. Comput. Phys.**, v. 227, n. 18, p. 8417–8436, 2008. Citation on page 52.

LEWINER, T.; LOPES, H.; VIEIRA, A. W.; TAVARES, G. Efficient implementation of marching cubes cases with topological guarantees. **Journal of Graphics Tools**, v. 8, n. 2, p. 1–15, 2003. Citation on page 37.

LIN, Y.; LIU, G.; WANG, G. A particle-based free surface detection method and its application to the surface tension effects simulation in smoothed particle hydrodynamics (SPH). **J. Comput. Phys.**, v. 383, p. 196 – 206, 2019. Citations on pages 20 and 52.

LIU, G.; LIU, M. **Smoothed Particle Hydrodynamics: A Meshfree Particle Method**. [S.l.]: World Scientific, 2003. Citations on pages 19, 51, and 56.

LIU, X.; LIN, P.; SHAO, S. An ISPH simulation of coupled structure interaction with free surface flows. **J. Fluids Struct.**, v. 48, p. 46–61, 2014. Citation on page 52.

LO, E. Y.; SHAO, S. Simulation of near-shore solitary wave mechanics by an incompressible SPH method. **Appl. Ocean Res.**, v. 24, n. 5, p. 275 – 286, 2002. Citation on page 52.

MACKLIN, M.; MÜLLER, M. Position based fluids. **ACM Trans. Graph.**, v. 32, n. 4, p. 104:1–104:12, 2013. Citations on pages 19 and 24.

MARRONE, S.; COLAGROSSI, A.; TOUZÉ, D. L.; GRAZIANI, G. Fast free-surface detection and level-set function definition in SPH solvers. **J. Comput. Phys.**, v. 229, n. 10, p. 3652–3663, 2010. Citations on pages 24, 31, 52, 63, 65, 66, and 71.

MERCIER, O.; BEAUCHEMIN, C.; THUEREY, N.; KIM, T.; NOWROUZEZAHRAI, D. Surface turbulence for particle-based liquid simulations. **ACM Trans. Graph.**, v. 34, n. 6, p. 202:1–202:10, 2015. Citation on page 42.

MOORE R. BAKER KEARFOTT, M. J. C. R. E. **Introduction To Interval Analysis**. [S.l.]: SIAM, 2009. Citation on page 54.

MÜLLER, M.; CHARYPAR, D.; GROSS, M. Particle-based fluid simulation for interactive applications. In: **Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation**. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2003. (SCA '03), p. 154–159. ISBN 1-58113-659-5. Available: <http://dl.acm.org/citation.cfm?id=846276.846298>. Citations on pages 24, 31, and 35.

NGUYEN, V. P.; RABCZUK, T.; BORDAS, S.; DUFLOT, M. Meshless methods: A review and computer implementation aspects. **Math. Comput. Simulat.**, v. 79, n. 3, p. 763 – 813, 2008. Citation on page 51.

OBEIDAT, A.; BORDAS, S. P. An implicit boundary approach for viscous compressible high reynolds flows using a hybrid remeshed particle hydrodynamics method. **J. Comput. Phys.**, v. 391, p. 347 – 364, 2019. Citation on page 69.

OLSHANSKII, M. A.; TEREKHOV, K. M.; VASSILEVSKI, Y. V. An octree-based solver for the incompressible Navier–Stokes equations with enhanced stability and low dissipation. **Computers & Fluids**, v. 84, p. 231 – 246, 2013. Citation on page 69.

OpenMP Architecture Review Board. **OpenMP Application Program Interface Version 3.1**. 2011. Citation on page 31.

ORTHMANN, J.; HOCHSTETTER, H.; BADER, J.; BAYRAKTAR, S.; KOLB, A. Consistent surface model for SPH-based fluid transport. In: **Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA'13)**. [S.l.: s.n.], 2013. p. 95–103. Citations on pages 19, 24, 31, and 43.

PARZEN, E. On the estimation of a probability density function and mode. **Annals of Mathematical Statistics**, v. 33, p. 1065–1076, 1962. Available: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.57.8666>. Citation on page 45.

PLATT, J. *et al.* Fast training of support vector machines using sequential minimal optimization. **Advances in kernel methods—support vector learning**, Cambridge, MA, v. 3, 1999. Citation on page 47.

POWERS, D. M. W. Evaluation: from Precision, Recall and F-measure to ROC, Informedness, Markedness and Correlation. **Journal of Machine Learning Technologies**, v. 2, n. 1, p. 37–63, 2011. Citation on page 32.

PREMZOE, S.; TASDIZEN, T.; BIGLER, J.; LEFOHN, A.; WHITAKER, R. T. Particle-based simulation of fluids. **Computer Graphics Forum**, v. 22, n. 3, p. 401–410, 2003. Citation on page 24.

RANDLES, P.; LIBERSKY, L. Smoothed particle hydrodynamics: Some recent improvements and applications. **Comput. Meth. Appl. Mech. Eng.**, v. 139, n. 1, p. 375– 408, 1996. Citation on page 52.

SANDIM, M.; CEDRIM, D.; NONATO, L. G.; PAGLIOSA, P.; PAIVA, A. Boundary detection in particle-based fluids. **Comput. Graph. Forum**, v. 35, n. 2, p. 215–224, 2016. Citations on pages 21, 23, 26, 27, 28, 29, 30, 32, 33, 35, 36, 37, 38, 43, 44, 46, 52, 66, and 71.

SANDIM, M.; OE, N.; CEDRIM, D.; PAGLIOSA, P.; PAIVA, A. Boundary particle resampling for surface reconstruction in liquid animation. **Computers & Graphics**, v. 84, p. 55 – 65, 2019. Citations on pages 21, 38, 39, 40, 41, 65, and 69.

SANDIM, M.; PAIVA, A. Supervised learning for boundary detection on particle systems. In: **Proceeding Series of the Brazilian Society of Computational and Applied Mathematics**. [S.l.]: SBMAC, 2020. v. 7, n. 1. Citations on pages 21, 45, 48, and 49.

SANDIM, M.; PAIVA, A.; FIGUEIREDO, L. H. de. Simple and reliable boundary detection for meshfree particle methods using interval analysis. **Journal of Computational Physics**, Elsevier BV, p. 109702, Jul. 2020. Available: <https://doi.org/10.1016/j.jcp.2020.109702>. Citations on pages 21, 54, 57, 59, 60, 61, 63, 64, 65, 66, 67, 68, and 71.

SILVA, R. Machado e; ESPERANÇA, C.; MARROQUIM, R.; OLIVEIRA, A. A. F. Image space rendering of point clouds using the HPR operator. **Computer Graphics Forum**, v. 33, n. 1, p. 178–189, 2014. Citation on page 42.

SOLENTHALER, B.; SCHLÄFLI, J.; PAJAROLA, R. A unified particle model for fluid–solid interactions. **Comput. Animat. Virtual Worlds**, v. 18, n. 1, p. 69–82, 2007. Citation on page 35.

VAPNIK, V.; CHERVONENKIS, A. Y. A class of algorithms for pattern recognition learning. **Avtomat. i Telemekh**, v. 25, n. 6, p. 937–945, 1964. Citation on page 47.

WANG, P.-P.; MENG, Z.-F.; ZHANG, A.-M.; MING, F.-R.; SUN, P.-N. Improved particle shifting technology and optimized free-surface detection method for free-surface flows in smoothed particle hydrodynamics. **Comput. Meth. Appl. Mech. Eng.**, v. 357, p. 112580, 2019. Citation on page 52.

WEN, Z.; SHI, J.; LI, Q.; HE, B.; CHEN, J. ThunderSVM: A fast SVM library on GPUs and CPUs. **Journal of Machine Learning Research**, v. 19, p. 797–801, 2018. Citation on page 74.

YU, J.; TURK, G. Reconstructing surfaces of particle-based fluids using anisotropic kernels. **ACM Trans. Graph.**, v. 32, n. 1, p. 5:1–5:12, 2013. Citation on page 35.

ZALESAK, S. T. Fully multidimensional flux-corrected transport algorithms for fluids. **J. Comput. Phys.**, v. 31, n. 3, p. 335–362, 1979. Citation on page 36.

ZHANG, Y.; SOLENTHALER, B.; PAJAROLA, R. Adaptive sampling and rendering of fluids on the GPU. In: **Proceedings of the Fifth Eurographics / IEEE VGTC Conference on Point-Based Graphics (PBG'08)**. [S.l.: s.n.], 2008. p. 137–146. Citation on page 24.

ZHU, Y.; BRIDSON, R. Animating sand as a fluid. **ACM Trans. Graph.**, v. 24, n. 3, p. 965–972, 2005. Citations on pages 11, 19, 24, 37, and 38.

APPENDIX

A

PROOF OF LEMMA 2.2.1

Lemma A.0.1. Let $\mathscr{P}$ be a r-sampling of a domain $\mathbf{\Omega}$ and $\mathbf{p}$ be a r-boundary point, that is, $\mathbf{p} \in \mathscr{B}_r$. There exists a viewpoint $\mathbf{V}$ such that $\mathbf{p}$ is in the set of visible points $\mathscr{H}_{\mathbf{V}}(\mathscr{P})$ resulting from HPR algorithm.

Proof. Since $\mathbf{p} \in \mathscr{B}_r$, there is at least one point $\mathbf{x} \in \partial B_r(\mathbf{p})$ that is not covered by any other ball $B_r(\mathbf{q})$ centered in $\mathbf{q} \in \mathscr{P} \setminus \{\mathbf{p}\}$. Therefore, a viewpoint $\mathbf{V}$ placed on $\mathbf{x}$ is closer to $\mathbf{p}$ than any other $\mathbf{q} \in \mathscr{P} \setminus \{\mathbf{p}\}$. By construction, the exponential flipping f is strictly monotonically decreasing along each ray from $\mathbf{V}$, then the inversion mapping will take $\mathbf{p}$ farther away from $\mathbf{V}$ than other point in $\mathscr{P} \setminus \{\mathbf{p}\}$. Thus, ensuring that $\hat{\mathbf{p}} = f(\mathbf{p})$ lies on the boundary of the convex hull of $\hat{\mathscr{P}} \cup \{\mathbf{V}\}$, i.e., $\mathbf{p} \in \mathscr{H}_{\mathbf{V}}(\mathscr{P})$. $\square$

APPENDIX

B

AD-HOC INTERVAL EVALUATION

Algorithm 5 shows pseudo code for computing the image of a 3D query box $Q = [x_{min}, x_{max}] \times [y_{min}, y_{max}] \times [z_{min}, z_{max}]$ by the function f_k defining the sphere S_k. As mentioned in the text, we avoid using outward rounding, because the geometric resolution is much lower than the numerical resolution of the floating point system.

Algorithm 5 – Ad-hoc interval evaluation of $f_k(Q)$

```
1: function FK($x_{min}, x_{max}, y_{min}, y_{max}, z_{min}, z_{max}$)
2:     global $x_k, y_k, z_k, r$
3:     $a_{min}, a_{max} \leftarrow$ SQR($x_{min}, x_{max}, x_k$)
4:     $b_{min}, b_{max} \leftarrow$ SQR($y_{min}, y_{max}, y_k$)
5:     $c_{min}, c_{max} \leftarrow$ SQR($z_{min}, z_{max}, z_k$)
6:     $f_{min} \leftarrow a_{min} + b_{min} + c_{min} - r^2$
7:     $f_{max} \leftarrow a_{max} + b_{max} + c_{max} - r^2$
8: end function
9: function SQR($t_{min}, t_{max}, t$)
10:     $t_{min} \leftarrow t_{min} - t$
11:     $t_{max} \leftarrow t_{max} - t$
12:     if $t_{min} \geq 0$ then
13:         $f_{min} \leftarrow t_{min}^2$
14:         $f_{max} \leftarrow t_{max}^2$
15:     else if $t_{max} \leq 0$ then
16:         $f_{min} \leftarrow t_{max}^2$
17:         $f_{max} \leftarrow t_{min}^2$
18:     else
19:         $f_{min} \leftarrow 0$
20:         $f_{max} \leftarrow$ MAX($t_{min}^2, t_{max}^2$)
21:     end if
22: end function
```

ICMC SÃO CARLOS USP