Journal of Computational Physics 451 (2022) 110838

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Partial optimal transport for a constant-volume Lagrangian mesh with free boundaries

Bruno Lévy

Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France

ARTICLE INFO

Article history:
Received 3 June 2021
Received in revised form 11 October 2021
Accepted 6 November 2021
Available online 15 November 2021

Keywords:
Fluids
Optimal transport
Free surface

ABSTRACT

This article introduces a representation of dynamic meshes, adapted to some numerical simulations that require controlling the volume of objects with free boundaries, such as incompressible fluid simulation, some astrophysical simulations at cosmological scale, and shape/topology optimization. The algorithm decomposes the simulated object into a set of convex cells called a Laguerre diagram, parameterized by the position of N points in 3D and N additional parameters that control the volumes of the cells. These parameters are found as the (unique) solution of a convex optimization problem – semi-discrete Monge-Ampère equation – stemming from optimal transport theory. In this article, this setting is extended to objects with free boundaries and arbitrary topology, evolving in a domain of arbitrary shape, by solving a partial optimal transport problem. The resulting Lagrangian scheme makes it possible to accurately control the volume of the object, while precisely computing the intersections with the domain boundary, the interactions, the collisions, and the changes of topology.

1. Introduction

Some numerical simulations require to control the volume of an object while allowing it to change shape and topology. For instance, in incompressible fluid simulation (see e.g. [1]), the volume of fluid is conserved, while the shape of the fluid can considerably vary throughout the simulation, and can change topology (split and merge). In some astrophysic simulations [2–4], the Universe considered at a cosmological scale can be modeled as a "fluid".[1] With the simplifying assumption that this fluid obeys the incompressible Euler equation, one can reconstruct the full dynamics of the Universe from the knowledge of the density fluid at current time [2–4]. To name another example, in shape and topology optimization (see e.g. [5]), one wants to find the shape of a given volume with maximum resistance (minimum compliance). Again, in this example, the considered shape can have an arbitrary topology, that can change during computation. There are several difficulties in the three applications mentioned above: (1) choosing a representation that can account for changes of topology, (2) controlling the volume, (3) tracking the interfaces and the changes of topology, and (4) tracking the interactions and the collisions between the simulated object and the boundary of the domain.

Clearly, to account for changes of topology, it is possible to represent the object as a density supported by an Eulerian grid (see for instance the *homogenization method* for shape optimization [6]) and enforce the constraint of volume conservation using Lagrange multipliers or similar techniques, then track isosurfaces in this grid. However, I think it is interesting to

E-mail address: Bruno.Levy@inria.fr.

[1] The "atoms" of this "fluid" correspond to galaxy clusters!

https://doi.org/10.1016/j.jcp.2021.110838
0021-9991/

experiment with an alternative Lagrangian representation that directly represents the simulated object and its boundary. The proposed alternative representation has the following properties:

- the new representation is a Lagrangian mesh that continuously depends on a set of N points in 3D space;
- each cell of the mesh has a prescribed volume for any position of the N points;
- the interactions between the cells and the boundaries of the domain are accurately computed, even when the domain's boundary features are smaller than the cell. The volume of the cells is accurately controlled, even in complicated geometric configurations;
- the surface is directly deduced from the N points (and the volume constraints), hence changes of topology are directly taken into account, without requiring injecting any additional information or using any additional heuristic.

The approach builds on recent advances in numerical optimal transport, that resulted in Lagrangian schemes for fluid simulation [7] or early Universe reconstruction [4]. In the works cited above, the object fills the entire simulation domain. In the rest of this article, after shortly introducing existing surface tracking / surface capturing methods (Section 2) and summarizing the semi-discrete optimal transport method (Section 3), I show how the mathematical setting can be extended to objects with free boundaries, by solving a *partial* optimal transport problem (Section 4). Then I detail the numerical algorithm that can solve this partial optimal transport problem (Section 5). Finally, I demonstrate some applications of the method to free-surface fluid simulation (Section 6).

2. Short review on numerical methods for free surfaces

Before entering the heart of the matter, I shall quickly overview the existing methods for free surfaces in the numerical simulation of fluids. This section is not a complete review on the topic, and is limited to a summary of the general ideas. For a more extensive review, one may refer to [8] for instance.

There are two main categories in numerical methods in fluid simulation, depending on how the physical quantities are represented:

- in *Lagrangian* methods, physical quantities are attached to particles that follow the fluid flow. On the one hand, it may give a natural representation of the free surface, on the other hand, this family of methods has the reputation to be difficult to implement. The method that I propose belongs to this category. It is similar to [9], that is also a Lagrangian method. The main difference resides in the analysis and the treatment of the geometric structure of the problem: the Partial Optimal Transport formulation presented here results in an algorithm that directly computes the free surface of the fluid without needing to introduce any "ghost particle";
- in *Eulerian* methods, physical quantities are expressed in terms of fields, that is, functions that depend on a location in space and time. In this setting, there are several ways of representing the fluid surface, detailed below;
- it is also possible to designed mixed methods, such as ALE (*Arbitrary Lagrangian Eulerian*) [10,11], that iteratively deforms a mesh of the initial fluid domain, with a special care to discretize time derivatives in a way that does not require to interpolate between two successive timesteps [12].

In Eulerian methods, the surface of the fluid can be represented in different ways. They can be classified into two main categories:

- the surface can be represented in a Lagrangian manner, by particles that carry physical quantities and/or geometric properties of the surface. Such methods are referred to as *interface tracking* [13–16]. In 2D, it is possible to directly study the changes of topology (bifurcation methods [17]), but it is hard to generalize to 3D. Changes of topology need to be taken into account by updating the connections between the vertices of a surface mesh [18]. In *front tracking* or *point set* methods, the connectivity of the mesh is reconstructed at each timestep, by using an indicator function [19,20]. Clearly, it is also possible to track particles in the whole fluid volume, as in the *Markers and Cells* (MAC) method [21]. Alternatively, in the *immersed boundary method*, the boundary can be explicitly represented as a curvilinear mesh [22], and the surface forces are represented by Dirac distributions supported by this mesh.
- the surface can be also represented in a Eulerian manner, using functions supported by a fixed grid. Then one talks about *interface capturing*. In the *level set method* [23,24], the surface is represented by a level set of a function ϕ. One typically uses functions that behave like the signed distance function. In the *volume of fluid method*, each cell of the mesh has an associated function α, that indicates the proportion of fluid in the cell [25]. There are several possibilities for the α functions (piecewise constant, linear, quadratic ...). One of the drawback of this method is that the function α is discontinuous. To mitigate this effect, it can be combined with the level set representation [26]. Another possibility, the *particle level set method*, consists in complementing the level set function ϕ with a set of massless particles that carry the sign of the distance function, used to disambiguate the topology [27,28]. Finally, there is also the *phase field method*, that represents a thin continuous transition around the interface, and that is well suited to model the energy of the system [29].

3. Volume control through optimal transport

In this section, I summarize the existing semi-discrete optimal transport framework with the objective of giving an intuition on the aspects that are important to control the volume of the cells in a Lagrangian mesh.[2] Then, in the next section, I explain how to extend this framework to simulated objects with free boundaries.

Let us suppose for now that the simulated object entirely fills a simulation domain Ω, that can be the 2D unit square or the 3D unit cube.[3] The goal is now to find a partition of Ω into a set of N cells V_i with the following properties:

- the set of cells $(V_i)_{i=1}^N$ depends on a set of parameters $(\mathbf{x}_i)_{i=1}^N$. These parameters can be interpreted as 2D (resp. 3D) points (hence a *Lagrangian* representation);
- the area (resp. volume) of each V_i's is controlled: $|V_i| = \nu_i$ for some prescribed $\nu_i's$ such that $\sum_{i=1}^N \nu_i = |\Omega| = 1$;
- the cells V_i *continuously* depend on the $\mathbf{x}_i$'s;

3.1. The Monge-Ampère equation

Such a Lagrangian mesh parameterization can be obtained through a specific (semi-discrete) version of the Monge-Ampère (MA) equation. In the general (continuous) setting, the MA equation may be thought of as seeking for an application $T : \Omega \to \Omega$ with controlled Jacobian, by solving for a potential $\psi : \Omega \to \mathbb{R}$:

$$\begin{cases} \det(\mathrm{Hess}(\psi)(\mathbf{x})) = \nu(\mathbf{x}) \quad \forall \mathbf{x} \in \Omega \\ \text{s.t. } \psi^{cc} = \psi \quad \text{where } \psi^c(\mathbf{x}) = \inf_{\mathbf{y}\in\Omega}[\|\mathbf{x}-\mathbf{y}\|^2 - \psi(\mathbf{y})] \end{cases} \tag{1}$$

where $\nu : \Omega \to \mathbb{R}^+$ is a square-integrable density.[4] In the constraint (second line), ψ^c corresponds to the Legendre-Fenchel transform of ψ, and the condition that applying it twice to ψ does not change ψ means that ψ is convex (because the graph of ψ^{cc} corresponds to the convex hull of the graph of ψ). From the solution of Eq. (1), one can deduce an *optimal transport map* $T : \mathbf{x} \mapsto \mathbf{x} - \frac{1}{2}\nabla\psi^c(\mathbf{x})$ that minimizes a "transport cost" while satisfying mass conservation [36]:

$$\begin{cases} \inf_T \left[\int_\Omega \|T(\mathbf{x}) - \mathbf{x}\|^2 d\mathbf{x}\right] \quad \text{s. t.} \\ \int_{T^{-1}(B)} d\mathbf{x} = \int_B \nu(\mathbf{x}) dx \quad \forall \text{ measurable set } B \subset \Omega. \end{cases} \tag{2}$$

With the viewpoint of optimal transport, ψ can be considered as the Lagrange multiplier of the volume conservation constraint (second line). The optimal transport map $T = \mathrm{Id} - \frac{1}{2}\nabla\psi^c$ is deduced from the gradient of the Legendre transform of the solution ψ of the Monge-Ampère equation (1). The left-hand side of the Monge-Ampère equation corresponds to the Jacobian of T. Hence, the Monge-Ampère equation is a mean of controlling the Jacobian of an application. The Kantorovich dual of (a relaxed version of) the optimization problem in Eq. (2) writes:

$$\begin{cases} \sup_\psi \left[K(\psi) = \int_\Omega \psi^c(\mathbf{x}) d\mathbf{x} + \int_\Omega \psi(\mathbf{x})\nu(\mathbf{x}) d\mathbf{x}\right] \\ \text{s. t. } \psi^{cc} = \psi \end{cases} \tag{3}$$

The Kantorovich dual is smooth and convex. These properties are interesting because they can be used to establish the existence and uniqueness of the optimal transport map and the associated Lagrange multiplier ψ. They can be also exploited to design a numerical solution mechanism. In the next subsection, I present the so-called *semi-discrete* setting, where the target density ν is replaced with a discrete probability measure, supported by a pointset $(\mathbf{x}_i)_{i=1}^N$.

3.2. Semi-discrete optimal transport

The equivalence between the Kantorovich dual problem (Eq. (3)) and the optimal transport problem (Eq. (2)), established in Brenier's polar factorization theorem [36] characterizes mathematical objects (probability measures) that can be less regular than the densities involved in the Monge-Ampère equation (Eq. (1)). For instance, the density ν can be replaced with a weighted sum of Dirac masses $\nu = \sum_{i=1}^N \nu_i \delta_{\mathbf{x}_i}$. A subset B of Ω is measured as follows by a so-defined ν:

$$\int_B \nu(\mathbf{x}) d\mathbf{x} = \sum_{j|\mathbf{x}_j \in B} \nu_j.$$

[2] The reader is referred to [30–35] for an extensive and general introduction on optimal transport.

[3] I will consider later in this article the case where Ω is an arbitrary simplicial set.

[4] In its general form, the Monge-Ampère equation also has a density μ on the left-hand side, but in the context of this article, I consider a uniform density $\mu = 1$.

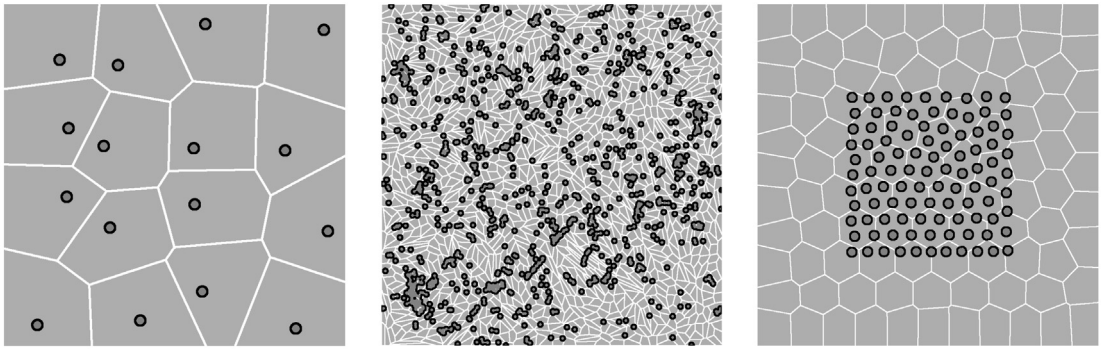

Fig. 1. Left: Example of Laguerre diagram. Center and Right: unlike Voronoi cells, Laguerre cells do not necessarily contain the points they are associated with, depending on both the distribution of the points (center) and the values of the ψ_i's (right).

In this setting, the potential function ψ is parameterized by a vector $(\psi_i)_{i=1}^N$. The functional K of the dual problem (Eq. (3)) becomes a function $K : \mathbb{R}^N \to \mathbb{R}$ that depends on the vector $(\psi_i)_{i=1}^N$:

$$\left\{\begin{array}{lll} K(\psi) &= \int_\Omega \psi^c(\mathbf{x})d\mathbf{x} &+ \int_\Omega \psi(\mathbf{x})\nu(\mathbf{x})d\mathbf{x} \\ &= \int_\Omega \psi^c(\mathbf{x})d\mathbf{x} &+ \sum_i \psi_i \nu_i \\ &= \int_\Omega \inf_i \left[\|\mathbf{x}-\mathbf{x}_i\|^2 - \psi_i\right] d\mathbf{x} &+ \sum_i \psi_i \nu_i. \end{array}\right. \tag{4}$$

The second line is obtained by taking into account the discrete definition of ν, and the third one by replacing ψ^c by its definition (in Equation (1)). The integral in the first term of $K(.)$ can be rearranged:

$$K(\psi) = \sum_i \int_{Lag_i\psi} (\|\mathbf{x}-\mathbf{x}_i\|^2 - \psi_i)d\mathbf{x} + \sum_i \psi_i \nu_i \tag{5}$$

where the partition of Ω into the sets Lag_i^ψ, called a *Laguerre diagram*, is defined as follows:

Definition 1. Given the unit square Ω (resp. unit cube), a set $\mathbf{X} = (\mathbf{x}_1, \dots \mathbf{x}_N)$ of points in Ω, a vector $\psi = (\psi_1, \dots \psi_n) \in \mathbb{R}^N$, the Laguerre diagram is the partition of Ω into the N regions Lag_i^ψ defined by:

$$Lag_i^\psi = \left\{\mathbf{x} \in \Omega \quad | \quad \|\mathbf{x}-\mathbf{x}_i\|^2 - \psi_i \;\leq\; \|\mathbf{x}-\mathbf{x}_j\|^2 - \psi_j \quad \forall j \neq i \right\}$$

Each region Lag_i^ψ is referred to as a *Laguerre cell*. Laguerre diagram can be considered as a generalization of Voronoi diagram (they are equivalent in the specific case where all the ψ_i's have the same value). Laguerre diagrams have been extensively studied and characterized in computational geometry [37] (in this context, if the cost corresponds to the squared distance, like in our case, Laguerre diagrams are called *power diagrams*). Two examples of Laguerre diagrams are shown in Fig. 1. It is worth mentioning that depending on the ψ_i's, a Laguerre cell *does not necessarily contain the point* it is associated with (unlike Voronoi cells). For instance, it is easy to check that one can translate the entire diagram by an arbitrary vector $\mathbf{u}$, without changing the $\mathbf{x}_i$'s, by setting $\psi_i = \sqrt{\max_j(\mathbf{u}\cdot\mathbf{x}_j) - \mathbf{u}\cdot\mathbf{x}_i}$, hence the cells can be arbitrarily far away from the $\mathbf{x}_i$'s.

The Laguerre diagram can also be considered as the minimization diagram of the familly of functions $f_i(\mathbf{x}) = \|\mathbf{x}-\mathbf{x}_i\|^2 - \psi_i$. The graphs of these functions (paraboloids) for a 2D diagram are pictured in Fig. 2-Left. The coefficient ψ_i 'shifts' the associated paraboloid along the Z axis. Seen from above (Fig. 2-Right), the function graphs appear as convex polygonal cells. The common boundary between two cells corresponds to the projection of a parabola included in a plane orthogonal to the picture, hence a straight segment.

Still considering Fig. 2-Left, each coefficient ψ_i corresponds to the amount of translation of the associated paraboloid along the Z axis. As one can imagine, lowering a paraboloid (that is, *increasing* ψ_i, because of the "-" sign) increases the size of the associated Laguerre cell. Conversely, raising it decreases the size of the cell.[5] One can also see that the Laguerre diagram does not change when adding a constant to all ψ_i's (the entire diagram is shifted along the Z axis and the image does not change when viewed from above).

Now the question is: given a vector of prescribed volumes $\nu_i \geq 0$, such that $\sum_i \nu_i = |\Omega| = 1$, is it possible to find the values of ψ_i such that $|Lag_i^\psi| = \nu_i$ for all i? In other words, is it possible to tune the heights of the paraboloids in such a way that the areas of the cells seen from above match the prescribed areas? The (positive) answer is given by the following

[5] A cell can even completely disappear if the corresponding paraboloid is shifted above all the other graphs.

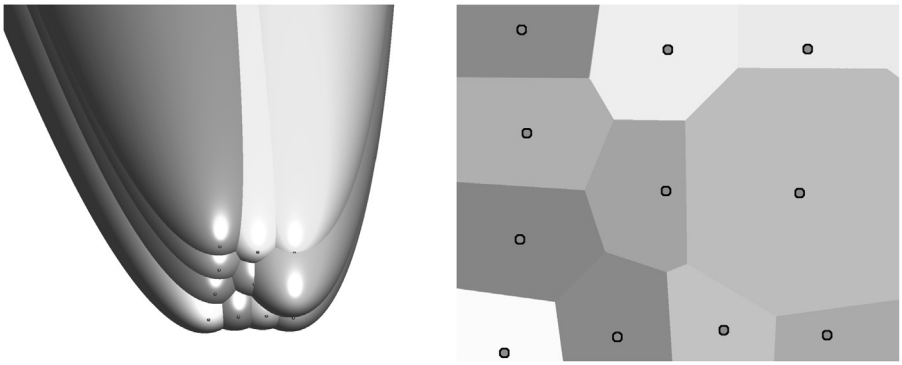

Fig. 2. Laguerre diagrams considered from an alternative viewpoint. Left: the Laguerre diagram is the minimization diagram of the family of functions $f_i(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}_i\|^2 - \psi_i$. Right: looking at these graphs from above, one sees a set of convex polygonal cells.

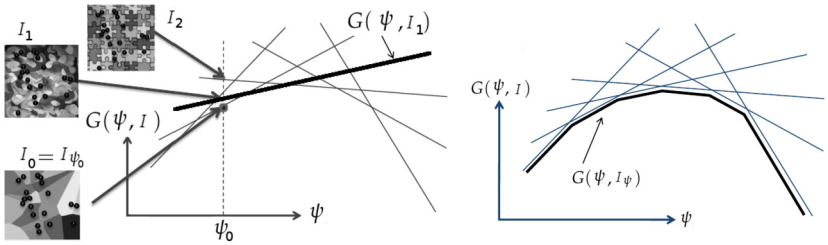

Fig. 3. Convexity of K: the graph of K is the lower envelope of a family of affine functions $G(\psi, I)$, parameterized by the Lagrange multipliers ψ and an index map I, that maps each point of Ω to an index in $[1 \dots N]$ (symbolized by colors on the left).

theorem [38] (it is also a direct consequence of Brenier's more general polar factorization theorem [36] considered in the specific semi-discrete setting). The main argument of the proof in [38] is summarized below:

Theorem 1. *Given a set of n points* $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_n)$ *in* Ω*, a set of positive prescribed volumes* $(v_1, v_2, \dots v_n)$ *such that* $\sum_i v_i = 1$*, there exists a unique (up to a translation) set of scalars* $(\psi_1, \psi_2, \dots \psi_n)$ *such that each Laguerre cell* Lag_i^{ψ} *has the prescribed volume* V_i*.*

Proof. (summarized, see [38,31] for a complete proof) Consider the Kantorovich dual function $K(\psi) : \mathbb{R}^N \to \mathbb{R}$ defined by:

$$\begin{aligned} K(\psi) &= \sum_i \int_{Lag_i^{\psi}} (\|\mathbf{x} - \mathbf{x}_i\|^2 - \psi_i) d\mathbf{x} + \sum_i \psi_i v_i \\ &= \int_{\Omega} \inf_i \big[\|\mathbf{x} - \mathbf{x}_i\|^2 - \psi_i \big] d\mathbf{x} + \sum_i \psi_i v_i \end{aligned}.$$

The function K has the following properties:

1. the function K is concave;
2. the function K is differentiable up to the second order;
3. the components of the gradients of K are given by:
   $\partial K / \partial \psi_i = v_i - |Lag_i^{\psi}|$.

I explain below how to obtain the concavity of K (the two other properties can be obtained by direct calculation). The second-order differentiability of K with respect to ψ is studied in [39,40], and its differentiability with respect to $\mathbf{x}$ in [41], as well as the expression of the second-order derivatives.

To show the concavity of K, let us introduce the following functional G, parameterized by ψ and by an *index map* $I : \Omega \to [1, \dots N]$ that associates one of the points $\mathbf{x}_{I(\mathbf{x})}$ to each point $\mathbf{x} \in \Omega$.

$$G(\psi, I) = \int_{\mathbb{R}^3} (\|\mathbf{x} - \mathbf{x}_{I(\mathbf{x})}\|^2 - \psi_{I(\mathbf{x})}) d\mathbf{x}.$$

Clearly, it is easy to verify that for a fixed index map I, G *affinely* depends on ψ. Among all the possible index maps I, let us consider now I_ψ that maps each point $\mathbf{x}$ to the index of the Laguerre cell it belongs to:

$$I_\psi(\mathbf{x}) = i \quad \forall \mathbf{x} \in Lag_i^\psi$$

We also have (from the definition of Lag_i^ψ):

$$I_\psi(\mathbf{x}) = \arg\min_i [\|\mathbf{x} - \mathbf{x}_i\|^2 - \psi_i]$$

which implies that for a given ψ_0, among all possible index maps I, I_{ψ_0} minimizes $G(\psi_0, I)$. Thus, the graph of $G(\psi, I_\psi)$ is the lower envelope of a family of affine functions, hence $\psi \mapsto G(\psi, I_\psi)$ is a concave function (see Fig. 3). Since $G(\psi, I_\psi)$ corresponds to the first term of K, and since the second term of K is linear in ψ, K is also concave. K has a unique maximizer ψ^*, where its gradient vanishes, and the expression of the derivatives indicates that the volumes of the Laguerre cells match the prescribed volumes. □

It is easy to verify that the optimal transport map $T_{\psi^*} = \mathrm{Id} - \frac{1}{2}\nabla\psi^c$ maps a point $\mathbf{x}$ of Ω to the point $\mathbf{x}_{I^{\psi^*}}(\mathbf{x})$:

$$\begin{aligned} T_{\psi^*}(\mathbf{x}) &= \mathbf{x} - \tfrac{1}{2}\nabla\left(\inf_i[\|\mathbf{x} - \mathbf{x}_i\|^2 - \psi_i]\right) \\ &= \mathbf{x} - \tfrac{1}{2}\nabla\left(\|\mathbf{x} - \mathbf{x}_i\|^2 - \psi_i\right) \qquad \text{where } i = I^{\psi^*}(\mathbf{x}) \\ &= \mathbf{x} - \tfrac{1}{2}(2\mathbf{x} - 2\mathbf{x}_i) \\ &= \mathbf{x}_i \end{aligned}$$

In other words, for all $\mathbf{x} \in Lag_i^{\psi^*}$, $T(\mathbf{x}) = \mathbf{x}_i$. Thus, the Laguerre cell Lag_i^ψ corresponds to the set of points $\mathbf{x}$ of Ω mapped to $\mathbf{x}_i$ through the optimal transport map. The theorem implies that the following three statements are equivalent:

1. The application T_{ψ^*} is the solution to the optimal transport problem (Eq (2));
2. the ψ_i^*'s are such that $|Lag_i^{\psi^*}| = \nu_i$ for all $1 \le i \le n$;
3. the vector $(\psi^*)_{i=1}^N$ is the unique (up to a translation) maximizer of the Kantorovich dual K.

In the context of this article, the initial motivation (controlling the volumes of a Lagrangian mesh) is satisfied by the second statement. The optimal transport map T_{ψ^*} in the first one can be considered as a "byproduct" that will not be used directly. In the (2D) context depicted in Fig. 1, the theorem means that by shifting correctly the paraboloids along the Z axis, one can make the areas of the Voronoi cells match prescribed areas. In addition, the third statement and the second-order differentiability of K [39,41] can be exploited to design a Newton algorithm [40]. It means that from any set of points $(\mathbf{x}_i)_{i=1}^N$, it is possible to compute a partition of Ω into a set of cells $V_i = Lag_i^\psi$ of prescribed volumes, that is $|V_i| = \nu_i$ for a set of positive ν_i's such that $\sum_i \nu_i = |\Omega_i|$. About the convexity constraint $\psi^{cc} = \psi$ in Eq. (3), in the semi-discrete setting, it is equivalent to the absence of empty Laguerre cells in the diagram ($Lag_i^\psi \neq \emptyset \quad \forall 1 \le i \le N$), as can be shown by direct computation (see [31] for details).

Before detailing the numerical solution mechanism that computes the ψ_i's, I shall explain how the geometric setting can be extended to objects with free boundaries.

4. Free boundaries through partial optimal transport

Let us now consider a different setting, where the simulated object does not fill the volume Ω entirely. That is, the sum of the prescribed volumes ν_i is smaller than the volume of Ω. This problem is referred to as a *partial optimal transport* problem. I show how partial optimal transport can be considered as a particular instance of optimal transport. To do so, I still consider the optimal transport problem (Eq. (2)) and its Kantorovich dual (Eq. (3)), with the difference that this time transport is done towards a set of "objects" (pointsets) $\mathcal{O}_i$:

$$\begin{aligned} &\inf_I \left[\int_\Omega d^2(\mathbf{x}, \mathcal{O}_{I(\mathbf{x})}) d\mathbf{x}\right] \\ &\text{subject to } |I^{-1}(\mathcal{O}_i)| = \nu_i \quad \forall i \end{aligned} \tag{6}$$

where:

- the index map $I : \Omega \to [1..N]$ assigns an object $\mathcal{O}_i$ to each point $\mathbf{x}$ of Ω;
- each $\mathcal{O}_i$ is a set of points of Ω (that can contain either a single point, or an integer number of points, or be a continuous subset of Ω);
- each $\mathcal{O}_i$ is supposed to receive a prescribed quantity of matter ν_i (constraint);
- the distance between a point $\mathbf{x}$ and an object $\mathcal{O}$ is defined by $d(\mathbf{x}, \mathcal{O}) = \inf_{\mathbf{y} \in \mathcal{O}} [\|\mathbf{x} - \mathbf{y}\|]$;
- $I^{-1}(\mathcal{O}_i) = \{\mathbf{x} \mid I(\mathbf{x}) = i\}$;
- Given an index map I, the corresponding map $T_I : \Omega \to \Omega$ is defined by $T_I(\mathbf{x}) = \arg\inf_{\mathbf{y} \in \mathcal{O}_{I(\mathbf{x})}} \|\mathbf{x} - \mathbf{y}\|^2$.

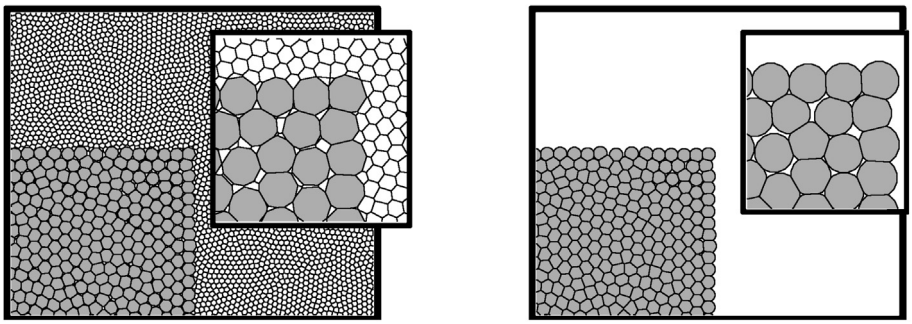

Fig. 4. Left: approximation of partial optimal transport using a discretization of the background ("ghost cells"). Right: exact solution to partial optimal transport, where the background is considered as a continuum. Each cell V_i^ψ is the intersection between the Laguerre cell Lag_i^Ψ and the ball centered on $\mathbf{x}_i$ of radius $\sqrt{\psi_i}$.

Consider now the dual problem:

$$\begin{array}{l}\sup_\psi \left[K(\psi) = \int\limits_\Omega \inf\limits_i \left[d^2(\mathbf{x}, \mathcal{O}_i) - \psi_i \right] d\mathbf{x} + \sum\limits_i \psi_i \nu_i \right] . \\ \text{subject to } \psi^{cc} = \psi\end{array}$$

It keeps the structure of our initial Kantorovich dual (Eq. (3), (4)), and the argument for the convexity, smoothness, existence and uniqueness of the solution still hold, but now I have a more general setting that I can use to account for free boundaries: I consider now a set of $N+1$ objects $\mathcal{O}_{i=0}^N$:

- The first object $\mathcal{O}_0$ is a set of M points $\{\mathbf{y}_1, \dots \mathbf{y}_M\}$ uniformly distributed in Ω;
- each $\mathcal{O}_i$ for $1 \leq i \leq N$ is the singleton $\{\mathbf{x}_i\}$;
- the object $\mathcal{O}_0$ is associated with $\nu_0 = |\Omega| - \sum_{i=1}^N \nu_i$, that is all the volume of Ω not affected to the points $\{\mathbf{x}_i\}$.

This configuration is depicted in Fig. 4-Left: the cells of the fluid are displayed in grey. Each of them is a convex polygon. The background points $\mathbf{y}_{i=1}^M$ fill the entire domain Ω, including the part occupied by the fluid. All the $\mathbf{y}_i$ points share the same Lagrange multiplier ψ_0. This may be interpreted as the fact that it takes no cost for background cells to exchange matter (air). For a part occupied by the fluid, each cell Lag_i^ψ that correspond to the fluid has a Lagrange multiplier ψ_i larger than ψ_0, then the fluid cell "shadows" the background cells. On the boundary of the fluid, the presence of the background cells limits the size of the fluid cells. The external boundary of the fluid is formed by straight segment shared by a fluid cell and a background cell (closeup in Fig. 4-Left).

A similar technique was used in [9], where it is proposed to insert points $\mathbf{y}_i$ in the vicinity of the boundary of the fluid (in the cited article, the corresponding cells are called "ghost cells"). The main advantage it that it allows directly reusing a discrete optimal transport implementation to solve transport problems with free boundaries. However, the "ghost cells" technique has two drawbacks:

- first, it is difficult to know *in advance* where to insert the ghost cells to make sure the boundary of the fluid is accurately represented, that is, there is no easy way of predetermining which ghost cells influence the fluid;
- second, the ghost cells have a significant computational cost: computing a Laguerre diagram costs $O(N\sqrt[d]{N})$ (where $d \in \{2,3\}$ is the dimension), and the Newton algorithm converges in $O(N\log(N))$ iterations (empirical results in [4]). The "ghost cells" technique computes a diagram with $N+M$ vertices (instead of N), where M needs to be sufficiently large to accurately capture the free boundary.

To overcome these limitations, I propose an alternative technique: let us now imagine that the number of background points tends to infinity. More simply put, the background object $\mathcal{O}_0$ becomes the entire Ω domain (and it is still associated with $\nu_0 = |\Omega| - \sum_{i=1}^N \nu_i$). Let us also remember that the vector $\psi_{i=0}^N$ is independent on a translation, thus, w.l.o.g. it is possible to choose $\psi_0 = 0$. Let us now consider a cell V_i^ψ associated with a fluid particle:

$$\begin{aligned} V_i^\psi &= \{\mathbf{x} \mid d^2(\mathbf{x}, \mathcal{O}_i) - \psi_i < d^2(\mathbf{x}, \mathcal{O}_j) - \psi_j \quad \forall j \neq i \} \\ &= \{\mathbf{x} \mid \|\mathbf{x} - \mathbf{x}_i\|^2 - \psi_i < \|\mathbf{x} - \mathbf{x}_j\|^2 - \psi_j) \quad \forall 1 \leq j \neq i \leq N \\ &\quad \text{and} \quad \|\mathbf{x} - \mathbf{x}_i\|^2 - \psi_i < d^2(\mathbf{x}, \mathcal{O}_0) - \psi_0 \} \\ &= \{\mathbf{x} \mid \|\mathbf{x} - \mathbf{x}_i\|^2 - \psi_i < \|\mathbf{x} - \mathbf{x}_j\|^2 - \psi_j) \quad \forall 1 \leq j \neq i \leq N \\ &\quad \text{and} \quad \|\mathbf{x} - \mathbf{x}_i\|^2 < \psi_i \}. \end{aligned}$$

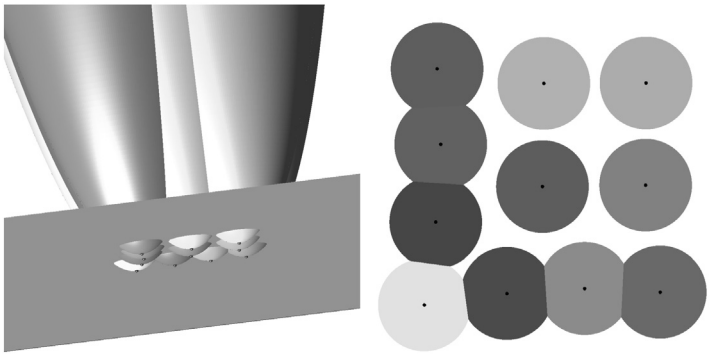

Fig. 5. Left: The minimization diagram of the familly of functions $f_i(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}_i\|^2 - \psi_i$ augmented with the graph of the function $f(\mathbf{x}) = 0$ (horizontal plane). Right: seen from below, the obtained cells V_i^{ψ} are the intersections between the Laguerre cells Lag_i^{ψ} and the disks of radii $\sqrt{\psi_i}$ centered on the $\mathbf{x}_i$'s (or the empty set if ψ_i is negative).

The last line is obtained by remembering that $\psi_0 = 0$, and by noticing that $d^2(\mathbf{x}, \mathcal{O}_0) = 0$ for all $\mathbf{x}$ in Ω since $\mathcal{O}_0 = \Omega$. In other words, V_i^{ψ} is the intersection between the Laguerre cell Lag_i^{ψ} and a disk of radius $\sqrt{\psi_i}$ centered on $\mathbf{x}_i$, as shown in Fig. 4-Right.[6] From this observation, not only "ghost cells" are no longer needed, but also we can much more accurately compute the boundary of the fluid, by computing the intersection between the Laguerre cells and a set of disks. Put differently, what is computed is the limit case where the number of ghost cells M tends to infinity. Not only the result will be more precise, but the overall computational cost will be significantly reduced, since the number of vertices in the Laguerre diagram is not increased.

As shown in Fig. 5, to gain more intuition about this setting, it is also possible to take the "minimization diagram" point of view: the distance to $\mathcal{O}_0 = \Omega$ is equal to zero on Ω, and $\psi_0 = 0$, thus the graph of the associated function f_0 is the horizontal plane $Z = 0$. The other functions f_i are shifted paraboloids just like before. Now, looking at the diagram from above, one will see the tip of the paraboloids intersected by the horizontal planes (disks). The paraboloids can also intersect each other, forming polygonal cells with straight edges (that correspond to projected parabolas), just like in the previous configuration.

5. Numerical solution mechanism

I shall now explain how to design a numerical solution mechanism. The associated algorithm takes the following inputs and produces the following outputs:

Input: – The domain Ω (can be $[0,1]^d$ or a simplicial mesh)
– a set of N points $\mathbf{x}_i \in \Omega$
– prescribed volumes $(\nu_i)_{i=1}^N$ such that $\sum_i \nu_i \leq |\Omega|$

Output: –the (unique) vector $\psi^* \in \mathbb{R}^N$ that maximizes $K(.)$
–the cells $(V_i^{\psi^*})_{i=1}^N$ such that $|V_i^{\psi^*}| = \nu_i$

The core of the algorithm solves the following optimization problem:

$$\begin{aligned} &\sup_{\psi} \left[K(\psi) = \sum_{i=1}^{N} \int_{V_i^{\psi}} (\|\mathbf{x} - \mathbf{x}_i\|^2 - \psi_i) d\mathbf{x} + \sum_{i=1}^{N} \nu_i \psi_i \right] \\ &\text{where } V_i^{\psi} = Lag_i^{\psi} \cap \{\mathbf{x} \mid \|\mathbf{x} - \mathbf{x}_i\|^2 \leq \psi_i\} \\ &\text{subject to } V_i^{\psi} \neq \emptyset \quad \forall 1 \leq i \leq N \end{aligned} \tag{7}$$

The KMT algorithm due to Kitagawa, Merigot and Thibert [40] is a Newton algorithm that produces a series of iterates $\psi^{(k)}$ that provably converges to the solution of Eq. (7). Each iterate satisfies the convexity constraint $\psi^{cc} = \psi$ (that is, $V_i^{\psi} \neq \emptyset \quad \forall k, \forall 1 \leq i \leq N$, see also [31]). Adapting the KMT algorithm to our specific context has numerical and geometrical aspects detailed below.

[6] or the empty set if ψ_i is negative. Note that a negative ψ_i cannot happen in our context, since it would contradict the convexity of ψ ($\psi^{cc} = \psi \Leftrightarrow V_i^{\psi} \neq \emptyset \ \forall 1 \leq i \leq N$), guaranteed by all the iterations of the KMT Newton algorithm [40] (more on this below).

5.1. Numerical aspects

The KMT algorithm follows the classical structure of a Newton algorithm:

(1) : $\psi \leftarrow [0 \dots 0]$
(2) : Loop
(3) : Compute the cells $(V_i)_{i=1}^N$
(4) : Compute the gradient $\nabla K(\psi)$
(5) : If $\|\nabla K(\psi)\|_\infty < \epsilon_K$ then Exit loop
(6) : Compute the Hessian matrix $\nabla^2 K(\psi)$
(7) : Solve for $\mathbf{p} \in \mathbb{R}^n$ in $\nabla^2 K(\psi)\mathbf{p} = -\nabla K(\psi)$
(8) : Find the descent parameter α
(9) : $\psi \leftarrow \psi + \alpha \mathbf{p}$
(10) : End loop

The algorithm above needs to evaluate the gradient and Hessian matrix of $K(.)$ at each main loop iteration. The coefficients of the gradient and Hessian matrix can be deduced from the cells $(V_i^\psi)_{i=1}^N$ that are computed at step (3). The algorithm that computes the cells is detailed later in the next subsection on the geometric aspects. The components $\partial K/\partial \psi_i$ of the gradient $\nabla K(.)$ are given by the following expression [40,31]:

$$\frac{\partial K}{\partial \psi_i} = \nu_i - |V_i^\psi|. \tag{8}$$

In other words, each component of the gradient corresponds to the prescribed volume ν_i associated with a point $\mathbf{x}_i$ minus the volume of the cell V_i^ψ. For the vector ψ^* that maximizes $K(.)$, all components of the gradient vanish, which means that each cell $V_i^{\psi^*}$ has exactly the prescribed volume ν_i.

This expression of the gradient leads also to a natural stopping criterion (line 5), the largest component of the gradient corresponds to the maximum volume error. I stop the algorithm as soon as it is smaller than a prescribed ϵ_K (typically one percent of ν_i).

I now consider the Hessian matrix computed at step (6). Adapting the formulas in [40,31] to our context, the coefficients of the Hessian are given by:

$$\begin{aligned}
\frac{\partial^2 K}{\partial \psi_i \partial \psi_j} &= \frac{1}{2}\frac{|V_{ij}^\psi|}{\|\mathbf{x}_j - \mathbf{x}_i\|} \quad \text{if } i \neq j \\
\frac{\partial^2 K}{\partial \psi_i^2} &= -\left(\sum_{j \neq i} \frac{\partial^2 K}{\partial \psi_i \partial \psi_j}\right) - \frac{1}{2}\frac{|V_{i0}^\psi|}{\sqrt{\psi_i}}
\end{aligned} \tag{9}$$

where $|V_{ij}^\psi| = |\partial V_i^\psi \cap \partial V_j^\psi|$ denotes the area of the intersection between the border of the cell V_i^ψ and the border of the cell V_j^ψ, that is, a polygonal facet in the Laguerre diagram clipped by the two balls $B(\mathbf{x}_i, \sqrt{\psi_i})$ and $B(\mathbf{x}_j, \sqrt{\psi_j})$. The term $|V_{i0}^\psi|$ corresponds to the free surface area in ∂V_i^ψ, that is the portion of the border of V_i^ψ included in the sphere $S(\mathbf{x}_i, \sqrt{\psi_i})$, that touches the unoccupied portion of Ω.

Step (7) of the algorithm computes the Newton step vector $\mathbf{p}$, by solving a linear system. Except the boundary term in $\partial^2 K/\partial \psi_i^2$, this linear system is identical as the one solved in [4]: it corresponds to a Poisson equation discretized with finite elements. The matrix is sparse, with a non-zero coefficient at coefficient (i, j) if and only if the cells V_i^ψ and V_j^ψ touch each other. The additional term $(1/2)\left(|V_{i0}^\psi|/\sqrt{\psi_i}\right)$ in $\partial^2 K/\partial \psi_i^2$ does not change the sparsity pattern of the Hessian: since the Lagrange multiplier associated with ψ_0 is fixed and equal to 0, there is no partial derivative with respect to it except the diagonal term. The same linear solver as in [4] can be used (Jacobi-preconditioned conjugate gradient [42] with sparse matrix stored in CRS format, with optional GPU acceleration), as well as the same stopping criterion $\|H\mathbf{p} - \mathbf{g}\|/\|\mathbf{g}\| \leq 10^{-3}$.

Once the step vector $\mathbf{p}$ is computed, a good descent parameter α needs to be found (step (8)). In the provably convergent KMT algorithm [40], the descent parameter α is determined as follows:

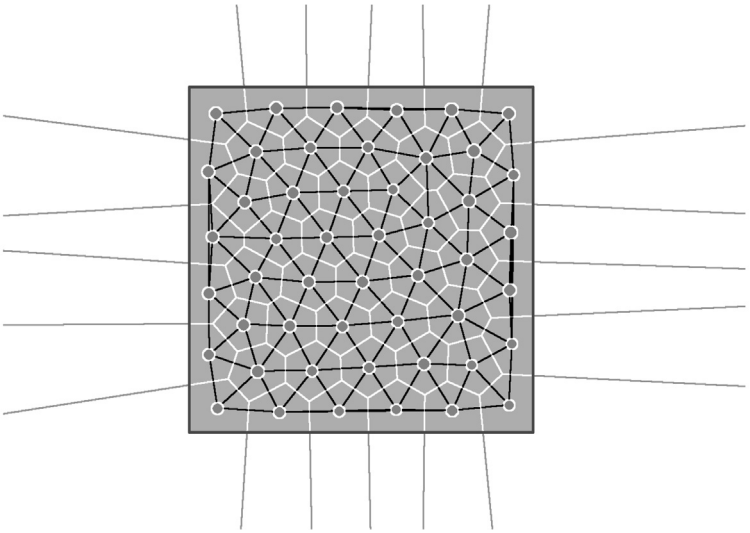

Fig. 6. The Bowyer-Watson algorithm computes the Laguerre diagram (cell boundaries displayed in white and light grey). It uses internally a dual representation (triangulation in black). One also needs to compute the intersection with Ω (the square in this example).

(1) : $\alpha \leftarrow 1$
(2) : Loop
(3) : If $\inf_i |V_i^{\psi+\alpha\mathbf{p}}| > a_0$
(4) : and $\|\nabla K(\psi+\alpha\mathbf{p})\| \leq (1-\alpha/2)\|\nabla K(\psi)\|$
(5) : then Exit loop
(6) : $\alpha \leftarrow \alpha/2$
(7) : Compute the cells $(V_i^{\psi+\alpha\mathbf{p}})_{i=1}^N$
(8) : End loop

where $a_0 = \frac{1}{2}\min\left(\inf_i |V_i^{\psi^{(0)}}|, \inf_i(v_i)\right)$.

The KMT algorithm iteratively halves the descent parameter α until two criteria are met: the volume of the smallest cell needs to be larger than a threshold a_0 (line 3), and the norm of the gradient needs to decrease sufficiently (line 4). The threshold a_0 for the minimum cell volume corresponds to (half) the minimum cell volume for $\psi = 0$ (also called Voronoi diagram) and minimum prescribed area v_i.

Equipped with the KMT algorithm above, one can compute the descent parameter α, by plugging the algorithm above into line (8) of the Newton algorithm at the beginning of this section.

5.2. Geometrical aspects

To evaluate the components of the gradient and coefficients of the Hessian of $K(.)$, the KMT algorithm needs to construct the cells $V_i^{\psi} = Lag_i^{\psi} \cap B(\mathbf{x}_i, \sqrt{\psi_i})$, that is, the intersection between Laguerre cells and balls centered on the $\mathbf{x}_i$'s for each $\psi^{(k)}$ iterate. To do so, I first compute the Laguerre diagram, using the classical Bowyer-Watson algorithm [43,44], see Fig. 6. A readily-available open-source implementation is available in our GEOGRAM library [45] (and also in the CGAL library [46] for programmers who have a taste for C++ templates). The details of our highly optimized implementation in GEOGRAM are given in [4].

Given the points $\mathbf{x}_i$ and the vector ψ, the Bowyer-Watson algorithm computes the Laguerre diagram. Internally, it uses the dual triangulation, formed by the black triangles in Fig. 6 (in 3D, they become tetrahedra). Each vertex of the Laguerre diagram is deduced from one of the triangles (resp. tetrahedra), and the Laguerre cells are obtained by traversing the triangles (tetrahedra) incident to a given $\mathbf{x}_i$. Note that the Boywer-Watson algorithm computes the Laguerre diagram in the entire $\mathbb{R}^d$ domain, with infinite Laguerre cells (grey straight lines in the Figure). To obtain the V_i cells, one needs to compute intersections with the domain Ω (and also with the balls centered on the $\mathbf{x}_i$'s):

$$V_i^{\psi} = Lag_i^{\psi} \cap \Omega \cap B(\mathbf{x}_i, \sqrt{\psi_i})$$

In 2D, constructing the cells V_i^{ψ} means computing intersections between convex polygons and disks, which does not represent too much difficulty (see Fig. 4). However, it becomes a challenging task in 3D: as can be seen, a large number of intersections needs to be computed, and 3D mesh intersection is known to be a delicate operation, subject to numerical

precision problems: the intersection algorithm depends on a small set of functions, called *geometric predicates*, that take as input some points, and that returns a discrete value. For instance, such a geometric predicate can tell whether a point $\mathbf{p}_1$ is above or below another point $\mathbf{p}_2$. These geometric predicates are often polynomials in the coordinates of the points. The limited precision of floating point numbers can have catastrophic consequences, for instance when the algorithm estimates that a certain point $\mathbf{p}_1$ is above another point $\mathbf{p}_2$, and later inconsistently estimates that $\mathbf{p}_2$ is above $\mathbf{p}_1$. Such inconsistent behavior can result in invalid combinatorics (see e.g. [47]).

It is important to stress that a *huge* number of intersections (billions) needs to be computed, hence these numerical issues that correspond to what may be initially thought of as very unlikely corner cases occur in fact *thousands times* in a typical simulation. The goal now is to find a strategy to overcome this robustness issue while keeping reasonable performance. The strategy is to express the problem in terms of a small number of "atomic" operations, that can be used as solid foundations to build the rest of the algorithm. These "atomic" operations are simple and can be robustly implemented:

- (1) the algorithm will solely manipulate *convex polytopes* and their intersections. A convex polytope C is defined as the intersection of N_v half-spaces Π_i^+:

$$C = \bigcap_{i=1}^{N_v} \Pi_i^+ = \bigcap_{i=1}^{N_v} \{\mathbf{x} \mid a_i x + b_i y + c_i z + d_i \leq 0\}$$

- (2) to avoid inconsistencies, all the used predicates will be computed in arbitrary precision [47], with arithmetic filters to speed-up computations in the easy cases [48].

A consequence of (1) is that the balls $B(\mathbf{x}_i, \sqrt{\psi_i})$ will be approximated by convex polytopes:

$$B(\mathbf{x}, R) \simeq \hat{B}(\mathbf{x}, R) = \bigcap_{i=1}^{N_u} \{\mathbf{y} \mid \|\mathbf{x} - \mathbf{y}\|^2 \leq \|\mathbf{x} + 2R\mathbf{u}_i - \mathbf{y}\|^2\}$$

where $(\mathbf{u}_i)_{i=1}^{N_u}$ is a predetermined set of unit vectors $\mathbf{u}$ that uniformly sample the unit sphere. While this strategy shares with the "ghost cells" approach the fact that it approximates the balls as convex polytopes, it differs from it in two important aspects:

- first, computations are *local* to a given V_i^ψ cell, with a small number of vertices (typically 50 to 200, as compared to the total number of points N), whereas the "ghost cells" strategy needs inserting all additional vertices into the *global* Laguerre diagram, with $N + M$ vertices;
- second, each $\hat{B}(\mathbf{x}, R)$ can be instanced from a *precomputed* unit sphere template $\hat{B}(\mathbf{0}, 1)$ without needing to compute any intersection, by scaling and translating the unit sphere template (no combinatorics needs to be computed).

This means a large number N_u (typically 100 to 200) of unit directions $\mathbf{u}_i$ can be used without significantly increasing the computation time: the "partial optimal transport" algorithm computes a solution that is equivalent to using $N_u \times N$ "ghost cells", without the associated computational cost. Not only does it significantly reduce the complexity, but also it makes it possible to use a large value for N_u, that gives a precise approximation of the balls $B(\mathbf{x}_i, \sqrt{\psi_i})$. Alternatively, one can also use a data structure for convex polytopes with spherical faces [49], but this comes with longer execution times, and robustness is more difficult to ensure.

The domain Ω can be either a convex polytope, or a simplicial mesh $\Omega = \bigcup_{i=1}^{N_T} T_i$. In both cases the intersections between Ω, Lag_i^ψ and $\hat{B}(\mathbf{x}_i, \sqrt{\psi_i})$ involved in the definition of V_i^ψ can be computed as intersections between convex polytopes.

5.2.1. Convex polytopes: H and V representation

The "atomic" operation computes the intersection between a convex polytope C and a half-space Π^+:

$$C \leftarrow C \cap \Pi^+.$$

In the end, to estimate the gradient and Hessian of $K(.)$, one needs to compute the volume of the cells and the areas of the cell borders. To do so, I need to convert from the initial representation of the cell, that is, as an intersection of half-spaces" (H-representation), $(\Pi_i^+)_{i=1}^{Nv}$ into an explicit representation of all the vertices and facets of the cell (V-representation). As shown in Fig. 7, if the planes are in generic position,[7] each vertex (black) is shared by three planes Π_i, Π_j, Π_k. It is then natural to represent the polytope in dual form (see Fig. 7), by a triangulation $(T_l = \{i_l, j_l, k_l\})_{l=1}^{N_t}$ shown in grey in the Figure (the same representation is exploited by the Boywer-Watson algorithm that I use to compute the Laguerre diagram). The coordinates q_x, q_y, q_z of the vertex $\mathbf{q}_l$ associated with triangle T_l are the solution of:

[7] I will discuss degenerate cases further on.

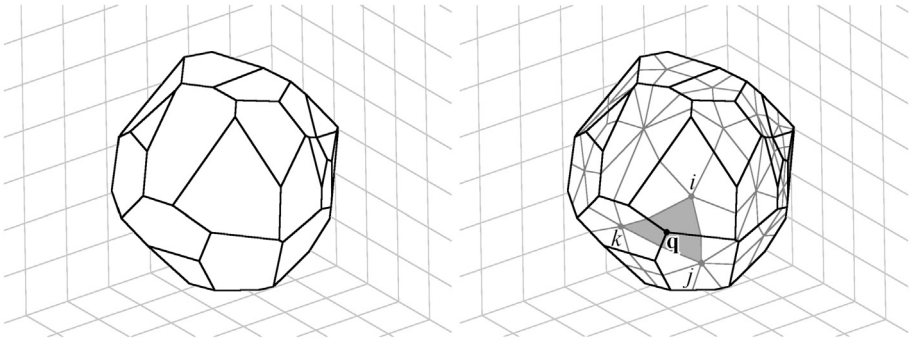

Fig. 7. Representation of convex polytopes.

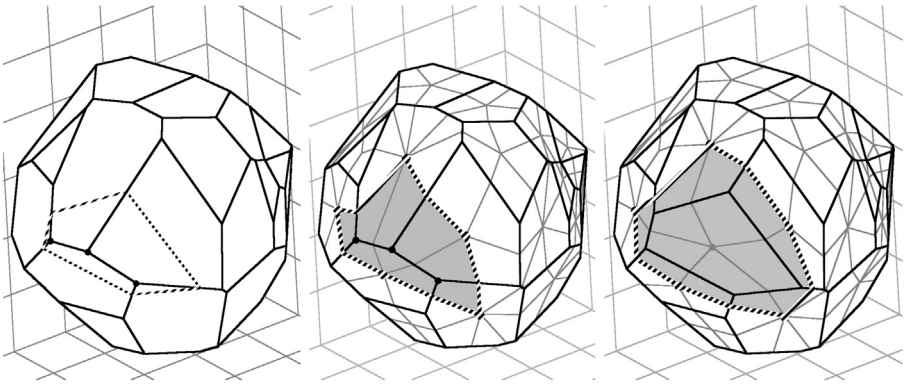

Fig. 8. Clipping a convex polytope by a half-space.

$$\begin{bmatrix} a_i & b_i & c_i \\ a_j & b_j & c_j \\ a_k & b_k & c_k \end{bmatrix} \begin{bmatrix} q_x \\ q_y \\ q_z \end{bmatrix} = - \begin{bmatrix} d_i \\ d_j \\ d_k \end{bmatrix}$$

Using Cramer's formula, the coordinates q_x, q_y, q_z are given by:

$$\begin{bmatrix} q_x \\ q_y \\ q_z \end{bmatrix} = -\frac{1}{\begin{vmatrix} a_i & b_i & c_i \\ a_j & b_j & c_j \\ a_k & b_k & c_k \end{vmatrix}} \begin{bmatrix} \begin{vmatrix} d_i & b_i & c_i \\ d_j & b_j & c_j \\ d_k & b_k & c_k \end{vmatrix} \\ \begin{vmatrix} a_i & d_i & c_i \\ a_j & d_j & c_j \\ a_k & d_k & c_k \end{vmatrix} \\ \begin{vmatrix} a_i & b_i & d_i \\ a_j & b_j & d_j \\ a_k & b_k & d_k \end{vmatrix} \end{bmatrix} \tag{10}$$

(the i-th component of the solution corresponds to the determinant of the system with its i-th column replaced with the r.h.s divided by the determinant of the system, see [37,50] for similar computations).

The algorithm to compute the intersection between a cell C and a half-space $\Pi^+(a, b, c, d)$ works in three phases (see Fig. 8):

Input: – A convex polytope C and a half-space Π^+
Result: – replaces C with $C \cap \Pi^+$
(1) : classify the vertices
(2) : discard the triangles
(3) : triangulate the hole

The first phase determines the triangles that correspond to vertices $\mathbf{q}_l$ that are on the negative side of Π^+ (highlighted in Fig. 8-Left), that is, $aq_x + bq_y + cq_z + d < 0$. This can be rewritten as:

$$\begin{vmatrix} a_i & b_i & c_i \\ a_j & b_j & c_j \\ a_k & b_k & c_k \end{vmatrix} \times \begin{vmatrix} a_i & b_i & c_i & d_i \\ a_j & b_j & c_j & d_j \\ a_k & b_k & c_k & d_k \\ a & b & c & d \end{vmatrix} \leq 0$$

by replacing the coordinates of $\mathbf{q}$ by their expression (one obtains an expression of the determinant above, developed w.r.t. to its last row). Note that the vertices of the triangles can be consistently oriented in such a way that the sign of the first term (the 3×3 determinant) is always positive. In the end, to classify a vertex, one only needs to compute the sign of the 4×4 determinant $|\Pi_i, \Pi_j, \Pi_k, \Pi|$ formed by the coefficients of the planes. This is the only predicate[8] used by the algorithm. Note also that during the computation of the intersections, only the equations of the Π_i planes are used, the $\mathbf{q}$ points do not need to be explicitly computed. To compute the exact sign of these determinants, I use expansion arithmetics [47], and arithmetic filters [48] to quickly determine the sign in the easy cases. The implementation in [51] generates the code that computes the predicate from its formula. This strategy ensures that all combinatoric decisions taken by the algorithm will remain consistent. The points $\mathbf{q}$ are computed once all clipping operations are computed, right before evaluating the volumes and areas involved in the gradient and Hessian of $K(.)$.

The second phase discards the triangles (i, j, k) such that $|\Pi_i, \Pi_j, \Pi_k, \Pi| < 0$ (highlighted in Fig. 8-Center). This leaves a hole in the triangulation (the border of the hole is shown as dashed lines). As often done in implementations of the Bowyer-Watson algorithm, the discarded triangles are kept in a linked list, so that they can be reused in the subsequent steps.

The third phase constructs a new triangle for each edge on the border of the hole created at the previous phase (see Fig. 8-Right).

The algorithm has the same structure as the Bowyer-Watson algorithm, with the difference that the `in_sphere` predicate, that determines which triangle to discard, is replaced with $\texttt{sign}|\Pi_i, \Pi_j, \Pi_k, \Pi|$. Another difference is the way the Bowyer-Watson determines the list of triangles to be discarded. In my case I test all the triangles. I could instead adapt the Bowyer-Watson strategy, that starts from a random vertex, and walks along the edges of the dual graph until it finds a triangle to be discarded, but since the number of triangles per polytope remains small in our case (typically < 100), I did not observe a speedup using this technique.

5.2.2. Computing the cells V_i^ψ

I shall now explain how to use the algorithm of the previous paragraph to compute the cells. Each cell $V_i^\psi = Lag_i^\psi \cap \Omega \cap \hat{B}(\mathbf{x}_i, \sqrt{\psi_i})$ is the intersection between three objects: the Laguerre cell Lag_i^ψ, the domain Ω, represented by a simplicial mesh, that is, a set of tetrahedra, and the (approximated) ball $\hat{B}(\mathbf{x}_i, \sqrt{\psi_i})$. Note that the domain Ω is not necessarily convex, but since it is decomposed into simplices, I only need to compute intersections between convex objects. The algorithm that computes a cell V_i^ψ is detailed below:

Input: – a simplicial mesh Ω
– the set of points $(\mathbf{x}_i)_{i=1}^N$ and the vector $(\psi_i)_{i=1}^N$
– the index i of a cell
Output: – A set of convex polytopes $\mathbf{C} = \{C_k\}$ such that $V_i^\psi = \bigcup C_k$

(1): $\mathcal{T} \leftarrow \{t \subset \Omega \mid B(\mathbf{x}_i, \sqrt{\psi_i}) \cap t \neq \emptyset\}$
(2): If $\exists t \in \mathcal{T} \mid \partial t \cap \partial\Omega \neq \emptyset$
(3): For each $t \in \mathcal{T}$
(4): $C \leftarrow t$
(5): $C \leftarrow C \cap Lag_i^\psi$
(6): $C \leftarrow \hat{B}(\mathbf{x}_i, \sqrt{\psi_i}) \cap C$
(7): $\mathbf{C} \leftarrow \mathbf{C} \cup \{C\}$

[8] besides `orient3d` and `in_weighted_sphere` used by the Bowyer-Watson algorithm that constructs the Laguerre diagram.

(8) : End for

(9) : Else

(10) : $C \leftarrow Lag_i^{\psi}$

(11) : $C \leftarrow \hat{B}(\mathbf{x}_i, \sqrt{\psi_i}) \cap C$

(12) : $\mathbf{C} \leftarrow \mathbf{C} \cup \{C\}$

(13) : End if

In line (1), I first determine the set of simplices $\mathcal{T}$ that have a non-empty intersection with the ball $B(\mathbf{x}_i, \sqrt{\psi_i})$. It is done using an axis-aligned bounding box tree (AABB tree), see e.g. [52].

Then I distinguish two different cases: **first case:** if one of the intersected tetrahedra touches the boundary $\partial\Omega$ of the domain, then V_i^{ψ} is not necessarily convex, but can be easily decomposed into convex objects: to do so, I compute the intersections $Lag_i^{\psi} \cap \hat{B}(\mathbf{x}_i, \sqrt{\psi_i}) \cap t$ for each t in $\mathcal{T}$. **Second case:** the ball $B(\mathbf{x}_i, \sqrt{\psi_i})$ is included in Ω (no tetrahedron in $\mathcal{T}$ touches $\partial\Omega$), then $V_i^{\psi} = Lag_i^{\psi} \cap B(\mathbf{x}_i, \sqrt{\psi_i})$ is convex and can be directly computed (lines (9) to (13)).

In lines (6) and (11), I compute the intersection between a convex polytope and the approximated ball $\hat{B}(\mathbf{x}_i, \sqrt{\psi_i})$. If C's bounding sphere centered on $\mathbf{x}_i$ has a radius smaller than $\sqrt{\psi_i}$ (that is, $\max_{\mathbf{q}\in C}\{\|\mathbf{x}_i - \mathbf{q}\| < \sqrt{\psi_i}\}$), then C is entirely contained in the ball and one can skip the intersection computation. Else, the approximated ball $\hat{B}(\mathbf{x}_i, \sqrt{\psi_i})$ is instanced (copied from the unit ball, translated and scaled), and then clipped by all the Π_i's from the H-representation of C (in this order, because $\hat{B}(\mathbf{x}_i, \sqrt{\psi_i})$ has a larger number of facets than C in general).

In line (4), the convex polytope C is initialized from a tetrahedron of Ω. Combinatorics are initialized from a fixed tetrahedron template, and the plane equation of each facet $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$ is given by $ax + by + cz + d = 0$, where $(a, b, c) = (\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_3 - \mathbf{p}_1)$ and $d = -(a, b, c) \cdot \mathbf{p}_1$.

In line (10), C is initialized from a Laguerre cell. Combinatorics are copied from the Laguerre diagram, also stored in dual representation (tetrahedra). The triangles of C correspond to the facets of the tetrahedra incident to vertex i opposite to i. The plane equations $\Pi_{i,j}$ are given by: $\mathbf{x} \in \Pi_{ij} \quad \Leftrightarrow \quad \|\mathbf{x} - \mathbf{x}_i\|^2 - \psi_i = \|\mathbf{x} - \mathbf{x}_j\|^2 - \psi_j$ or: $ax + by + cz + d = 0$ where $(a, b, c) = 2(\mathbf{x}_i - \mathbf{x}_j)$ and $d = \mathbf{x}_j^2 - \mathbf{x}_i^2 - \psi_j + \psi_i$.

Up to now I have supposed that all planes were in generic position (that is, each vertex of the polytope is shared by exactly three planes). To handle the specific configurations where more than three planes meet at a single vertex, one would think of using the symbolic perturbations, as in the "simulation of simplicity" technique [50]. In our case, the symbolic perturbation is trivial: one can use the predicate $|\Pi_i, \Pi_j, \Pi_k, \Pi| \leq 0$ as is (provided that it is implemented with exact arithmetics, see [51] and references herein). It means that each time a vertex $\mathbf{q}$ is exactly located on the boundary of a clipping half-space Π^+ the associated triangle will be discarded. To evaluate the efficiency of this technique, degenerate configurations are tested in the next section.

It is not necessary to store the set of convex polytopes $\mathbf{C}$: one can process them in a "streaming" manner, when assembling the gradient and Hessian of $K(.)$. Then, the algorithm stores a single C at any time, and has only a small memory overhead besides the storage taken by the Laguerre diagram (stored as a 3D triangulation). Moreover, the algorithm is very simple to parallelize (then it stores one convex polytope per thread). When assembling the gradient and Hessian, it just needs a synchronization primitive (spinlock) to ensure that two threads do not modify the same entry at the same time.

6. Numerical experiments

I shall now test the different components of the algorithm, first the lowest-level one (convex polytopes clipping), then the partial optimal transport, and finally demonstrate the algorithm used to implement a basic free-surface fluid simulator in 3D.

6.1. Testing convex polytope clipping

The robustness of the convex polytope clipping algorithm is tested using some highly degenerate configurations that are created on purpose. In Fig. 9-Left, the algorithm computes the intersection between 10000 (top) and 30000 (bottom) half-spaces tangent to a sphere with random normal vector. The same sequence of half-spaces is fed to the algorithm 5 times, which creates degenerate configurations with half-spaces that exactly correspond to existing facets. On the right, a cube is clipped by 30 (top) and 500 (bottom) random half-spaces tangent to a cone. This creates a highly degenerate vertex, common to all facets. Again, the sequence of half-spaces is fed to the algorithm 5 times. In both configurations, the implemented exact predicates robustly handles all the degeneracies. Execution time is smaller than 1 second for both tests.

6.2. Testing partial optimal transport

I shall now test the partial optimal transport algorithm in simple geometric settings. In the numerical experiment displayed in Fig. 10, the lower half of a cube is homogeneously sampled with a set of points, and partial optimal transport

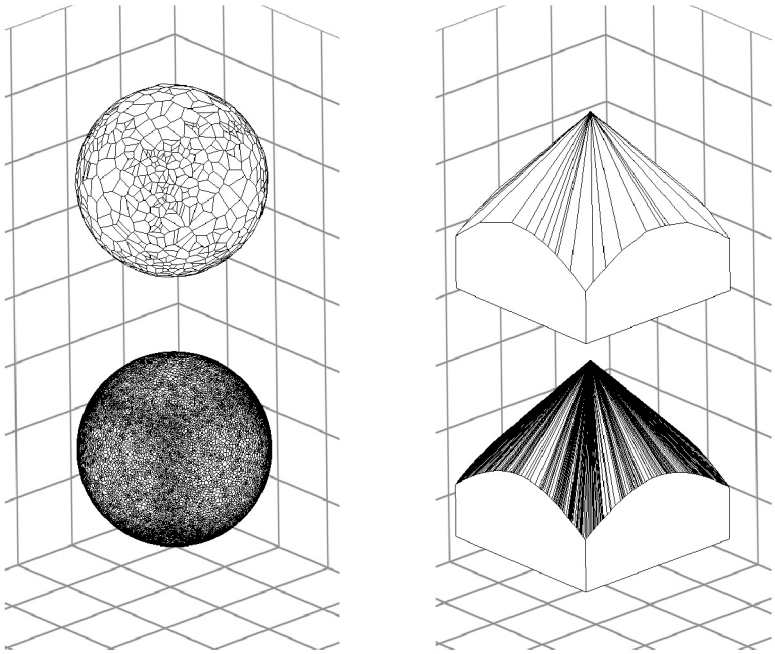

Fig. 9. Testing the robustness of the convex polytope clipping algorithm.

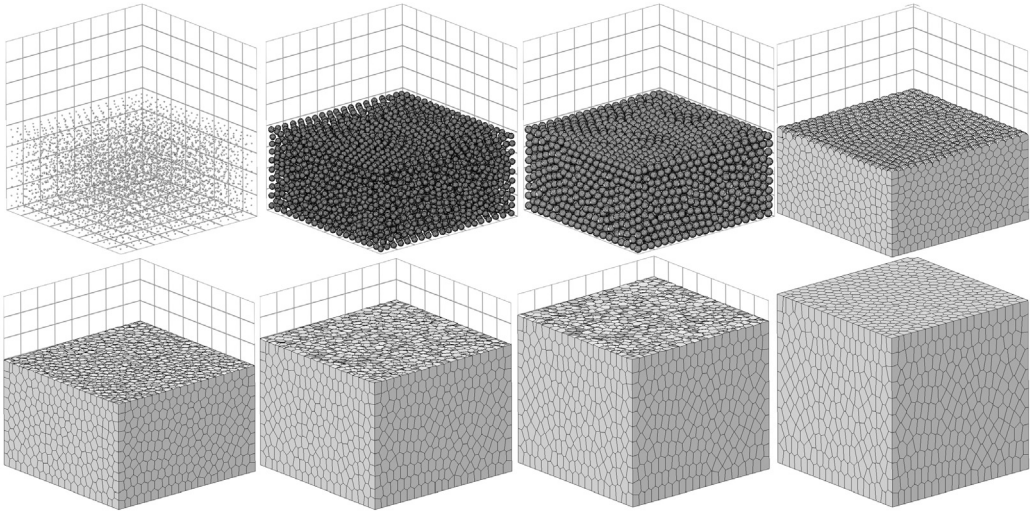

Fig. 10. Testing partial optimal transport between cube and points sampled in lower-half of the cube. Varying fluid volume from 0.1 to 1.0.

is computed with different values for the fluid volume. In this configuration, if the fluid volume v is lower than 0.5, the expected result is a set of spheres centered on the points, and if v is greater than 0.5, the expected result is a paralleleliped of height v. Newton iterations are stopped when the largest cell volume error is less than 1%. The experimental results are displayed in Fig. 10 and the volume errors and numbers of Newton iterations are reported in Table 1. Note that the points remain in the lower half of the cube. Clearly, for values of v greater than 0.5, some cells do not contain the points they are associated with, forming a configuration that is similar to the 2D one depicted in Fig. 1-right on page 4). It can be also remarked that starting from $v = 0.5$, the required number of Newton iterations increases. This is because there are more interactions between the cells that "compete" to fill the volume, that is, the Hessian of K is less sparse.

In Fig. 11, I am testing a configuration with a non-convex domain, represented by a tetrahedralized mesh. A single cell is moving vertically in a zone that has a "constriction", and the volume of the cell is constrained to remain constant (Table 2). Newton iterations are stopped when the cell volume error is less than 1%. The algorithm accurately computes the intersections between the cell and the domain, even with geometric details that are smaller than the cell. Note that these intersections need to be computed only when the cell has a non-empty intersection with a tetrahedron adjacent to the

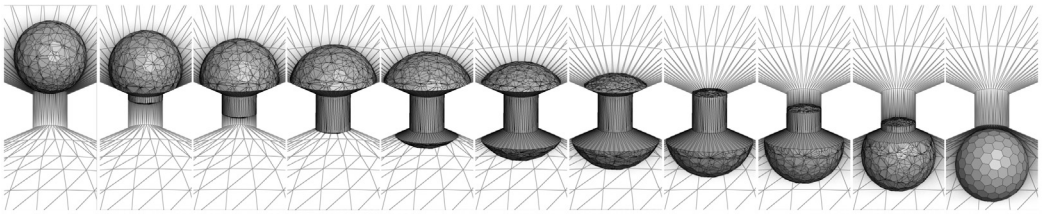

Fig. 11. Testing partial optimal transport in domain of arbitrary shape. Cell passing through a constriction and keeping its volume constant.

Table 1
Partial optimal transport in a cube. Total volume error.

fluid volume	0.1	0.3	0.4	0.5
vol. error	$3.2e-5$	$1.6e-5$	$4.1e-4$	$3.0e-5$
Newton iter.	3	3	3	9
fluid volume	0.625	0.75	0.875	1.0
vol. error	$3.0e-4$	$1.7e-6$	$4.1e-4$	0.0
Newton iter.	9	9	8	6

Table 2
Partial optimal transport in domain of arbitrary shape.

timestep	1	2	3	4	5	
vol. error	$3.19e-5$	$2.40e-5$	$2.84e-4$	$2.10e-3$	$1.43e-4$	
Newton iter.	3	3	3	3	4	
timestep	6	7	8	9	10	11
vol. error	$2.22e-4$	$1.43e-4$	$2.10e-3$	$2.84e-4$	$2.40e-5$	3.19e-5
Newton iter.	4	4	3	3	3	3

Table 3
Enright test, surface L_1 error.

N	$t=0$	$t=25$	$t=50$	$t=75$	$t=100$	mean	order
$5K$	$1.14e-3$	$1.35e-3$	$1.33e-3$	$1.23e-3$	$1.30e-3$	$1.27e-3$	N/A
$30K$	$6.18e-4$	$7.17e-4$	$6.40e-4$	$7.26e-4$	$7.77e-4$	$6.95e-4$	1.01
$150K$	$3.69e-4$	$4.19e-4$	$3.83e-4$	$4.60e-4$	$5.21e-4$	$4.30e-4$	0.895
$500K$	$2.54e-4$	$2.82e-4$	$2.60e-4$	$3.30e-4$	$3.92e-4$	$3.04e-4$	0.864
$1M$	$2.12e-4$	$2.28e-4$	$2.07e-4$	$2.77e-4$	$3.40e-4$	$2.52e-4$	0.811

boundary of the domain, hence the overall timing of the algorithm remains reasonable (see simulation results and timings in the next subsection).

6.3. Testing surface tracking

I shall now test the algorithm used in a typical surface tracking scenario, that is Enright test [27] and Zalesak test [53]. Enright test consists in applying a periodic motion to a sphere. The motion is specified in analytic form, and designed to introduce important surface variations, while the volume is supposed to stay constant. To measure the accuracy of the method, I first constructed a set of reference surfaces, by applying the motion to a high-resolution surface (500K vertices), remeshed at each timestep using the algorithm in [54]. The reference surface is validated by measuring both total volume loss (less than 3%) and distance between the surfaces at the initial and final timesteps (less than 8e-4). Then I applied the motion field to a pointset obtained by uniformely sampling a sphere. At each timestep, to ensure the sampling remains uniform, the pointset is regularized by moving each point to the centroid of its Laguerre cell [55]. The result with 30K cells is displayed in Fig. 12. Worst cell volume error and total volume loss are reported in Tables 4 and 5 respectively. The geometric surface error for different numbers of cells is reported in Table 3. In our Lagrangian setting, it is estimated as the integral of the distance to the nearest point on the reference surface. Since the reconstructed surface is approximated as sphere portions, without a curvature-adapted sampling the method is of order at most 1. A similar result (order $\simeq 1$) is obtained for the Zalesak test (Fig. 13 and Table 6). Note that in our purely Lagrangian setting, and since our partial optimal transport method is invariant by a rigid motion, the resconstructed surface at each timestep of the Zalesak test is the same (up to the rigid motion).

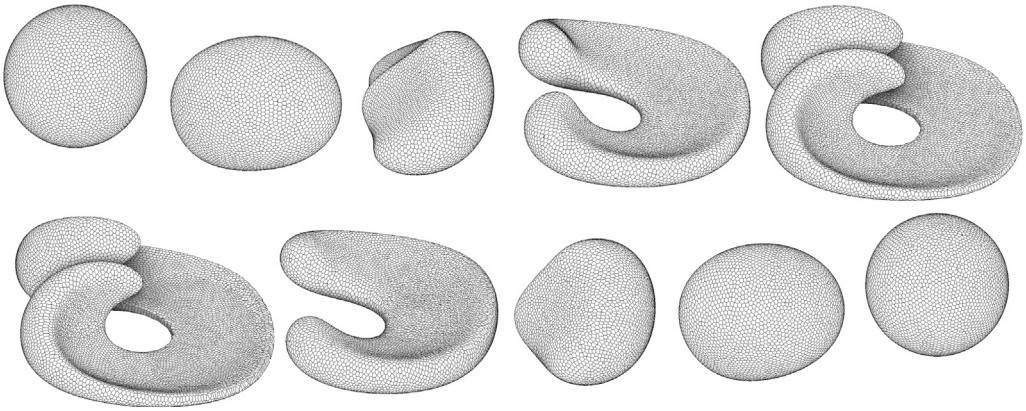

Fig. 12. Enright test, using 50K cells.

Table 4
Enright test, worst cell volume error (relative to cell volume).

N	$t=0$	$t=25$	$t=50$	$t=75$	$t=100$
$5K$	$3.80-3$	$3.54e-4$	$3.89e-4$	$7.84e-3$	$7.87e-4$
$30K$	$6.22e-4$	$8.09e-3$	$2.27e-3$	$9.55e-3$	$4.02e-3$
$150K$	$2.73e-3$	$9.96e-3$	$2.22e-3$	$4.34e-3$	$5.49e-3$
$500K$	$3.28e-3$	$4.54e-3$	$2.28e-3$	$1.44e-3$	$3.28e-3$
$1M$	$7.26e-3$	$6.09e-3$	$2.29e-3$	$5.63e-3$	$1.18e-3$

Table 5
Enright test, total volume loss.

N	$t=0$	$t=25$	$t=50$	$t=75$	$t=100$	mean
$5K$	$1.55e-4$	$1.12e-5$	$9.58e-6$	$1.53e-3$	$2.17e-5$	$3.45e-4$
$30K$	$9.43e-5$	$2.02e-4$	$1.42e-5$	$2.66e-4$	$3.35e-5$	$1.22e-4$
$150K$	$1.13e-5$	$9.44e-5$	$1.03e-5$	$1.51e-5$	$2.56e-5$	$3.13e-5$
$500K$	$5.63e-6$	$2.71e-5$	$2.52e-6$	$6.64e-7$	$5.42e-6$	$8.26e-6$
$1M$	$1.35e-5$	$1.44e-5$	$1.63e-6$	$9.98e-6$	$1.75e-6$	$8.25e-6$

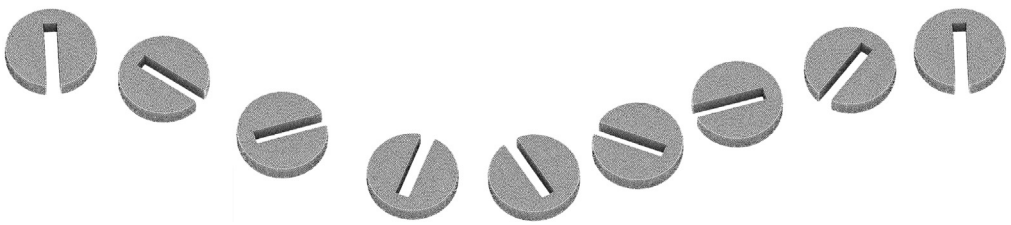

Fig. 13. Zalesak test, 30K vertices.

Table 6
Zalesak test, surface L_1 error.

N	surface L_1 error	order
$5K$	$6.60e-4$	N/A
$30K$	$3.64e-4$	1
$150K$	$2.16e-4$	0.970
$300K$	$1.72e-4$	0.988

6.4. Testing the Lagrangian mesh representation: a toy free-boundary fluid simulator

I shall now demonstrate how the algorithm can be used to implement a Lagrangian mesh with controlled volume that can change topology, with a very simple incompressible Navier-Stokes simulator for fluids with free boundaries. In the context of this article, this "toy" simulator is meant to demonstrate that our Lagrangian mesh can reproduce some of the

typical behaviors of a fluid while accurately tracking collisions and changes of topology. A more extensive quantitative evaluation of the simulator and the calibration of its parameters will be the subject of another article.

The simulator is obtained by adding viscosity and surface tension terms to the Gallouet-Merigot scheme [56], that simulates an incompressible Euler fluid. Inspired by Brenier and Benamou's point of view on incompressible fluids [36,57], the Gallouet-Merigot scheme softly projects the motion onto the incompressibility constraint by adding a "spring" force, that moves each point $\mathbf{x}_i$ towards the centroid of its cell V_i. This additional "pressure" force makes the motion of the fluid tangent to the manifold of incompressible motions, and the computed fluid motion provably converges to the solution of an incompressible Euler fluid, that is, a geodesic on the manifold of incompressible motions (see [58,59] on geodesic flows and the Euler-Arnold equation). The reader is referred to [56] for all the details.

The fluid is modeled as a set of V_i cells, parameterized by the $\mathbf{x}_i$ points. The volume of each V_i cell is controlled and remains constant throughout the whole simulation. All the cells have the same mass m. Each cell is subjected to the following forces:

$$
\begin{aligned}
F_p(\mathbf{x}_i) &= \tfrac{1}{\epsilon_p^2}(\mathbf{g}_i - \mathbf{x}_i) \qquad \text{where } \mathbf{g}_i = \tfrac{1}{|V_i|}\textstyle\int_{V_i} \mathbf{x}dx \\
F_g(\mathbf{x}_i) &= -mg\mathbf{z} \\
F_v(\mathbf{x}_i) &= \mu\hat{\Delta}\mathbf{v}(\mathbf{x}_i) \quad = \mu\textstyle\sum_{j\in N_i} \frac{1}{2\|\mathbf{x}_j-\mathbf{x}_i\|}\,|V_{ij}|\,(\mathbf{v}_j-\mathbf{v}_i) \\
F_t(\mathbf{x}_i) &= \gamma\hat{\Delta}\mathbf{x}(\mathbf{x}_i) \quad = \gamma\textstyle\sum_{j\in N_i} \frac{1}{2\|\mathbf{x}_j-\mathbf{x}_i\|}\,|V_{ij}|\,(\mathbf{x}_j-\mathbf{x}_i).
\end{aligned}
$$

Following [56], the "pressure" F_p may be thought of as a "spring" that connects each point $\mathbf{x}_i$ to the centroid $\mathbf{g}_i$ of the associated cell V_i with a stiffness ϵ_p. The additional forces are viscosity and surface tension: viscosity F_v is proportional to the Laplacian of the velocity. Surface tension F_t is derived in a volumetric manner, from the mutual attraction in the fluid. Inside the fluid, the net result is zero. On the boundary, the net result is a force that pulls towards the interior of the fluid. Both viscosity and surface tension use the $\mathbb{P}_1$ Laplacian:

$$
\hat{\Delta} f(\mathbf{x}) = \sum_{j\in N_i} \frac{1}{2\|\mathbf{x}_j-\mathbf{x}_i\|}\,|V_{ij}|\,(f(\mathbf{x}_j)-f(\mathbf{x}_i))
$$

where $|V_i j| = |\partial V_i \cap \partial V_j|$ denotes the area of the facet common to the cells V_i and V_j, and where N_i denotes the set of cells touching V_i. In the expressions of F_v and F_t, it is computed component-wise.

The simulation results shown below use a semi-implicit time-stepping, with implicit integration for the viscosity, as follows:

$$
\begin{aligned}
\mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} + \delta t\mathbf{v}^{(k)} \\
\mathbf{v}^{(k+1)} &= \mathbf{v}^{(k)} + \tfrac{\delta t}{m}\left(F_p + F_g + F_t + \mu\hat{\Delta}\mathbf{v}^{(k+1)}\right) \\
\text{or: } \left(\mathrm{Id} - \mu\tfrac{\delta t}{m}\hat{\Delta}\right)\mathbf{v}^{(k+1)} &= \mathbf{v}^{(k)} + \tfrac{\delta t}{m}\left(F_p + F_g + F_t\right) \\
\text{or: } \left(\hat{\Delta} - \tfrac{m}{\mu\delta t}\mathrm{Id}\right)\mathbf{v}^{(k+1)} &= -\tfrac{1}{\mu}\left(\tfrac{m}{\delta t}\mathbf{v}^{(k)} + F_p + F_g + F_t\right).
\end{aligned}
$$

The linear system is solved for each component x, y, z separately. Note that the matrix of this linear system corresponds to $\hat{\Delta}$ plus a diagonal term. To assemble it, the same code that constructs the Hessian of $K()$ can be reused.

Sub-stepping is used whenever the CFL condition is violated, that is whenever a displacement $\delta t\ \mathbf{v}$ is such that more than one cell is crossed, that is whenever there exists an i such that $\delta_t\ |\mathbf{v}_i| > \sqrt{\psi_i}$.

The method is tested in four different configurations (see videos in the webpage https://members.loria.fr/blevy/papers/POT):

- a simulated crown splash, shown in Fig. 14, computed with 500000 cells. The simulated mesh captures all the changes of topology, and preserves the fine details of the motion throughout the simulation;
- a fluid in a zigzag domain (Fig. 15), demonstrating the interactions between the fluid and the boundary of the domain;
- the same zigzag domain (Fig. 16) with a higher surface tension;
- a simulation of the experimental results in [60] shown in Fig. 17, where a droplet slides along a slope and bounces over obstacles. An example of a simulation mesh is shown in Fig. 18, as well as a cross-section, revealing the internal structure of the mesh, and the effect of surface tension that clusters the points $\mathbf{x}_i$ near the surface (due to the volume constraint, the associated cells are more elongated than the others). Some frames of the full animation are shown in Fig. 19. The fluid-boundary interactions and the typical oscillations of the droplet are reproduced in the simulation.

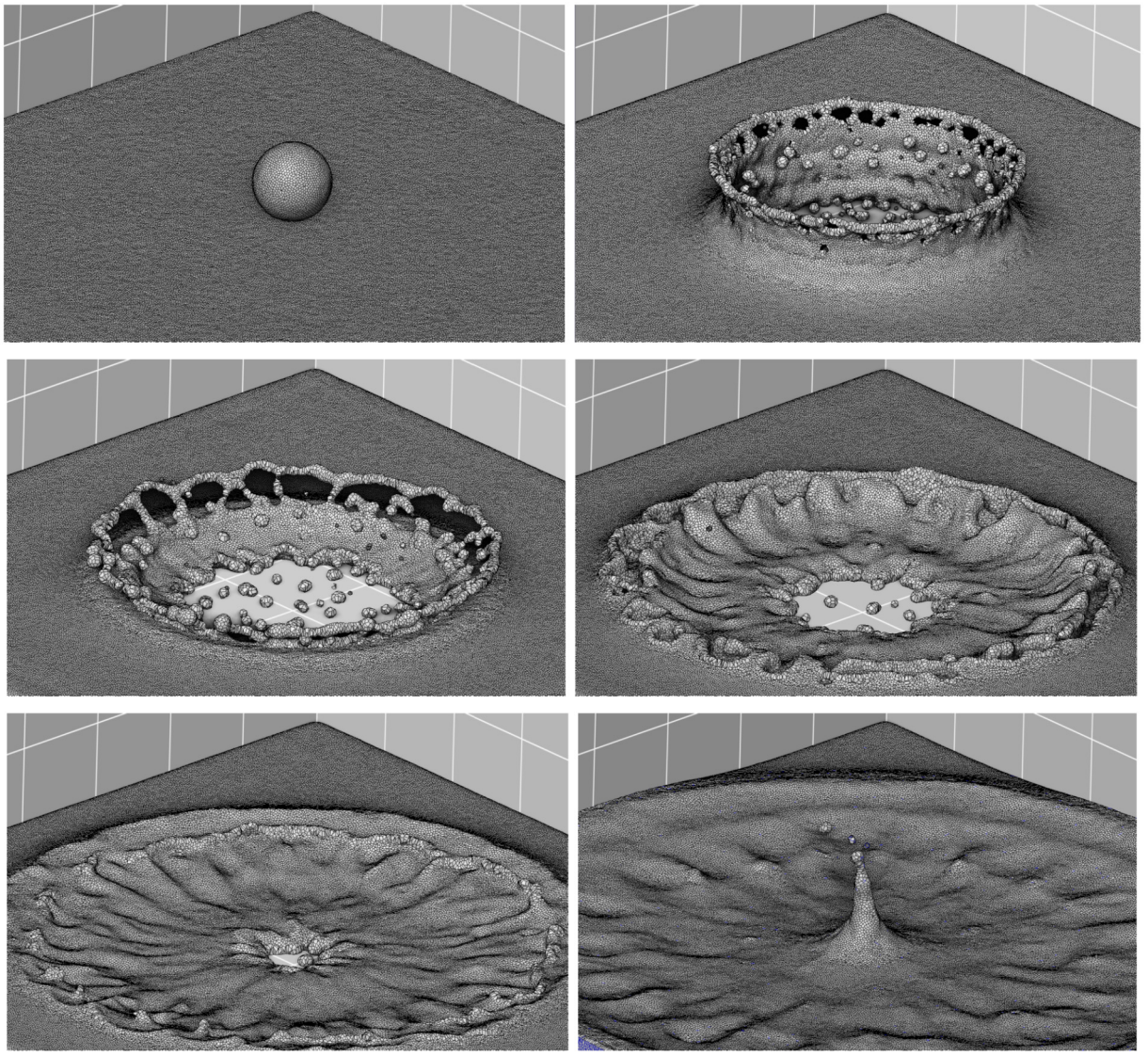

Fig. 14. Testing surface tension and topology changes: crown splash.

The table below indicates the parameters for the four simulations:

	g	m	μ	γ	N	time (per timestep)
Splash	10	3	0.001	1.5	500000	98s
Zigzag 1	1	3	0.001	0.5	25000	12s
Zigzag 2	1	3	0.001	2	5000	2.8s
Hurdles	0.3	3	0.001	3	5000	3.5s

The table reports the parameters for gravity (g), viscosity (μ), surface tension (γ), the number of cells (N) and the computation time for one timestep. The domain Ω is normalized in the $[0,1]^3$ box. For fine-scale tension-dominated simulations (zigzag and hurdles), the simulation space is globally scaled-down by applying a scaling factor to g. In the four simulations, the threshold for Newton iterations ϵ_K is set to 1%. During the whole simulation, each cell volume $|V_i|$ remains within a 1% tolerance from the specified volume. Note that unlike some methods that use an Eulerian mesh, volume errors do not accumulate, the 1% bound is met by all cells at all timesteps. The parameter ϵ_p for the "pressure" is set to 0.004, and the timestep is set to 0.004. The spheres are approximated using $N_u = 162$ half-spaces.

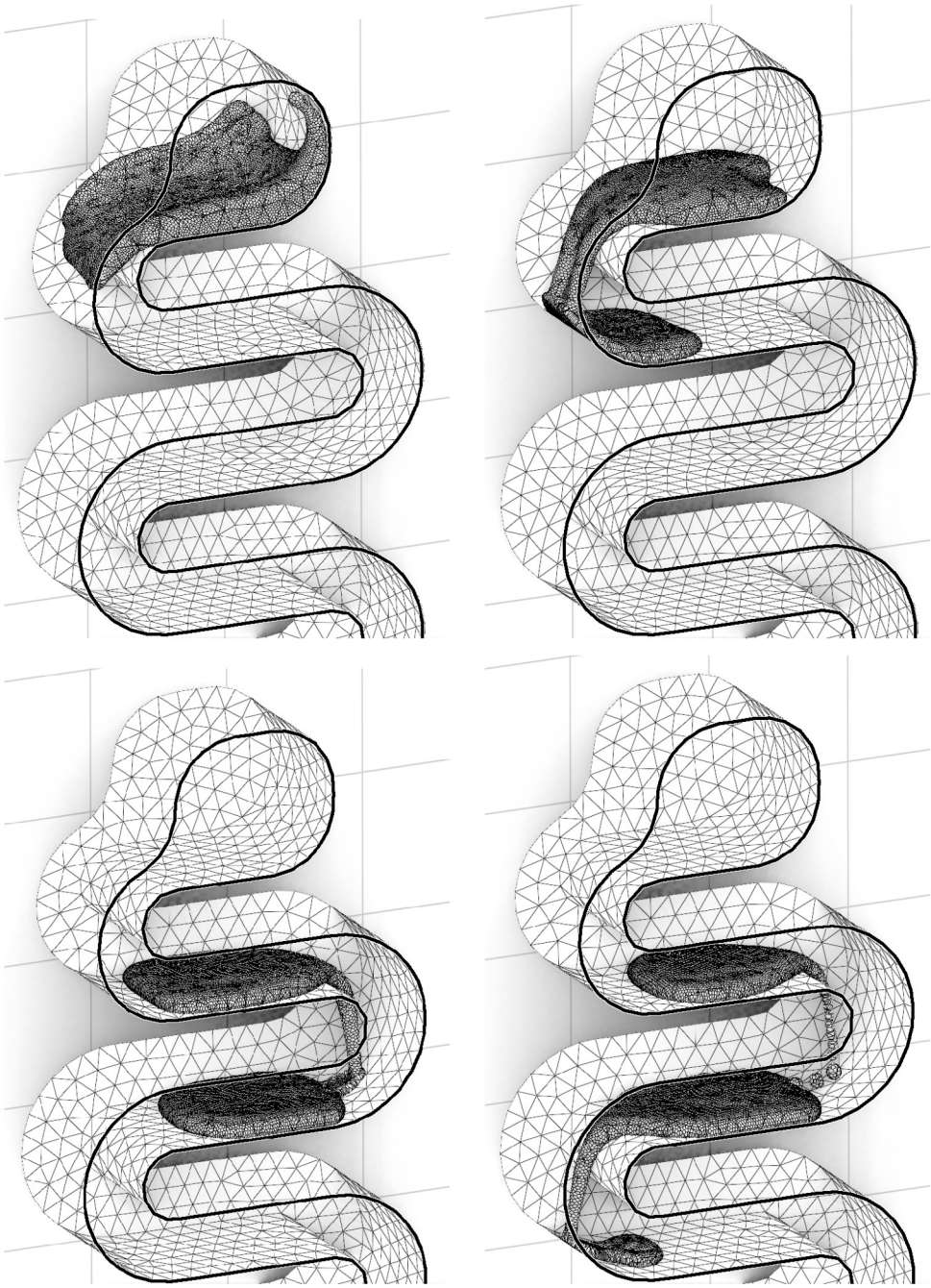

Fig. 15. Free-surface flow and interactions with boundaries: moderate surface tension.

I shall now report timing breakdown, measured with the crown splash example (Intel Core i9-9880H, 2.3 GHz, multi-threaded implementation):

phase	time
Laguerre (Bowyer-Watson)	9.7%
Linear solve	2.2%
Evaluate gradient	32.8%
Evaluate and assemble Hessian	41.5%
Euler update and implicit viscosity lin. solve	13.8%

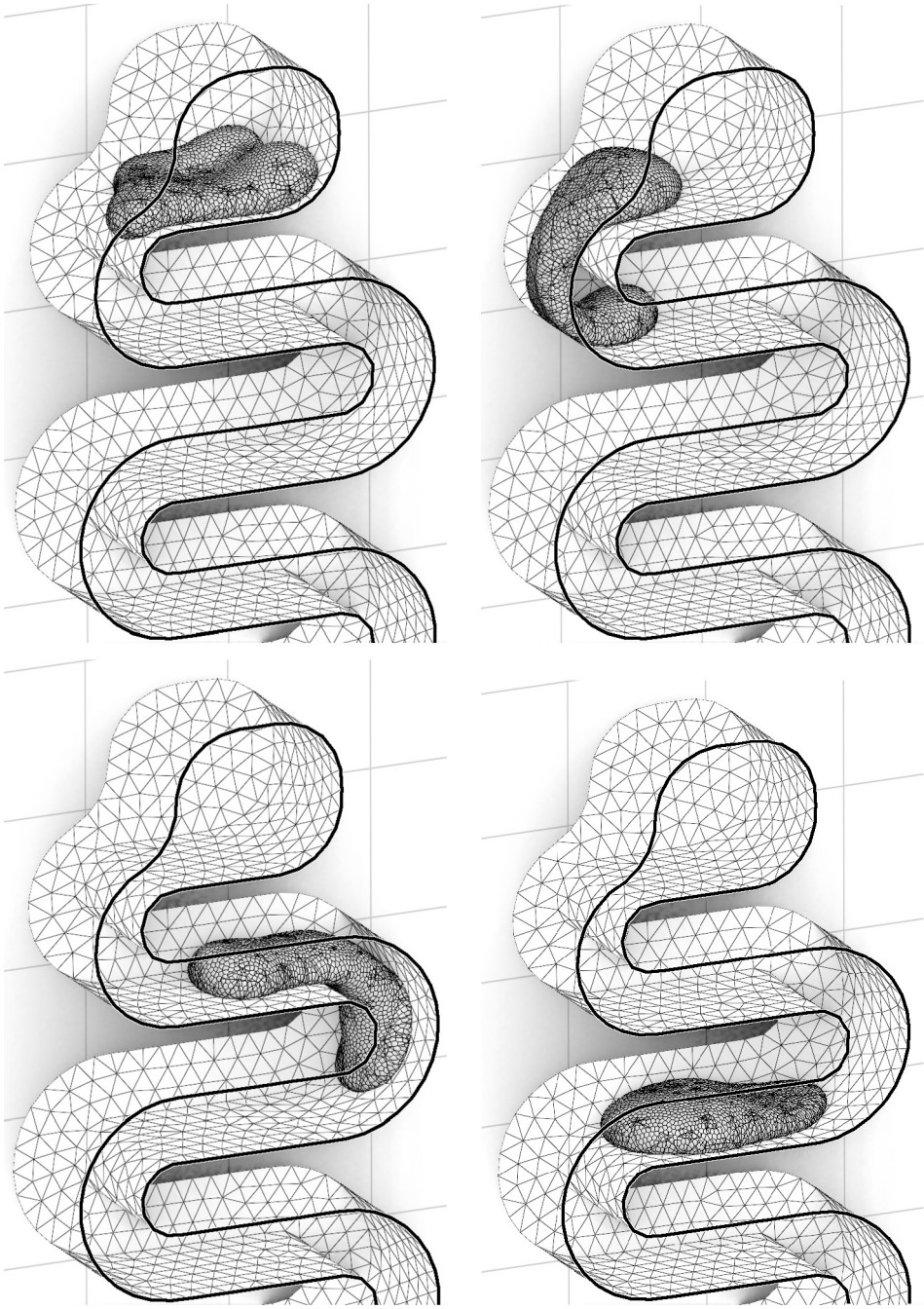

Fig. 16. Free-surface flow and interactions with boundaries: higher surface tension.

The phases that take the largest amount of time are evaluating the gradient and Hessian. It is because these phases involve all the geometric computations (query the Axis-Aligned Bounding Box tree, compute intersections between convex polytopes and compute volumes and areas).

Concerning the geometric predicate used by the convex polytope clipping algorithm (the sign of a 4x4 determinant), during 100 timesteps of the crown splash example, it was called 15752996161 times (15.7 billions), and the arithmetic filter triggered the arbitrary precision mode 70034 times (that is, 0.0004%). Clearly, degenerate configurations represent a tiny proportion, but given the huge number of computed intersections, they appear in each simulation. Our robust intersection algorithm properly handles them, with negligible overhead, thanks to the arithmetic filters (see also [51] and references herein).

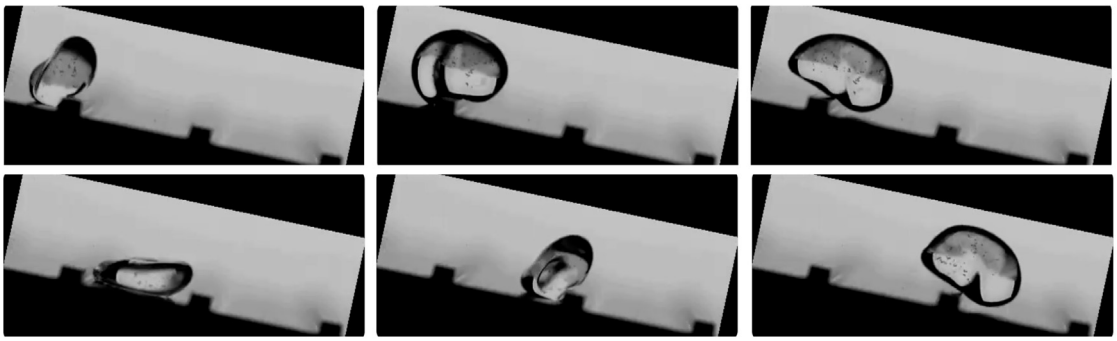

Fig. 17. Droplet hurdles race, courtesy Hélène de Maleprade, Rachid Bendimerad, Christophe Clanet and David Quéré.

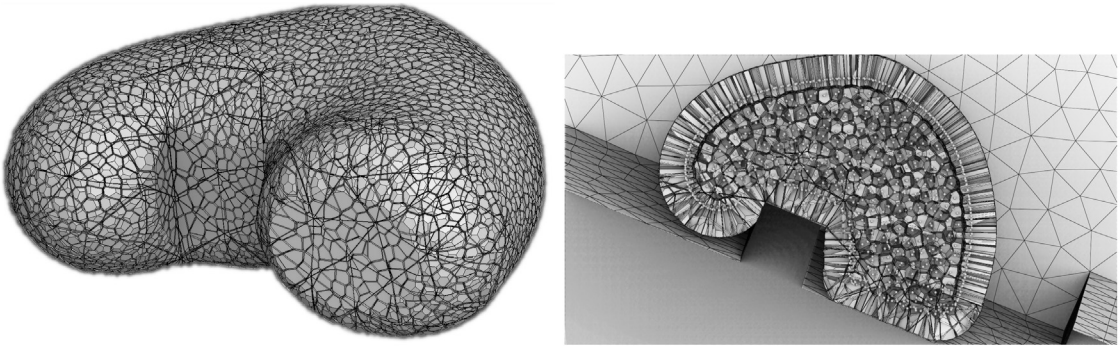

Fig. 18. Simulated droplet hurdles race: Mesh details and cross-section.

7. Conclusions and future works

The experimental results tend to confirm that semi-discrete partial optimal transport can be used to implement a Lagrangian scheme for free-surface fluid simulation. The precise analysis, comparison to state of the art codes and calibration of the parameters will be the topic of another article. Note also that the convergence to the Merigot-Gallouet scheme to the solution of the incompressible Euler equation is guaranteed [56], but it is no longer the case for Navier-Stokes, with the added viscosity and surface tension terms.

I used here a simple semi-implicit time stepping, it would be interesting to implement a symplectic scheme: in the end this will result in a simulator with both precise conservation of volume and precise conservation of the Hamiltonian. However, this will require a careful analysis for sub-stepping.

Adaptivity is another aspects that would be interesting to study, that is, creating and deleting cells dynamically, during the simulation, using larger cells in zones where the flow is simple, and smaller cells near the boundary of the fluid, to better capture the fluid shape and to more accurately represent surface tension (see Fig. 20). However, the "toy" simulator implemented here does not support exchanging matter between the cells. A principled procedure to compute the evolution of cell volumes remains to be developed. To even better adapt the geometry of the flow, anisotropic cells could be used, defined by a tensor attached to each point, and cells that are intersections between Laguerre cells and ellipsoids.

Performance can be improved, using a GPU implementation of the clipped Laguerre diagram [61–63]. This will require to find a way of making the representation more compact (convex polytopes with 200 faces do not fit well in GPU caches). Direct representation of curved polytopes [49] may be an option.

Finally, it is important to stress the fact that the so-defined free surface Lagrangian mesh *smoothly* depends on the parameters $\mathbf{x}_i$. Namely, the function K is differentiable up to the second order in both ψ and $\mathbf{x}$, and the second-order derivatives have simple expressions (see [41]). As a consequence, this representation can be used to optimize an objective function that depends on a shape of prescribed volume, where the shape is parameterized by the $\mathbf{x}_i$'s. Then, the objective function will *smoothly* depend on the $\mathbf{x}_i$'s, making it possible to optimize it with efficient numerical methods (Newton, LBFGS, ...).

In future works, I plan to explore different applications, in astrophysics (extensions of [4]), and also in shape and topology optimization that may benefit from the differentiability of the representation. In both cases the motion is derived from a variational principle (action minimization and minimization of compliance respectively). Using the differentiable parameterization of the free-surface volume introduced here, I think that it will be possible to deduce the time evolution of the

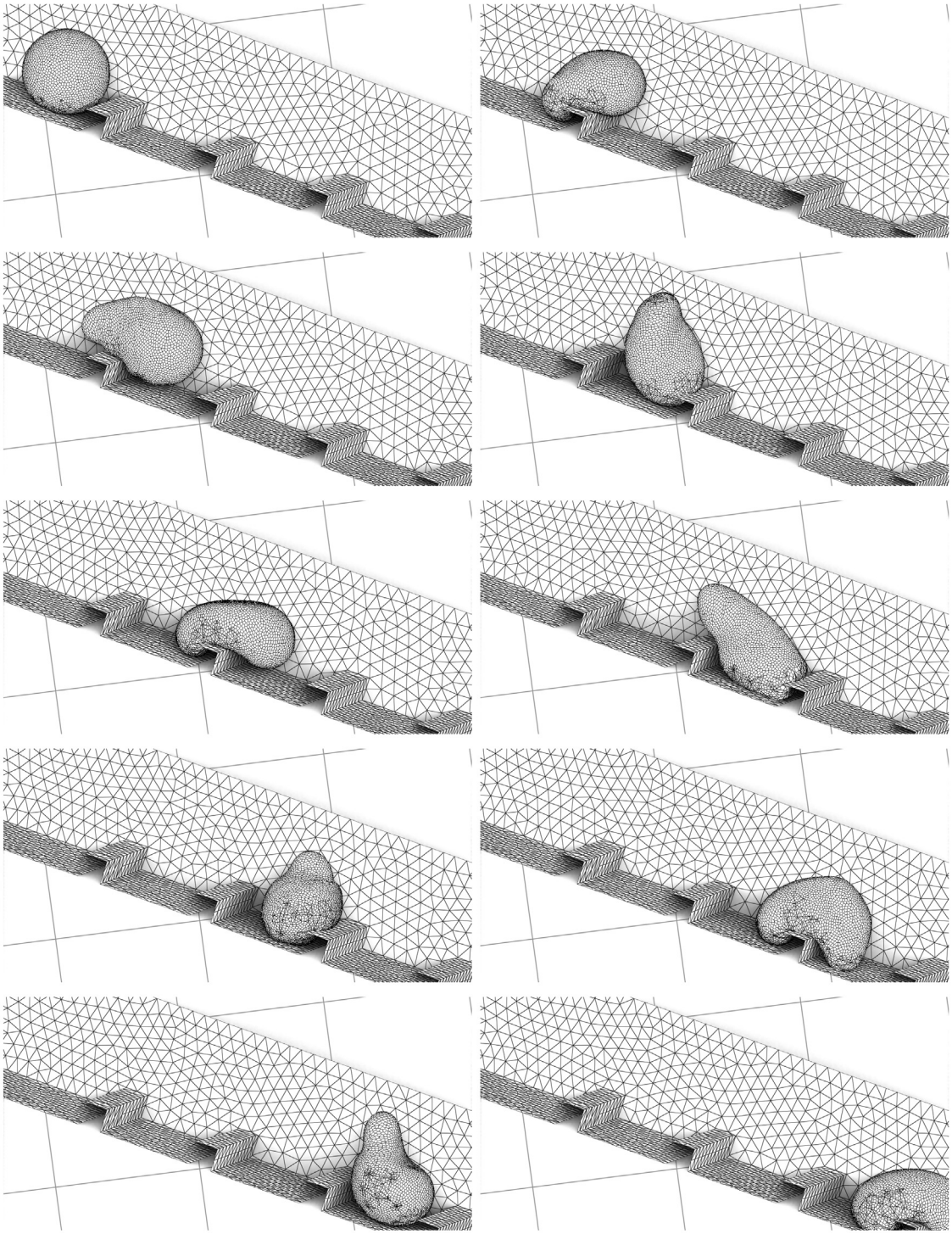

Fig. 19. Simulated droplet hurdles race.

parameters from the variational principle, by applying some forms of the chain rule and the adjoint method. Exploiting the structure of the so-discretized variational problem (symplectic geometry), enforcing conservation laws may be easier. It is also interesting to notice that the current trend of numerical methods called "artificial intelligence" mostly relies on representations that are differentiable with respect to the parameters. I think the representation described here can be used to represent latent spaces of shapes, and efficiently fit them to databases of 3D objects.

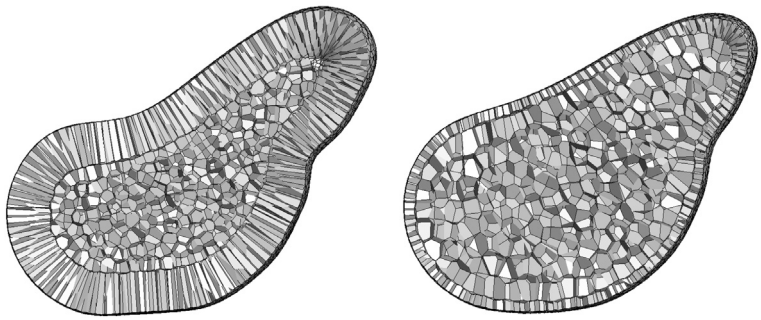

Fig. 20. Left: since surface tension gathers points on the boundary, if cell volumes are all the same, then cells adjacent to the boundary are elongated, creating artificial surface tension. Right: this effect can be mitigated by assigning smaller volumes to boundary cells.

CRediT authorship contribution statement

Bruno Lévy: Conceptualization, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

I wish to thank Quentin Mérigot, Yann Brenier and Jean-David Benamou for many discussions, Hélène de Maleprade for sharing the photo of her amazing “droplet hurdles race” experiment (with Rachid Bendimerad, Christophe Clanet and David Quéré), and Kevin Mattheus Moerman for indicating me Hélène’s work on Twitter (and challenging me to simulate it!), David Lopez for his help with the Enright and Zalesak tests, and the anonymous reviewers for their suggestions that helped improving this work.

References

[1] P. Frey, D. Kazerani, T. Ta, An adaptive numerical scheme for solving incompressible two-phase and free-surface flows, Int. J. Numer. Methods Fluids 2018 (2018).
[2] Y. Brenier, U. Frisch, M. Hénon, G. Loeper, S. Matarrese, R. Mohayaee, A. Sobolevskii, Reconstruction of the early universe as a convex optimization problem, Mon. Not. R. Astron. Soc. 346 (501) (2003).
[3] U. Frisch, S. Matarrese, R. Mohayaee, A. Sobolevskii, A reconstruction of the initial conditions of the universe by optimal mass transportation, Nature 417 (2002) 260.
[4] B. Lévy, R. Mohayaee, S. von Hausegger, A fast semi-discrete optimal transport algorithm for a unique reconstruction of the early Universe, Working paper or preprint, Dec. 2020, https://hal.archives-ouvertes.fr/hal-03081581.
[5] P. Geoffroy-Donders, G. Allaire, O. Pantz, 3-d topology optimization of modulated and oriented periodic microstructures by the homogenization method, J. Comput. Phys. 401 (2020) 108994, https://hal.archives-ouvertes.fr/hal-01939201.
[6] G. Allaire, Shape Optimization by the Homogenization Method, Applied Mathematical Sciences, vol. 146, Springer, New York, 2002.
[7] Q. Mérigot, J. Meyron, B. Thibert, An algorithm for optimal transport between a simplex soup and a point cloud, CoRR, arXiv:1707.01337 [abs], 2017.
[8] E. Maitre, Review of numerical methods for free interfaces, in: Ecole Thématique “Modèles de champ de phase pour l’évolution de structures complexes”, Les Houches, France, 2006, pp. 1–28, https://hal.archives-ouvertes.fr/hal-00104028.
[9] F. de Goes, C. Wallez, J. Huang, D. Pavlov, M. Desbrun, Power particles: an incompressible fluid solver based on power diagrams, ACM Trans. Graph. 34 (4) (2015) 50:1–50:11, https://doi.org/10.1145/2766901.
[10] J. Donea, S. Giuliani, J. Halleux, An arbitrary Lagrangian-Eulerian finite element method for transient dynamic fluid-structure interactions, Comput. Methods Appl. Mech. Eng. 33 (1) (1982) 689–723, https://doi.org/10.1016/0045-7825(82)90128-1, https://www.sciencedirect.com/science/article/pii/0045782582901281.
[11] B. Maury, O. Pirroneau, Characteristics ale method for the unsteady 3d Navier-Stokes equations with a free surface, Int. J. Comput. Fluid Dyn. 6 (1996) 175–188, https://doi.org/10.1080/10618569608940780.
[12] J.-F. Gerbeau, M. Vidrascu, P. Frey, Fluid–structure interaction in blood flows on geometries based on medical imaging, Comput. Struct. 83 (2) (2005) 155–165, https://doi.org/10.1016/j.compstruc.2004.03.083, Advances in Analysis of Fluid Structure Interaction, https://www.sciencedirect.com/science/article/pii/S0045794904003001.
[13] J.M. Hyman, Numerical methods for tracking interfaces, Phys. D: Nonlinear Phenom. 12 (1) (1984) 396–407, https://doi.org/10.1016/0167-2789(84)90544-X, https://www.sciencedirect.com/science/article/pii/016727898490544X.
[14] J. Du, B. Fix, J. Glimm, X. Jia, X. Li, Y. Li, L. Wu, A simple package for front tracking, J. Comput. Phys. 213 (2006) 613–628.
[15] Z. Tukovic, H. Jasak, A moving mesh finite volume interface tracking method for surface tension dominated interfacial fluid flow, Comput. Fluids 55 (2012) 70–84.
[16] Q. Zhang, A.L. Fogelson, MARS: an analytic framework of interface tracking via mapping and adjusting regular semialgebraic sets, SIAM J. Numer. Anal. 54 (2) (2016) 530–560, https://doi.org/10.1137/140966812.

[17] J. Glimm, J. Grove, B. Lindquist, O. McBryan, G. Tryggvason, The bifurcation of tracked scalar waves, SIAM J. Sci. Stat. Comput. 9 (1988), https://doi.org/10.1137/0909006.
[18] T. Brochu, R. Bridson, Robust topological operations for dynamic explicit surfaces, SIAM J. Sci. Comput. 31 (2009) 2472–2493, https://doi.org/10.1137/080737617.
[19] D.J. Torres, J.U. Brackbill, The point-set method: front-tracking without connectivity, J. Comput. Phys. 165 (2000) 620–644.
[20] S. Shin, S. Abdel-Khalik, V. Daru, D. Juric, Accurate representation of surface tension using the level contour reconstruction method, J. Comput. Phys. 203 (2005) 493–516, https://doi.org/10.1016/j.jcp.2004.09.003.
[21] F.H. Harlow, J.E. Welch, Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface, Phys. Fluids 8 (12) (1965) 2182–2189, https://doi.org/10.1063/1.1761178, https://aip.scitation.org/doi/pdf/10.1063/1.1761178, https://aip.scitation.org/doi/abs/10.1063/1.1761178.
[22] C.S. Peskin, The immersed boundary method, Acta Numer. 11 (2002) 479–517, https://doi.org/10.1017/S0962492902000077.
[23] Y. Chang, T. Hou, B. Merriman, S. Osher, A level set formulation of Eulerian interface capturing methods for incompressible fluid flows, J. Comput. Phys. 124 (2) (1996) 449–464.
[24] M. Theillard, A volume-preserving reference map method for the level set representation, J. Comput. Phys. 442 (2021), https://doi.org/10.1016/j.jcp.2021.110478.
[25] C. Hirt, B. Nichols, Volume of fluid (vof) method for the dynamics of free boundaries, J. Comput. Phys. 39 (1) (1981) 201–225, https://doi.org/10.1016/0021-9991(81)90145-5, https://www.sciencedirect.com/science/article/pii/0021999181901455.
[26] M. Sussman, E.G. Puckett, A coupled level set and volume-of-fluid method for computing 3d and axisymmetric incompressible two-phase flows, J. Comput. Phys. 162 (2000) 301–337.
[27] D. Enright, R. Fedkiw, J. Ferziger, I. Mitchell, A hybrid particle level set method for improved interface capturing, J. Comput. Phys. 183 (2002) 83–116, https://doi.org/10.1006/jcph.2002.7166.
[28] D. Enright, F. Losasso, R. Fedkiw, A fast and accurate semi-Lagrangian particle level set method, Comput. Struct. 83 (2004) 479–490.
[29] D. Jacqmin, Calculation of two-phase Navier–Stokes flows using phase-field modeling, J. Comput. Phys. 155 (1) (1999) 96–127, https://doi.org/10.1006/jcph.1999.6332, https://www.sciencedirect.com/science/article/pii/S0021999199963325.
[30] Q. Merigot, B. Thibert, Optimal transport: discretization and algorithms, arXiv:2003.00855, 2020.
[31] B. Lévy, E.L. Schwindt, Notions of optimal transport theory and how to implement them on a computer, Comput. Graph. 72 (2018) 135–148, https://doi.org/10.1016/j.cag.2018.01.009.
[32] G. Peyré, M. Cuturi, Computational optimal transport, Preprint, https://optimaltransport.github.io/, 2017.
[33] F. Santambrogio, Optimal transport for applied mathematicians, in: Calculus of Variations, PDEs, and Modeling, in: Progress in Nonlinear Differential Equations and Their Applications, vol. 87, Birkhäuser/Springer, Cham, 2015.
[34] C. Villani, Optimal Transport: Old and New, Grundlehren der Mathematischen Wissenschaften, Springer, Berlin, 2009, http://opac.inria.fr/record=b1129524.
[35] C. Villani, Topics in Optimal Transportation, Graduate Studies in Mathematics, American Mathematical Society, Providence, R.I., 2003, http://opac.inria.fr/record=b1122739.
[36] Y. Brenier, Polar factorization and monotone rearrangement of vector-valued functions, Commun. Pure Appl. Math. 44 (1991) 375–417.
[37] F. Aurenhammer, Power diagrams: properties, algorithms and applications, SIAM J. Comput. 16 (1) (1987) 78–96.
[38] F. Aurenhammer, F. Hoffmann, B. Aronov, Minkowski-type theorems and least-squares partitioning, in: Symposium on Computational Geometry, 1992, pp. 350–357.
[39] B. Lévy, A numerical algorithm for L_2 semi-discrete optimal transport in 3d, ESAIM: M2AN (2015).
[40] J. Kitagawa, Q. Mérigot, B. Thibert, A Newton algorithm for semi-discrete optimal transport, CoRR, arXiv:1603.05579 [abs], 2016.
[41] F. De Gournay, J. Kahn, L. Lebrat, Differentiation and regularity of semi-discrete optimal transport with respect to the parameters of the discrete measure, Numer. Math. 141 (2) (2019) 429–453, https://hal.archives-ouvertes.fr/hal-01721681.
[42] M.R. Hestenes, E. Stiefel, Methods of conjugate gradients for solving linear systems, J. Res. NIST 49 (6) (1952) 409–436, https://doi.org/10.6028/jres.049.044.
[43] A. Bowyer, Computing Dirichlet tessellations, Comput. J. 24 (2) (1981) 162–166, https://doi.org/10.1093/comjnl/24.2.162.
[44] D. Watson, Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes, Comput. J. 24 (2) (1981) 167–172.
[45] Inria, project ALICE-PIXEL, Geogram: a programming library of geometric algorithms, http://alice.loria.fr/software/geogram/doc/html/index.html, 2014–2021.
[46] J. Boissonnat, O. Devillers, S. Pion, M. Teillaud, M. Yvinec, Triangulations in CGAL, Comput. Geom. 22 (1–3) (2002) 5–19, https://doi.org/10.1016/S0925-7721(01)00054-2.
[47] J.R. Shewchuk, Robust adaptive floating-point geometric predicates, in: Proceedings of the Twelfth Annual Symposium on Computational Geometry, Philadelphia, PA, USA, May 24–26, 1996, 1996, pp. 141–150, http://doi.acm.org/10.1145/237218.237337.
[48] A. Meyer, S. Pion, FPG: a code generator for fast and certified geometric predicates, in: Real Numbers and Computers, Santiago de Compostela, Spain, 2008, pp. 47–60, https://hal.inria.fr/inria-00344297.
[49] H. Leclerc, Q. Mérigot, F. Santambrogio, F. Stra, Lagrangian discretization of crowd motion and linear diffusion, SIAM J. Numer. Anal. 58 (4) (2020) 2093–2118, https://doi.org/10.1137/19M1274201.
[50] H. Edelsbrunner, E.P. Mücke, Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms, ACM Trans. Graph. 9 (1) (1990) 66–104, https://doi.org/10.1145/77635.77639, http://doi.acm.org/10.1145/77635.77639.
[51] B. Lévy, Robustness and efficiency of geometric programs: the predicate construction kit (PCK), Comput. Aided Des. 72 (2016) 3–12, https://doi.org/10.1016/j.cad.2015.10.004.
[52] P. Terdiman, Opcode 3d collision detection library, http://www.codercorner.com/Opcode.htm, 2005.
[53] S.T. Zalesak, Fully multidimensional flux-corrected transport algorithms for fluids, J. Comput. Phys. 31 (6) (1979), https://doi.org/10.1016/0021-9991(79)90051-2, https://www.osti.gov/biblio/5312964.
[54] B. Lévy, N. Bonneel, Variational anisotropic surface meshing with Voronoi parallel linear enumeration, in: X. Jiao, J. Weill (Eds.), Proceedings of the 21st International Meshing Roundtable, IMR 2012, October 7-10, 2012, Springer, San Jose, CA, USA, 2012, pp. 349–366.
[55] S. Xin, B. Lévy, Z. Chen, L. Chu, Y. Yu, C. Tu, W. Wang, Centroidal power diagrams with capacity constraints: computation, applications, and extension, ACM Trans. Graph. 35 (6) (2016) 244:1–244:12, https://doi.org/10.1145/2980179.2982428.
[56] T.O. Gallouët, Q. Mérigot, A lagrangian scheme à la Brenier for the incompressible euler equations, Found. Comput. Math. (May 2017), https://doi.org/10.1007/s10208-017-9355-y.
[57] J. Benamou, Y. Brenier, A computational fluid mechanics solution to the Monge-Kantorovich mass transfer problem, Numer. Math. 84 (3) (2000) 375–393, https://doi.org/10.1007/s002110050002.
[58] T. Tao, The Euler-Arnold equation, http://terrytao.wordpress.com/2010/06/07/the-euler-arnold-equation/, 2010.
[59] R. Abraham, J. Marsden, Foundations of Mechanics, Behamin-Cummings, London, 1978.
[60] H. de Maleprade, R. Bendimerad, C. Clanet, D. Quéré, Droplet hurdles race, Appl. Phys. Lett. 118 (2021) 171601, https://doi.org/10.1063/5.0043908.
[61] D. Yan, W. Wang, B. Lévy, Y. Liu, Efficient computation of clipped Voronoi diagram for mesh generation, Comput. Aided Des. 45 (4) (2013) 843–852, https://doi.org/10.1016/j.cad.2011.09.004.

[62] N. Ray, D. Sokolov, S. Lefebvre, B. Lévy, Meshless Voronoi on the GPU, ACM Trans. Graph. 37 (6) (2018) 265:1–265:12, https://doi.org/10.1145/3272127.3275092.
[63] J. Basselin, L. Alonso, N. Ray, D. Sokolov, S. Lefebvre, B. Lévy, Restricted power diagrams on the GPU, in: Eurographics'2021, Vienne, Austria, 2021, pp. 1–12, https://hal.archives-ouvertes.fr/hal-03169575.