Computer Vision and Image Understanding 112 (2008) 274–285

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu

Fast PDE approach to surface reconstruction from large cloud of points

Marco Marcon *, Luca Piccarreta, Augusto Sarti, Stefano Tubaro

Dipartimento di Elettronica e Informazione—Politecnico di Milano, Piazza Leonardo Da Vinci, 32, 20133 Milano, Italy

ARTICLE INFO

Article history:
Received 4 June 2007
Accepted 13 May 2008
Available online 3 June 2008

Keywords:
Surface from points
3D reconstruction
Implicit surface
Partial differential equations
Level-set methods
Volume of fluid
3D point wrapping

ABSTRACT

In this article we propose an algorithm for fast reconstruction of 3D surfaces starting from large sets of unorganized sample points. The proposed algorithm is based on the temporal evolution of a volumetric implicit function. The evolving front can be thought as the surface that separates two different fluids obeying specific fluid dynamics laws. One remarkable feature of this approach is its ability to model complex topologies using a set of intuitive tools derived from fluid physics: Global and local surface descriptors are used allowing the parallelization of the algorithm on different processes each of one can operate on different sub-sets of the whole cloud with different resolutions and accuracies. Tests on large and complex clouds of 3D points show an high efficiency of the proposed approach: between one and two orders of magnitude faster than traditional implicit solutions.

1. Introduction

The problem of building surfaces from unorganized sets of 3D points has recently gained a great deal of attention. In fact, in addition to being an interesting problem of topology extraction from geometric information, its applications are becoming more and more numerous. For example, the acquisition of large numbers of 3D points is becoming easier and more affordable using, for example, 3D-scanners [1]. There are a number of other applications where objects are better described by their external surface rather than by unorganized data (clouds of points, data slices, etc.). For example, in medical applications based on CAT scans or NMRs it is often necessary to visualize some specific tissues such as the external surface of an organ starting from the acquired 3D points. This can be achieved by selecting the points that belong to a specific class (organ boundary, tissue, etc.) and then generating the surface from their interpolation. In most cases the definition of this surface is an ill-posed problem as there is no unique way to connect points of a dataset into a surface, therefore it is often necessary to introduce constraints for globally or locally controlling the surface behavior. As a matter of fact, the resulting surface often turns out to exhibit a complex topology due to noise in the acquired data or ambiguities in the case of non-convex objects [2]. In order to overcome such problems, surface wrapping algorithms need to incorporate specific constraints on the quality of the data-fitting (surface closeness to the acquired points), on the maximum surface curvature and roughness, on the number of resulting triangles, etc.

The existing surface reconstruction methods can be classified into two broad categories: the former describes the surface as an implicit function while the latter describes it in an explicit form. Explicit (boundary) representations describe the surface in terms of point connections, and traditional approaches are based on Delaunay triangulation and Voronoi diagrams [3,4]. Another well-known explicit approach is a parametric surface description based on NURBS [5,6]. One example of surface-oriented solution, proposed in [7,2] is based on the computation of the signed Euclidean distance between each sample point and a linearly regressed plane that approximates the local tangent plane. Curless and Levoy [8] developed an explicit algorithm tuned for laser range data, which is able to guarantee a good rejection of outliers (points whose coordinates were not correctly acquired). Another well-known approach is the α-shape [9,10], which associates a polyhedral shape to an unorganized set of points through a parameterized construction. Bajaj et al. [11] recently used the α-shape approach as a first step in a complete reconstruction pipeline. Finally, algorithms based on "Delaunay sculpting" were used for progressively eliminating tetrahedra from the Delaunay triangulation, based on their "circumspheres" (see, for example, Boissonnat [12–14]) or other approaches are based on the Medial Axis mesh reduction [15]. The explicit approach is usually very fast but it often requires a surface subdivision into simpler sub-surfaces in order to correctly model complex surfaces and avoid intersections of multiple triangles. Implicit representations, on the other hand, define the surface as a constraint in 3D space (volumetric representation), which enables the description of complex surfaces through a simple

* Corresponding author. Fax: +39 0223999611.
E-mail address: marcon@elet.polimi.it (M. Marcon).

1077-3142/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.cviu.2008.05.008

volumetric description. This approach to surface modeling also simplifies the 3D point location noise management; the merging of different object portions irrespective of resolutions; and the interpolation of poorly sampled regions.

A classical approach to implicit surface representation [16] is based on a combination of smooth basis functions (primitives), such as blobs, to find a scalar function where all data points are close to one of its iso-contours, further approaches based on anisotropic basic functions can be found in [17]. The basis functions are coupled together and the change of a single data point could affect the coefficients associated to all basis functions. This makes the incremental updates, the deformation, and the interaction with the surface quite difficult. Another approach uses the data set to define a signed distance function on a cubic grid and identifies the zero iso-contour (level-set) of the signed distance function as the reconstructed implicit surface [18]. When dealing with dynamic systems, level-set representations can easily cope with shape changes and track moving surfaces and their topological changes. Level-set techniques were simultaneously introduced by Osher and Sethian [19–21] and Zhao and Osher [48], and, with reference to different applications, by [22]. In their general form, such techniques require the definition of a volumetric function, which is updated at every time step until a certain amount of time has elapsed, or until the evolving front (level-set) reaches a stable configuration. In principle, the evolution of a volumetric function defined on a voxel-set of N voxels per side, would require an order of N^3 voxels to be updated for a number of iterations that is proportional to N (assuming that the front will pass through the whole voxel-set). This number of voxels to be updated can be reduced from an order of N^4 to an order of N^3 by limiting the volume of interest to a "narrow band" surrounding the evolving front. A further reduction to an order of $N^2\log(N)$ updates can be achieved using a multi-resolution approach [19,23]. Further application of implicit function to triangular meshes deformation using metaballs is made by Ilic and Fua [24]. Implicit representations, although computationally demanding and memory consuming, offer numerous advantages over explicit ones:

- the surface can be easily manipulated by acting directly on the implicit function;
- meaningful features such as the 3D skeleton of an object, can be readily extracted;
- different portions of the same object can be smoothly merged together by using simple logical operators on their implicit functions;
- multi-resolution implementations can be quite easily implemented;
- algorithms operating on the implicit function can be parallelized, therefore the effort for complex object reconstructions can be shared between different computers.

In the algorithm proposed by Zhao et al. [25], the basic surface reconstruction problem was approached in the analog domain by using differential geometry and Partial Differential Equations (PDEs). In particular, the level-set method applied to the implicit surface turned out to provide a powerful framework for the modeling, the analysis, and the deformation of surfaces. In their approach, the implicit function represents the signed distance (in the 3D space) from the surface, where the sign tells us whether the considered point is inside or outside the object, and the zero level-set represents the surface.

Our approach, that was introduced in [26,27], significantly differs from the above level-set method in the definition of the volumetric function, which we derived from the Volume of Fluid (VoF) description [28]. The VoF description is a popular interface tracking algorithm used in fluid dynamics. This method, developed over two decades ago by Youngs [29] from an original idea of Noh [30], has since then become a frequent choice in Eulerian[1] modeling of Interfacial flows, thanks to its robustness and effectiveness. As a matter of fact, this solution has proven particularly effective for those dynamics where interfaces are prone to undergoing topological changes such as merging, splitting, hole piercing/filling, etc. In our approach we adopt the "Piecewise Linear Interface Calculation" (PLIC) suggested by Gueyffier et al. [28] to track the evolving surface. In particular, we define the surface as the interface of separation between two incompressible fluids: one representing the internal volume of the object and one corresponding to the external volume. Our algorithm has the remarkable advantage of dramatically reducing the computational cost with respect to traditional level-set approaches [25] and enabling a more intuitive and physically plausible parameter tuning. The time-evolution of our approach relies on the Navier–Stokes (N–S) PDEs, which offer the most general description of a fluid flow, unlike the classical level-set evolutive paradigm, which is based on the Hamilton–Jacobi (H–J) PDE. It is important to emphasize that the VoF approach allows us to effectively deal with many phenomena and problems that often occur in data acquisition, such as acquisition noise on the 3D data points, non-uniform data density, surfaces of complex topology or shapes that are normally difficult to deal with (e.g. sharp blades, steep grooves, creases, etc.).

2. Level-sets and volume of fluid

The level-set approach is one of the most widespread methods for representation and evolution of implicit surfaces (see, for example, [31]). In this section we briefly describe how this approach can be applied to the point-cloud wrapping problem with specific reference to the work of Osher and Zhao [25,32].

We define $\mathscr{S}$ as a general data set containing the 3D coordinates of the points. Let $d(\mathbf{x}) = dist(\mathbf{x}, \mathscr{S})$ be the function that describes the distance between a generic 3D point $\mathbf{x}$ and the nearest point in $\mathscr{S}$. In order to measure how well a given surface Γ fits the data set $\mathscr{S}$, we can define the energy functional

$$E(\Gamma) = \left[\int_{\Gamma} d^p(\mathbf{x})\mathrm{d}s\right]^{\frac{1}{p}}, \quad 1 \leqslant p \leqslant \infty, \tag{1}$$

which is computed over the set of all surface points, and ds is the infinitesimal surface area. The energy functional (1) is independent of the adopted surface description and is invariant under rotation and translation. For $p = 2$ it measures the Root Mean Square (RMS) distance. As p tends to infinity, $E(\Gamma)$ tends to measure the maximum distance. Intermediate values of p, indeed, exhibit an intermediate behavior by attributing more or less importance to distance peaks.

As described in [32], looking for the surface that minimizes the functional (1) is similar to enveloping the data set with a membrane with certain elastic properties and have this membrane to evolve in time until it comes to rest. This time evolution paradigm can be described as

$$\frac{\partial \Gamma}{\partial t} = -\left[\int_{\Gamma} d^p(\mathbf{x})ds\right]^{\frac{1}{p}-1} d^{p-1}(\mathbf{x})\left[\nabla d(\mathbf{x})\cdot\mathbf{n} + \frac{1}{p}d(\mathbf{x})\kappa\right]\mathbf{n}, \tag{2}$$

and its minimum can be found by solving the Euler–Lagrange equation

[1] In the Eulerian approach the dispersed particles are treated as a continuum and turbulent dispersion is described by Fick's diffusion equation, similarly to molecular diffusion. Opposite to this approach is the Lagrangian one, which defines the trajectory of a single particle by solving the particle motion equations assuming a known turbulent fluid velocity field. Particle concentration is estimated from the statistics of a large number of trajectories.

$$d^{p-1}(\mathbf{x})\left[\nabla d(\mathbf{x})\cdot\mathbf{n}+\frac{1}{p}d(\mathbf{x})\kappa\right]=0, \tag{3}$$

where $\mathbf{n}$ is the unit normal (pointing outward) and κ is the mean curvature. The term $\nabla d(\mathbf{x})\cdot\mathbf{n}$ in Eq. (3) describes the surface "attraction" while $d(\mathbf{x})\kappa$ accounts for surface tension. The scalar function $d(\mathbf{x})$ allows the surface to be more flexible in regions where the data set is denser and to be more rigid where the sampling density is low.

The evolution equation (2) involves the surface's mean curvature and is a nonlinear parabolic equation. A stable time-explicit scheme for solving this equation requires a restrictive time-step size $\Delta t=\mathcal{O}(h^2)$ where h is the spatial grid step-size.

2.1. Level-set evolution

Zhao [25,32] proposed an evolutive model based on the physical phenomenon of convection: the convection of a flexible surface Γ in a velocity field $\mathbf{v}(\mathbf{x})$ is described by the differential equation

$$\frac{\partial\Gamma(t)}{\partial t}=\mathbf{v}(\Gamma(t)). \tag{4}$$

The distance function $d(\mathbf{x})$ to the data set $\mathscr{S}$ represents the potential field for the convection model and the velocity field $\mathbf{v}$ is oriented towards the minima of the potential field $\mathbf{v}=-\nabla d(\mathbf{x})$. This leads to the convection equation

$$\frac{\partial\Gamma(t)}{\partial t}=-\nabla d(\mathbf{x}). \tag{5}$$

Given a data set $\mathscr{S}$, the evolving surface can be thought of as made of a set of particles that move within the field described by the distance functional. Each particle is thus attracted to the closest point in $\mathscr{S}$ unless it is equally distant from two or more data points. These special locations in space correspond to the boundaries of the Voronoi volumes of the data points. It is important to notice, however, that Voronoi boundaries always have a zero volumetric measure. In fact, in the two-dimensional case, the Voronoi boundary is a piecewise linear curve (each segment of the curve lies on the axis of the segment that joins two data points). In the three-dimensional case the boundaries of the Voronoi regions are represented by a piecewise planar surface (each planar patch lies on the axial plane of the segment that joins two data points). We will later show that, since no volume is ever occupied by such Voronoi boundaries, such sets of points do not interfere with surface evolution towards the rest (final) condition. In Zhao's approach [25] the ambiguity at equi-distant points is avoided by adding a small surface force. This tension, however, is automatically added by finite difference schemes in the form of "numerical viscosity". As a result, the equi-distant points on the curve or surface get "dragged" by the neighboring surface points so that the whole curve or surface will globally evolve towards the data set until it comes to rest.

2.2. Volumetric functions in level-set methods

The level set method is based on a continuous PDEs that describes the temporal evolution of a volumetric function. As the surface corresponds to the zero levelset of this function, a correct steering of the evolving surface can only be achieved through a proper definition and control of the volumetric function, of its dynamics (PDE), and its initial condition.

The front evolution starts from an initial (closed) 3D surface $\Gamma(t)|_{t=0}$, and we assume that the final surface configuration is enclosed in this initial surface. We need to control the front evolution in such a way that, as time goes by, the surface will converge towards the final solution. One way to do so is to force the local front to move according to a local velocity vector $\mathbf{v}$, which is assumed to be locally normal to the surface and whose magnitude is linked to the local curvature for reasons that will be clarified later. Surface steering is achieved by including in the definition of the velocity vector other terms that account, for example, for data fitting or other local surface measurements.

Let $\phi(\mathbf{x},t)$ be the scalar (volumetric) function whose zero level set represents the evolving surface $\Gamma(t)$. In order to define this function, we start with specifying its initial condition as

$$\phi(\mathbf{x},t)|_{t=0}=D, \tag{6}$$

where D is the 'signed' distance between $\mathbf{x}$ and $.\Gamma(t)|_{t=0}$, which is positive (negative) when the point $\mathbf{x}$ is outside (inside) the surface. This way we have a volumetric function $\phi:\mathbb{R}^3\times\mathbb{R}^+\mapsto\mathbb{R}$ that evolves in time starting from a reasonable initial condition (signed Euclidean distance from the Γ surface).

The evolution equations of the individual level surfaces define a corresponding evolution of the scalar function $\phi(\mathbf{x},t)$ where $\mathbf{x}\in\mathbb{R}^3$. The level set of magnitude k at time t (i.e. the surface of distance k from the surface at time t) is represented by

$$\Gamma_k=\{\mathbf{x}\mid\phi(\mathbf{x},t)=k\}. \tag{7}$$

By definition the surface Γ_k has constant level over time, therefore its time derivative is zero

$$\frac{\partial\phi(\Gamma_k,t)}{\partial t}+\nabla\phi(\Gamma_k,t)\cdot\frac{\partial\Gamma_k}{\partial t}=0, \tag{8}$$

therefore

$$\frac{\partial\phi}{\partial t}=-\nabla\phi\cdot\frac{\partial\Gamma_k}{\partial t}=|\nabla\phi|\frac{\partial\Gamma_k}{\partial t}\cdot\mathscr{N}, \tag{9}$$

where $\mathscr{N}=-\nabla\phi/|\nabla\phi|$ is the surface normal.

Using Eq. (5) the final evolutive equation for the level set is:

$$\frac{\partial\phi}{\partial t}=\nabla d(\mathbf{x})\cdot\nabla\phi. \tag{10}$$

This implementation suffers from inherent inaccuracy in its numerical scheme and, even if the starting implicit function $\phi(t)|_{t=0}$ is a signed distance, its time evolution will generally divert from this condition. In fact, all level sets in the convection model are simultaneously attracted to the data set, therefore they tend to become more and more densely packed together. Different procedures were proposed to avoid this source of numerical inaccuracy. One popular solution consists of the re-initialization of the volumetric function [33,34], which is used for locally re-distancing the level sets without affecting the motion of the zero level set. An interesting alternative was proposed by Gomes and Faugeras [35], who defined a different implementation of the Hamilton-Jacobi PDE that inherently preserves the distance function.

2.3. Volumetric functions in VoF methods

Our volumetric function closely resembles that of the VoF method, originally introduced by Noh and Woodward [30] to model the interface evolution between different materials. Since its first appearance, the VoF method has been formulated in a variety of forms and re-introduced under different names such as the "cell method" and the "partial fractions method". The key idea behind this technique is to define a fixed computational grid and assign to each grid cell a value that describes the relative proportions of two materials contained in that cell. Our particular modeling metaphor is based on two immiscible fluids of opposite densities[2] $\rho=1$ and $\rho=-1$, which identify the surface's outside and inside, respectively. More specifically, cells completely filled with outer fluid take on the value +1 while cells that are only filled with inner fluid will

[2] This is a conventional choice that allows us to set the levelset of interest to zero.

take on the value −1. When filled with a mix of both fluids, the cell is assigned an intermediate value in the]−1,+1[range, which measures the density of the mix. An example of the described VoF description is shown in Fig. 1. Notice that the above metaphor is quite common in computational physics for applications of interface tracking [36–39]. In this section will show how to use it for the problem of three-dimensional surface reconstruction from a sparse point-set.

As a first step we define a bounding box that completely encloses the point cloud. We assume that this box is completely filled with inner fluid and that the whole space outside of the box is filled with outer fluid. We define the rules of evolution so that both fluids are set in motion by the presence of data points, which act as attractors (see Fig. 2). This fluid migration takes place in compliance with laws of conservation. According to such rules, the front starts propagating inward (towards the cloud of points) and stops only when the inner fluid is confined "inside" the cloud of points and the interface between the two fluids wraps the point-cloud.

From an implementative standpoint, we need to define a voxel-set whose voxels describe the local fluid density. As voxel-set can only have a finite number of voxels, we cannot model the whole outer space. Knowing, however, that the front propagates inward, we can transparently replace the infinite space outside of the bounding box with just an additional layer of voxels whose value is permanently set to 1. Roughly speaking, this situation corresponds to replacing the outer bulk of fluid with a closed membrane that "essudes" outer fluid in sufficient quantity to compensate possible density changes caused by the propagating front. The situation is similar to replacing an electrical line of infinite length with a resistor that matches the line's characteristic impedance.

Setting the value of the outer layer of cells to 1 means preventing the outer fluid from thinning and keeping its density constant. Indeed, we need to guarantee a similar behavior for the inner fluid. We do not want, for example, the fluid to pile up inside the object because it would antagonize the shrinking of the interface between the two fluids with an action that depends on the initial condition of the fluids (on the size of the bounding box). In order to avoid this problem, it would seem reasonable to place one or more drains somewhere within the object, which would act as valves for keeping the inner fluid's density constant. Drains of this sort correspond to inner cells whose value is permanently set to −1. Although implementing such drains is quite straightforward, this is an undesirable burden to endure, as it needs to define the "inside" of the point-cloud, which may not be a trivial topological problem to solve. A simpler solution consists of limiting the volumetric function through a simple clipping operator that guarantees that $|\phi(\mathbf{x},t)| \leqslant 1$. As the clipping would never take place near the interface between the fluids, we could expect it not to interfere with its propagation. We will see, however, that this clipping turns out to be unnecessary as, in fact, our fluid conservation law is expressed in differential form, therefore its range of validity is only local. The implementation of the evolutionary equations will prevent fluids from piling up far from the moving front.

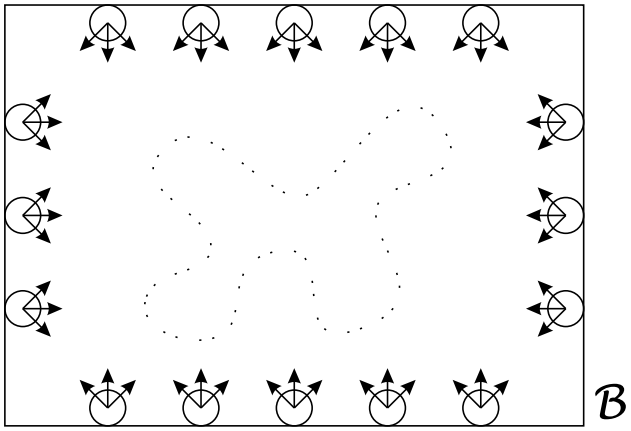
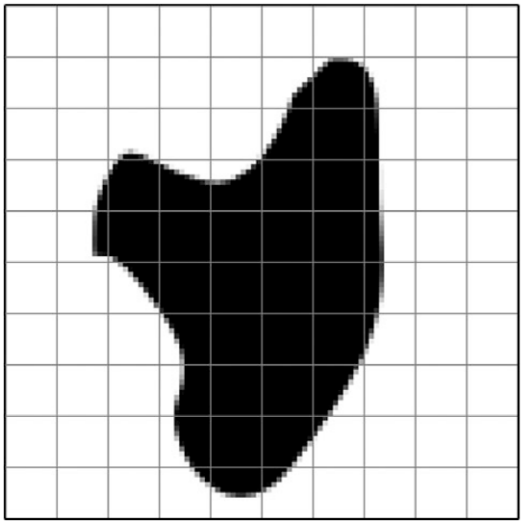


Fig. 2. The outer fluid, coming from the boundary cells, flows inward until it fills the space around the point cloud.

One aspect that needs to be carefully assessed is the existence of many different closed surfaces that could wrap the cloud of points in a plausible fashion [40]. Dealing with multiple solutions requires, as usual, the specification of additional constraints, which could be local or global. A number of issues could be considered for the definition of such constraints, which concern different conditions of acquisition, instruments precision and resolution. Exact surfaces that strictly honor the 3D data can be required in some applications but usually, due to data acquisition noise, smooth interpolating surfaces are preferable to avoid roughness. Furthermore, in some applications a highly structured surface could be required in regions where points are acquired with high density, while smooth interpolating surfaces could be advisable for poorly sampled and noisy regions. Traditional approaches [40,16] usually face these constraints by defining a global energy functional that



1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
1.00	1.00	1.00	1.00	1.00	0.64	-0.78	0.72	1.00	1.00
1.00	1.00	0.92	1.00	1.00	-0.42	-1.00	0.40	1.00	1.00
1.00	0.82	-0.92	-0.42	-0.40	-1.00	-1.00	0.40	1.00	1.00
1.00	0.48	-0.98	-1.00	-1.00	-1.00	-1.00	0.28	1.00	1.00
1.00	1.00	0.34	-1.00	-1.00	-1.00	-1.00	0.30	1.00	1.00
1.00	1.00	1.00	-0.36	-1.00	-1.00	-0.98	0.76	1.00	1.00
1.00	1.00	1.00	-0.18	-1.00	-1.00	-0.24	1.00	1.00	1.00
1.00	1.00	1.00	-0.20	-1.00	-0.78	0.86	1.00	1.00	1.00
1.00	1.00	1.00	0.88	-0.06	0.66	1.00	1.00	1.00	1.00

Fig. 1. (Left) Fixed grid applied to a closed surface. (Right) The Volume of Fluid description (each cell value describes the relative amount of the two fluids).

accounts for all such needs, which must then be minimized. For example, an energy term that is aimed at minimizing the surface roughness could be

$$E = \int_{\Omega} \left(\frac{\partial^2 s}{\partial x^2} + \frac{\partial^2 s}{\partial y^2} + \frac{\partial^2 s}{\partial z^2} + 2\frac{\partial^2 s}{\partial x \partial y} + 2\frac{\partial^2 s}{\partial y \partial z} + 2\frac{\partial^2 s}{\partial x \partial z} \right) \mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z, \quad (11)$$

which measures the global curvature, where Ω is the whole domain and s represents the surface. Another energy term that accounts for data fitting can be Eq. (1). In our approach, on the other hand, due to the specific differential formulation, most of the constraints can be locally defined to steer the surface evolution towards the final result. More details follow in the next section.

3. Front evolution

As already explained in the previous section, the front evolution follows a computational scheme that complies with a two-fluid quasi-physical paradigm based on the conservation of fluid mass in its differential (local) form. The volume is initially filled with inner fluid (all inner voxels are assigned the value -1). An outer layer of voxels is kept at the value +1, which means that such voxels are kept full of outer fluid. During the evolution both fluids are attracted towards the data points. As shown in Fig. 2, the outer fluid invades the space until the interface between the two fluids wraps the point cloud.

The outer fluid is prevented from flowing inside the point cloud because, once it closely wraps the data points, the inner fluid antagonizes further front propagations, and also because of a certain inherent viscosity in the outer fluid.

The solution that we propose is local, as all constraints and operators depend on parameters and features of local validity. The only global constraint that we use is in the stopping condition for the PDE evolution. Further details are given in Section 7

A first initialization step to rapidly find an initial guess could also be used, a possible approach could be based on a fast tagging algorithm, as suggested in [25], where external and interior grid points correspond to the exterior and interior fluid of the proposed approach, anyway, most of these approaches require structures (e.g. heap tables,...) updated at each step and sorting algorithms that could be time consuming.

Starting from an arbitrary initial condition, the surface evolution follows a time-dependent PDE. The 'time' parameterizes a family of implicit functions ϕ (whose zero level-set is the evolving surface) that converges towards the minimization of its energy. The PDE that we use is a simplified version of the Navier–Stokes equation for the mass conservation, as shown in [41].

The law of mass conservation is independent from the nature of the involved fluid or of the forces that act upon it. This law expresses the empirical fact that fluid mass can neither disappear nor be created except through sources or drains. This fact allows us to realistically model the motion of the fluid within the system. In our specific problem, we use this postulate to locally describe the behavior of the two fluids near the zero level-set. In its general form, the conservation law applied to a fluid confined in the volume Ω can be expressed in terms of the variations of the implicit function ρ; the fluxes that cross the volume boundary $\mathscr{B}$; and the fluid sources Q. The flux vector $\mathbf{G}$ represents the amount of fluid per second that crosses an infinitesimal surface element $\mathrm{d}\mathscr{B}$ in the perpendicular direction. This vector includes two components, a diffusive contribution $\mathbf{G}_D$ and a convective one $\mathbf{G}_C$. Generally speaking, the mass conservation law states that the rate of change $\frac{\partial}{\partial t}\int_{\Omega}\rho\,\mathrm{d}\Omega$ of the fluid density ρ within the volume Ω is equal to the net contribution from the sources and from the fluxes that cross the region boundary $\mathscr{B}$, i.e.

$$\frac{\partial}{\partial t}\int_{\Omega}\rho\,\mathrm{d}\Omega = \int_{\mathscr{B}}\mathbf{G}\cdot\mathrm{d}\mathscr{B} + \int_{\Omega}Q_V\,\mathrm{d}\Omega + \int_{\mathscr{B}}\mathbf{Q}_{\mathscr{B}}\cdot\mathrm{d}\mathscr{B}, \quad (12)$$

where the surface element vector $d\mathscr{B}$ points inward, Q_V represents the isotropic volume sources, while $\mathbf{Q}_{\mathscr{B}}$ are the oriented surface sources whose net contribution depends on the orientation of the surface element vector $d\mathscr{B}$. If we consider the Gauss theorem for continuous fluxes and surface sources, we obtain:

$$\frac{\partial}{\partial t}\int_{\Omega}\rho\,\mathrm{d}\Omega = \int_{\Omega}\nabla\cdot\mathbf{G}\cdot\mathrm{d}\Omega + \int_{\Omega}Q_V\,\mathrm{d}\Omega + \int_{\Omega}\nabla\cdot\mathbf{Q}_{\mathscr{B}}\cdot\mathrm{d}\Omega. \quad (13)$$

This expression is valid for an arbitrary volume Ω, therefore it also defines the conservation law in its differential form

$$\frac{\partial\rho}{\partial t} = \nabla\cdot\mathbf{G} + Q_V + \nabla\cdot\mathbf{Q}_{\mathscr{B}}. \quad (14)$$

In the absence of volume sources these temporal changes of ρ depend only on the flux contribution through the surface $\mathscr{B}$ and not on the flux values inside the volume Ω. A more detailed description can be given if we split the flux vector into its two convective ($\mathbf{G}_C$) and diffusive ($\mathbf{G}_D$) components. Given a density ρ in a flow of velocity $\mathbf{v}$, the convective term $\mathbf{G}_C$ is given by $\mathbf{G}_C = \mathbf{v}\rho$, which corresponds to the amount of fluid per volume unit that is transported by that motion. The diffusive flow, on the other hand, is defined as the density changes that take place in fluids at rest due to thermal motion. This term is usually proportional to the gradient of ρ, i.e. $\mathbf{G}_D = \xi\nabla\rho$ where ξ is the diffusivity constant. The conservation law can then be written as

$$\frac{\partial\rho}{\partial t} = \nabla\cdot(\mathbf{v}\rho) + \nabla\cdot(\xi\nabla\rho) + Q_V + \nabla\cdot\mathbf{Q}_{\mathscr{B}} \quad (15)$$

In order to solve this equation on a discrete grid (voxel-set) we use the implicit function ϕ that describes the relative proportions of the two fluids that occupy each cell volume, as proposed in the VoF method. In our case, the definition of this function can be modified to accommodate the fluid density function ρ: for further details see [28].

4. Velocity field

One major difference between our method and the level-set approach is in the fact that, instead of steering the evolution of the implicit function ϕ using a velocity field that corresponds to the distance gradient (see Eq. (5)), we use directly the distance vector field. With this choice of velocity field, a point will be assigned a velocity that is proportional to the vector $\mathbf{v}$ that joins that point to the closest data point. This way data points act as attractors for both fluids and, therefore, for the evolving front.

Even if this attraction field has no physical equivalent, it is physically consistent thanks to its irrotational characteristic (as we will see later). This fact guarantees algorithmic convergence. The motion of both fluids can be easily handled by specifying the evolution of the implicit function ϕ as shown in Section 5 since it indicates their relative presence in each voxel. In Fig. 3 we show an example of a two-dimensional non-convex cloud of points; the velocity vector in each point is shown in Fig. 4 along with the iso-velocity contours.

As the velocity is oriented towards the nearest data point, the velocity field depends only on a specific data point inside the Voronoi region of the point itself. In order to illustrate this concept more clearly, the velocity field inside the regions A and B of Fig. 4 is shown in Figs. 5 and 6.

As already said before, the magnitude of the velocity vector is proportional to the distance vector. This causes the interface between the two fluids to slow down as it approaches the cloud of points, and guarantees a gentle convergence towards the desired

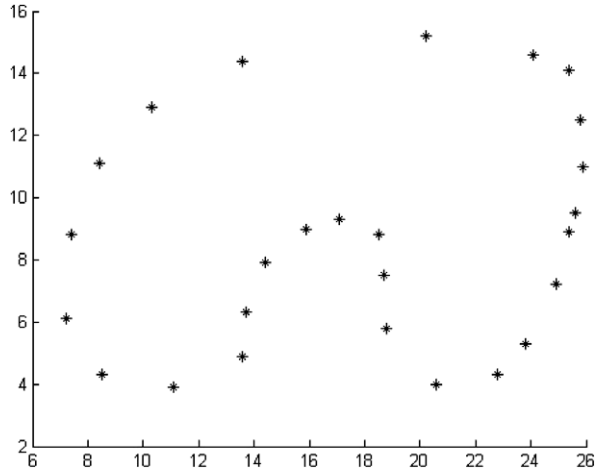


Fig. 3. An example of a two-dimensional non-convex cloud of points.

surface. In what follows we will show how the viscosity acts on the final wrapping result.

If we look further into the 2D example, we notice (see Fig. 6) that in the proximity of the line that connects two data points the velocity field is almost parallel to the line itself because

- the two fluids keep being separated when, after the evolution, a stable solution is reached (steady state);
- the outer fluid "does not penetrate" the cloud of points
- the interface between the two fluids turns out to be a "good surface" for wrapping the 3D data points

Notice that, due to the discretization of the velocity field, the outer fluid can find its way between two data points. In this case the vector field in that region would push the outer fluid back until an equilibrium between the two competing fluids is reached (see Fig. 6 on the left). This action on the part of the velocity field, however, this approach does not prevent the outer fluid from penetrating concave regions of the data set. An example of this situation is shown in Fig. 6, where the outer fluid can cross the dotted line on the right-hand side while is not possible in the left side. Notice that the front evolution convergence is guaranteed by the fact that each Voronoi cell contains a local central field. This implies that the contribution of each sample point can be considered zero outside its cell. The global velocity field can then be considered as the superposition of all the contributions from non-overlapping cells and it maintains irrotational and conservative behavior, which grants a stable and unique solution [42].

5. Fluid evolution equation

Wherever no sources are present Eq. (15) becomes

$$\frac{\partial \phi}{\partial t} = \nabla \cdot (\mathbf{v}\phi) + \xi \nabla^2 \phi. \tag{16}$$

This equation can be readily completed with proper initial conditions and a Dirichlet condition to describe the virtual fluid sources at the boundary

$$\begin{cases} \frac{\partial \phi(\mathbf{x},t)}{\partial t} = \nabla \cdot (\mathbf{v}\phi(\mathbf{x},t)) + \xi \nabla^2 \phi(\mathbf{x},t) & t \geqslant 0,\ \mathbf{x} \in \Omega \\ \phi(\mathbf{x},t) = +1, & t \geqslant 0,\ \mathbf{x} \in \mathscr{B} \\ \phi(\mathbf{x},0) = -1, & \mathbf{x} \in \Omega \end{cases} \tag{17}$$

The first equation of (17) is a first-order PDE, which can be discretized using a standard finite differences scheme. The second term of the right-hand side of this equation accounts for the contribution of a parabolic second-order PDE with the same formulation as the Heat Equation [43]; The second equation of (17) is a standard Dirichlet condition for border constraint while the third equation represents the system's initial condition. We recall that solving the classical Heat Equation is equivalent to performing a Gaussian filtering [44]. In what follows, for the sake of simplicity, we propose a one-dimensional example to explain this fact, but its extension to the n-dimensional case is rather straightforward.

Let consider a one-dimensional implicit function whose temporal evolution is governed by the following PDE:

$$\frac{\partial u(x,t)}{\partial t} - \frac{\partial^2 u(x,t)}{\partial x^2} = 0. \tag{18}$$

The explicit solution of a PDE with an initial condition of the form $u(x,0) = u_0(x)$ is given by

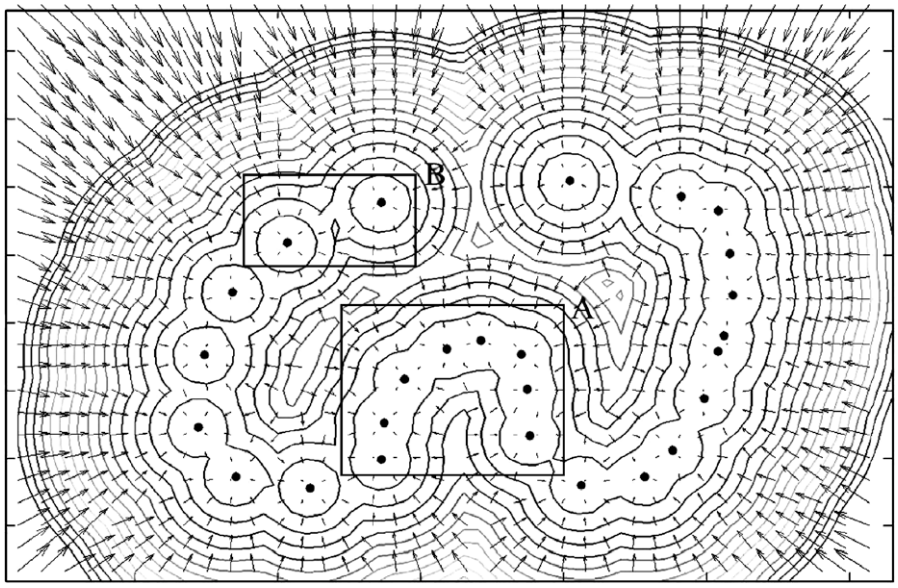


Fig. 4. The velocity vector field is oriented towards the nearest point and is proportional to the distance from it. The iso-velocity contours are also shown. The 'A' and 'B' regions are expanded in Figs. 5 and 6, to explain the behavior of the fluids in convex and non-convex regions.

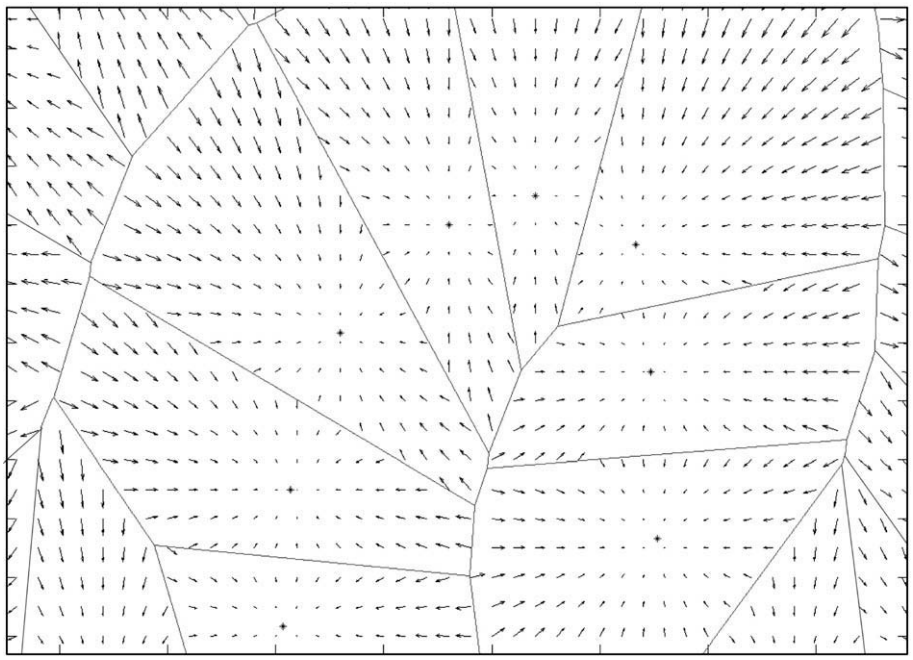


Fig. 5. Voronoi subdivision of the inner region 'A' of Fig. 4, in each cell the field is strictly central and oriented towards the point.

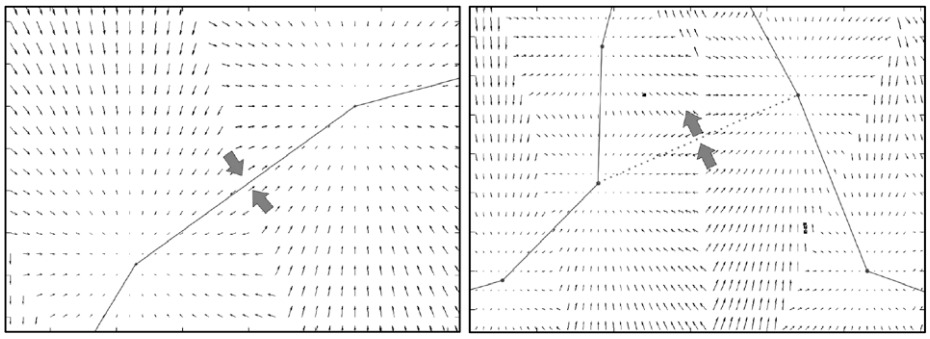


Fig. 6. (Left) Velocity field in square B of Fig. 5: flux from outside is unable to flow inside due to the repulsion. (Right) Velocity field in lower part of square A of Fig. 5, velocity is always oriented towards the inner part of the hole allowing outer fluid to penetrate the convex-hull honoring non-convex clouds of points.

$$u(t,x) = \int_{\mathbb{R}} G_{\sqrt{2t}}(x-y)u_0(y)\,dy = (G_{\sqrt{2t}} * u_0)(x), \tag{19}$$

where the operator '$*$' denotes a convolution, while $G_\sigma(x)$ represents the Gaussian kernel:

$$G_\sigma(x) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{|x|^2}{2\sigma^2}\right). \tag{20}$$

This corresponds to low-pass filtering u_0. Eq. (19) gives the correspondence between the time t and the scale parameter σ of the Gaussian kernel. In our case this corresponds to smoothly interpolating the surface over the 3D data set. It should be quite clear how the diffusive constant ξ has an impact on the final surface: low values of ξ imply that the surface will be follow more closely the sample points at a price of an increased roughness. High levels of ξ, on the other hand, produce a smoother surface that is able to dampen the impact of noisy sample points, further details about the choice of the diffusive constant will be given in Section 6.

We will now show the 1-dimensional case of the evolutive algorithm of the proposed PDE. The extension to the 3-dimensional case is trivial. If we consider only the first term, the PDE becomes:

$$\phi(x,t+1) - \phi(x,t) = \frac{1}{2}(v(x+1)\phi(x+1,t) - v(x-1)\phi(x-1,t)), \tag{21}$$

where the initial value condition (Cauchy problem) consists of a $\phi = -1$ everywhere in the bounding box (the inner fluid fills all the available space) except for the boundary $\mathscr{B}$ where ϕ is always kept equal to +1 due to the presence of outer fluid sources. The second term represents a parabolic PDE and its finite difference approximation becomes

$$\phi(x,t+1) - \phi(x,t) = \xi(\phi(x-1,t) - 2\phi(x,t) + \phi(x+1,t)). \tag{22}$$

If we merge the two Eqs. (21), (22) and the Cauchy condition, we obtain

$$\begin{cases} \phi(x,t+1) - \phi(x,t) = \frac{1}{2}(v(x+1)+2\xi)\phi(x+1,t) \\ \qquad\qquad\qquad\qquad -2\xi\phi(x,t) - \frac{1}{2}(v(x-1)-2\xi)\phi(x-1,t) \\ \phi(x,0) = -1 \qquad\qquad x \notin \mathscr{B} \end{cases} \tag{23}$$

The flux of the outer fluid from the boundary of the system domain is described by

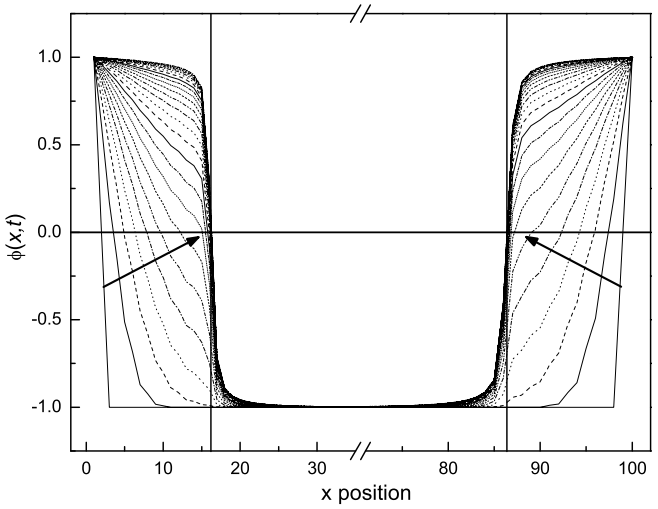


Fig. 7. Evolution of the implicit function $\phi(x,t)$ in the one-dimensional case: dashed lines are used for representing the implicit function at different time steps while arrows indicate the evolution towards the final value. At the last iteration the outer fluid fills all the external space outside the two points on the horizontal axis in positions 16.3 and 86.4. The inner fluid shrinks until it remains mainly confined within that gap. The VoF grid is located at integer coordinates.

$$\phi(x,t) = +1 \quad x \in \mathscr{B},\ \forall t \geqslant 0. \tag{24}$$

Fig. 7 shows the evolution of the fluid front in a one-dimensional case according to Eq. (23). The cloud of points is here represented by just two points on the horizontal axis located in positions 16.3 and 86.4, respectively. The spatial grid is placed at integer positions. The outer fluid starts flowing from the boundary and pushes the inner fluid inside while filling all the space outside the two points.

At the end of the evolution the zero level-set of the implicit function is located exactly in correspondence to the two points. The analysis of the evolution of the zero level-set curves shows also that the interface can be easily localized with sub-voxel accuracy. Obviously we lay no claims to the physical correctness of our equations since further terms should be consider to faithfully handle a real Newtonian fluid (i.e. viscous stresses). Furthermore conditions may happen (in particular far from the clouds of points, where the velocity is higher), where the total fluid amount for a grid cell overflows the range between −1 and +1; since we are dealing with an incompressible fluid this would have no meaning and to prevent it we clip the range of ϕ between −1 and +1 at each iteration. This has no drawback on the accuracy of the final reconstruction since it acts only on voxels far from the cloud of points and can be compared to the narrow band approach described in [33,34] where, to speed up the convergence of the level-set to the final solution its evolution is considered only in a narrow band across the zero level set.

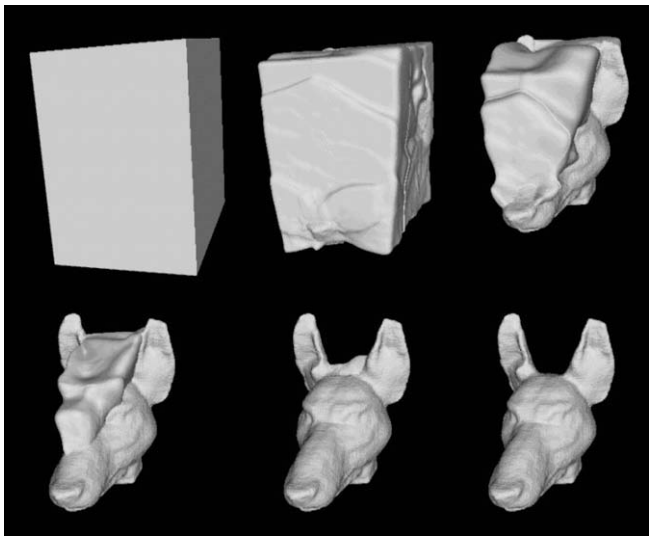


Fig. 8. Six different steps of the zero level-set during the evolution of the implicit function steered by the three-dimensional version of Eq. (23).

In Fig. 8 we provide six snapshots of the zero level-set evolution of the implicit function in a 3D environment. The reconstructed surface, represented by the zero level-set, progressively wraps the cloud of points starting from the initial bounding shape according to Eqs. (23) and (24).

6. Automatic adaptation of the diffusive constant

As shown in the previous Sections the diffusive constant ξ is a delicate parameter as it is strictly connected to fluid viscosity. High values of ξ result in a smooth but often inaccurate surface while low values result in a surface that strictly honors the data but is sensitive to acquisition noise. Our choice was made to provide good results with point clouds acquired with laser scanners. As a matter of fact, the portions of the object with high sample density are usually those acquired from multiple viewpoints because we need high accuracy and/or the topology is complex, while smoother or less critical areas usually exhibit a lower sample density. In order to account for that, we established an inverse dependency between the diffusive constant ξ and the local point density:

$$\xi = \frac{\alpha}{1+\rho_p}, \tag{25}$$

αbeing a parameter that indicates the estimated noise in the point location (the higher the value the smoother the surface). The density ρ_p at a given point is defined as the number of samples that fall into a cube of volume V/N around that point, where V is the volume of the point cloud (approximated by the volume of the smallest parallelepiped that contains the point cloud) and N is the total number of samples. This means that if all the samples would be uniformly distributed in the considered volume (like in a cubic crystal) in each cubic cell of volume V/N there would be just one sample. In our case, since we have to define the ρ_p variable for each voxel, we define it as the sum of all the points falling inside a cube of volume V/N centered at the voxel center. The point density is then 0 if there are no samples in the neighborhood of the considered voxel and the '*1*' at the denominator of Eq. 25 is added to prevent division by zero in the diffusive constant. In surface regions with a low point density the diffusive constant ξ corresponds to a smooth surface, while in high-density zones ξ becomes very low, to guarantee that the surface will closely honor the points. A 2D example is shown in Fig. 9 where the surface wraps a square with different point density on the corners. Changing the parameter α, on the other hand, has an impact on the global fluid viscosity depending to the sample noise. In Fig. 10 we show the behavior of the fluid in a 2D environment where the sources of the external fluid are placed just on the top of the domain. The points originate from an equally sampled sinusoid with varying frequency: different values of α turn out to inhibit fluid penetration at different heights, which leads to different curvatures on the final surface.

7. Convergence to the final solution

Convergence to the final solution is a long and debated term in computational fluid dynamics, as reported in [47], and is common practice to use Lax's equivalence theorem which states that for lin-

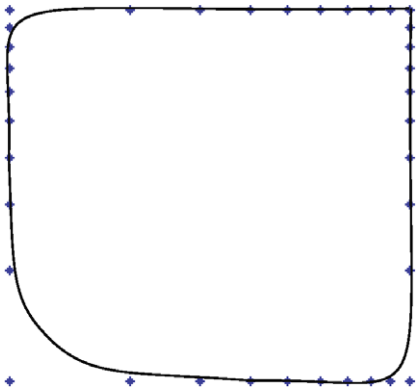


Fig. 9. Example of automatic adaptation of the diffusive constant: the corners are characterized by different sample densities therefore the final surface turns out to be sharp or smooth, accordingly.

ear problems a necessary and sufficient condition for convergence is that the method is both consistent and stable. Where'consistency' means that the numerical schemes produce systems of algebraic equations which can be demonstrated to be equivalent to the original governing equation while'stability' is associated with damping of errors as the numerical method proceeds. Anyway, our equations as most of the Computational Fluid Dynamics, are non-linear; in such problems consistency and stability are necessary conditions for convergence but not sufficient. Our inability to prove conclusively that a numerical solution scheme is convergent is perhaps somewhat unsatisfying from a theoretical standpoint, but this is a common issue on this topic and is a regular practice to stop fluid evolution whenever the variations of each voxel are beneath a threshold. We also followed this practice choosing a threshold of 0.01 for the variation of the implicit function of a voxel. We experimented, on common clouds of points, that convergence is usually reached in a number of iterations that is between three to four times the number of voxel in the wider dimension of the voxel-set.

8. Speeding up convergence and real-fluid emulation: the vorticity term

In the previous Section we described in general terms how our algorithm works. Due to the close similarity of our approach with fluid dynamics we also analyzed the system's behavior as it implements additional fluids properties. In particular, we introduced the phenomenon of vorticity to evaluate its impact on the surface modeling process. We also worked on the fusion of multiple surfaces based on the fusion of the corresponding fluid configurations. Vorticity is strictly correlated to the turbulent flow regime, neglecting further fluid dynamics details [47], we just recall that when the external forces (convective effect) exceed viscous forces (internal fluid forces) the fluid overcomes the critical Reynolds number and its flow from laminar become chaotic. This behavior can be simulated varying randomly (obviously within a certain range) and isotropically the forces acting on each fluid cell; This results on our model is an arbitrary variation of velocity field for each voxel at each iteration. The fluid evolution can take advantage of this behavior in particular conditions as shown below.

Due to the surface's roughness and high curvature, the velocity field rapidly changes direction. This usually results in a slower motion of the fluids towards a final configuration. The vorticity term tends to overcome this drawback and improve convergence time. Vorticity describes the presence of turbulence in fluid motion. Its introduction allows us to describe local variations in the velocity term and to localize whirls. From the physical point of view the vorticity's contribution is defined as the rotational component in the velocity vector field

$$\zeta = \nabla \times \mathbf{v}(\mathbf{x}), \tag{26}$$

whose intensity characterizes the turbulent status. The rotational behavior can be illustrated by defining an imaginary loop C around the whirl, and then integrating the flow velocity around that loop

$$\Lambda = \int_C \mathbf{v} \cdot \mathbf{dl}. \tag{27}$$

Stokes' theorem applied to Eq. (27) gives

$$\Lambda = \int_C \mathbf{v} \cdot \mathbf{dl} = \int_S (\nabla \times \mathbf{v}) \cdot \mathbf{dS} = \int_S \zeta \cdot \mathbf{dS}, \tag{28}$$

where the right hand-side integral is computed throughout the surface S defined by the closed loop C. The vorticity thus represents the *local* rotational characteristics of the flow.

The presence of whirls can locally develop high-pressure gradients, and may generate instabilities in the flow. At the same time, however, it favors penetration in narrow openings and prevents fluid from stagnating in 'low speed' regions. Although the localization of vortices and the determination of their intensity is a diffi-

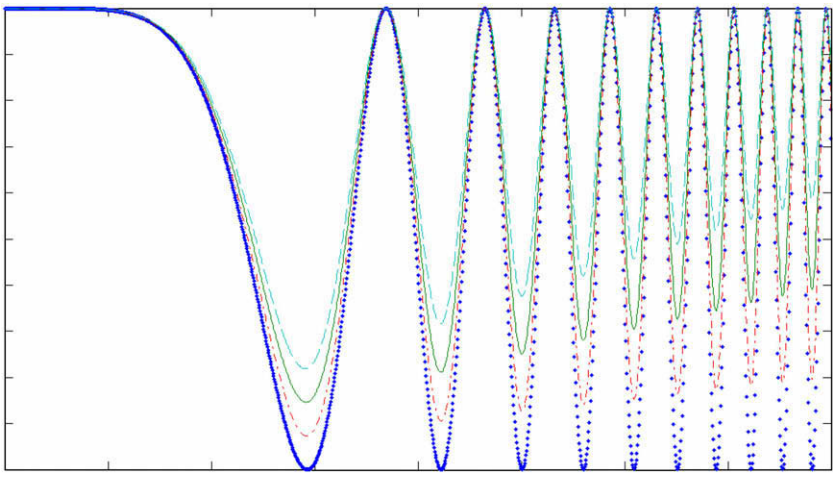


Fig. 10. Fluid behavior on a sampled sinusoid with varying frequency: different values of α result in different penetration depths and final surface's crispness. The three broken lines from top to bottom are the reconstructions that correspond to increasing values of α.

cult numerical problem to solve, we cannot neglect them if we want to correctly model our system. In particular, vorticity's introduction does not preserve the irrotational property of the velocity vector field and does not guarantee that the system will converge to a static final solution, therefore we must be very cautious in using it.

In our solution we artificially introduce the presence of vortices by modulating the velocity field with small oscillations around the initial configuration. In fact, we are not interested in the introduction of static whirls and we know that there is a close correlation between the velocity vector and the vector that points towards the nearest point (Section 7). A modulation in the velocity field can thus be interpreted as a plastic deformation of the cloud of points in different directions. Roughly speaking, this operation is similar to forcing jello through a funnel by squeezing the funnel itself. The modulation of the velocity function is obtained by altering the speed of expansion along each one of the three spatial axes in turn. More specifically, for each axis that we consider, we multiply the velocity component along that axis by a given coefficient, while we divide the other two components by the same factor. At the next iteration we change the axis along which apply the expansion. This results in a cyclic modulation of the velocity function that induces the presence of whirls. A expansion coefficient ranges from 1 to 1.5; larger values result in an excessive fluid, which requires a longer time to settle down after turning off the vorticity. Using vorticity can significantly boost the convergence and the penetration of the external fluid in grooves and holes. In the final iterations, however, this term must be turned off to allow the fluids to gently converge to an accurate and stable solution. Vorticity, as a rule of thumb, can be applied when the wrapping surface is close to the final result (no significant variations take place between consecutive iterations); its activation results in a better fluid penetration in stagnant regions; anyway, we found that after its extinction, its a good rule to play at least 10 iterations to let the system settle down to the final configuration.

9. Octree surface subdivision for reconstruction improvement

An important topic in 3D surfaces from point clouds is the ability to provide high accuracy in complex and dense regions, and, at the same time, keep the total number of triangles in uniform regions modest to avoid wasting memory. A possible approach to improve surface detail without increasing the total voxel-set resolution is based on the octree approach [45]. In order to employ this multi-resolution voxel structure we reused the point density ρ_p already defined in Section 6. If the density exceeds a given threshold (we experimentally found that 4 is a reasonable choice of that threshold) the voxel and its 26 nearest neighbors are subdivided in 8 subvoxels (octree subdivision), each one inheriting the value of the parent voxel. The velocity field is then recomputed over the sub-voxels. The evolution process can then be restarted just on the sub-voxel set. In particular, the role of the source will now be played by the bordering sub-voxels.

10. Experimental results and implementational details

In this section we show the results of some surface reconstruction experiments from classical clouds of points. We will show, in particular, that the system is able to achieve a remarkable reconstruction speed and manage large clouds of points in high spatial resolution without special implementation requirements.

There is another important aspect in our algorithm that has not been emphasized enough. It concerns the possibility to easily parallelize the algorithm through an individual processing of different parts of the given cloud of points. Each one of these processes can be assigned different resolutions and different control parameters. When all the individual reconstructions have reached a stable solution, we can merge them into a larger voxel-set, and with a resolution that corresponds to the maximum one among those used for the individual portions. In order to smoothly merge different regions, we make sure that a small overlapping region is defined across separate zones. At the end of the process all regions are placed in their final position and each voxel value is set as (see Figs. 11):

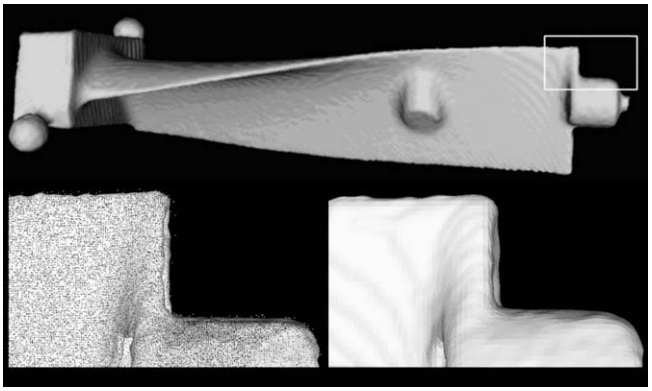


Fig. 11. A sharp blade-like mechanical element; the model is obtained from a cloud of about 1.5 million points on a grid of $180 \times 70 \times 70$ cells. Bottom: two enlargements of the framed zone with different viscosity levels: the viscosity on the left is lower than on the right. The acquired points are affected by noise and a higher level of viscosity allows us to obtain a smoother and more faithful reconstruction of the model.

$$F(\mathbf{x}) = \inf_i F_i(\mathbf{x}), \tag{29}$$

where the inf operator is used above all the implicit i functions whose voxel-set contains the voxel in position $\mathbf{x}$. The farther inside the voxel, the lower the implicit function, therefore we use the lower value because it belongs to the cloud of points that better contains the point. The rabbit surface, shown in Fig. 12 (right) was obtained by merging three different and partially overlapping implicit functions (see Fig. 12 on the left). The single portions are separately rendered and are then placed together using the previously-described operator. Octree subdivision can be applied in the joining regions and, as described above for this approach, the level-set evolution can be restarted inside the octree subdivision to obtain a gentle fusion among the parts. furthermore, a simpler merging solution can be based on restarting the evolutive equation in a narrow region close to the joining surfaces: obviously, since we do not have the velocity field for the whole cloud of points we can just apply the diffusive term that will result in a smooth seamless fusion of the surfaces. Attention must be paid to limit the region of its application to avoid over-smoothing of the whole surface.

The proposed algorithm was run on an AMD Athlon™ XP processor running at 2.1 GHz with 512 MB RAM under Windows™ 2000. The Wolf, the Teapot and the Blade are interesting data-sets, as they were acquired using different multi-camera systems and methods. The data, anyway, is a bit noisy. The resulting mesh is obtained using the marching cubes algorithm [46]: the voxel-set triangulation approximates the zero level-set of the implicit function placing the vertices of the triangles between voxels that have different signs in their implicit function. This means, as shown in Section 5 (Fig. 7), that such two voxels belong to different sides (inside–outside) of the considered object. The resolution of the final surface is higher that the voxel-set one (and even of the octtree subdivision, if applied) since linear interpolation is used be-

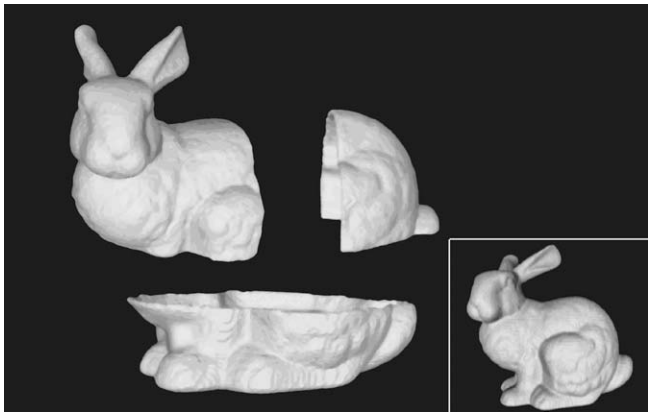


Fig. 12. Parallelization of the proposed algorithm: surfaces of different parts of the rabbit's cloud of points are separately computed and then merged into a single surface as shown on the right-hand image.

Table 1
Reconstruction performances on different clouds of points shown in Figs. 11–13, no parallelization of the algorithm is used

Model	Resolution	Points	Time(s) our	Time(s) L-S
Bunny	$180 \times 178 \times 140$	35780	40	360
Bunny	$100 \times 99 \times 78$	35780	4	48
Buddha	$144 \times 350 \times 144$	3836	105	2215
Teapot	$256 \times 200 \times 179$	33061	110	4210
Blade	$300 \times 70 \times 179$	1550316	120	7560

The last two columns show a comparison between our approach and the classical Level-set.

Fig. 13. Some examples of surface reconstruction from different clouds of points.

tween different voxel values of the implicit function $\phi(\mathbf{x}, t)$. This allows us to obtain a better estimate of the zero level-set and, therefore, the position of each triangle vertex. Comparisons were made, for reconstruction time, with the level-set approach [25] that we keep as the most significant approach to surface reconstruction based on a PDE evolution of an implicit function. In Table 1 we provide some results obtained using the proposed approaches(see Fig. 13).

Fig. 14. (Up) Our reconstruction of the dragon head. (Down) The same reconstruction using Raindrop Geomagic 7; curves are also drawn on both models to estimate surface roughness.

A further comparison was conducted to evaluate the quality of our reconstruction solution in comparison with a commercial software package (Geomagic Studio 7[3]) specialized for triangulating clouds of points. The results are shown in Fig. 14. The model generated with Geomagic required further effort to delete some wrong links among parts of the cloud of points. Even if the data fitting (average distance of points from the reconstructed surface) in our model is slightly worse than the Geomagic triangulation (less than 2%), our model exhibits a smoother surface. In order to measure surface roughness we sliced both models with a set of planes (in Fig. 14 we show curves from five parallel planes intersections) and then we examined their curvature. The average curvature in our reconstruction is almost two orders of magnitude lower than the Geomagic one.

11. Conclusions

We proposed a novel volumetric approach to surface modeling from unorganized sets of points which is able to overcome the typical problems of computational efficiency of level-set methods. In addition, we gave the algorithm the ability to model complex topologies using advanced fluid-dynamics properties of fluids; this allows us to combine computational efficiency with an intuitive effect of different parameters such as viscosity and vorticity. The re-

[3] Geomagic Studio 7 is a trademark of Raindrop Geomagic, Inc.

sults in terms of both computational efficiency and topological flexibility are quite encouraging and make the approach extremely usable.

References

[1] P. Dias, V. Sequeira, F. Vaz, J. Goncalves, Registration and fusion of intensity and range data for 3D modelling of real world scenes, in: Fourth International Conference on 3-D Digital Imaging and Modeling, 3DIM Proceedings, 2003, pp. 418–425.

[2] H. Hoppe, Surface reconstruction from unorganized points, Ph.D. Thesis in Computer Science and Engineering, University of Washington, 1994.

[3] N. Amenta, M. Bern, D. Eppstein, The crust and the β-skeleton: combinatorial curve reconstruction, Graph. Models Image Process. 60 (2) (1998) 125–135.

[4] H. Edelsbrunner, Shape reconstruction with delaunay complex, in: Proceedings of LATIN'98: Theoretical Informatics, Lecture Notes in Computer Science, vol. 1380, Springer-Verlag, 1998, pp. 119–132.

[5] L. Piegel, W. Tiller, The NURBS Book, second ed., Springer-Verlag, Berlin, Germany, 1996.

[6] D.F. Rogers, An Introduction to NURBS, Morgan Kaufman, 2000.

[7] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle, Surface reconstruction from unorganized points, in: SIGGRAPH'92 Proceedings, 1992, pp. 71–78.

[8] B. Curless, M. Levoy, A volumetric method for building complex models from range images, in: SIGGRAPH'96 Proceedings, 1996, pp. 303–312.

[9] H. Edelsbrunner, D. Kirkpatrick, R. Seidel, On the shape of a set of points in the plane, IEEE Trans. Inf. Theory 29 (4) (1983) 551–559.

[10] H. Edelsbrunner, E.P. Mücke, Three-dimensional alpha shapes, ACM Trans. Graph. 13 (1994) 43–72.

[11] C. Bajaj, F. Bernardini, G. Xu, Automatic reconstruction of surfaces and scalar fields from 3D scans, in: SIGGRAPH'95 Proceedings, 1995, pp. 109–118.

[12] J.-D. Boissonnat, Geometric structures for three-dimensional shape reconstruction, ACM Trans. Graph. 3 (1984) 266–286.

[13] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, C.T. Silva, Computing and rendering point set surfaces, IEEE Trans. Comput. Graph. Vis. 9 (1) (2003) 3–15.

[14] A. Adamson, M. Alexa, Ray tracing point set surfaces, in: Shape Modeling International, 2003.

[15] N. Amenta, S. Choi, R. Kolluri, The power crust, in: Proceedings of 6th ACM Symposium on Solid Modelling, 2001, pp. 249–260.

[16] G. Turk, J. O'Brien, Shape transformation using variational implicit functions, in: SIGGRAPH'99 Proceedings, 1999, pp. 335–342.

[17] A. Adamson, M. Alexa, Anisotropic point set surfaces, in: Proceedings of the 4th International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa, 2006, pp. 7–13.

[18] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle, Surface reconstruction from unorganized points, in: SIGGRAPH'92 Proceedings, 1992, pp. 71–78.

[19] J. Sethian, Level Set Methods and Fast Marching Methods Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science, Cambridge University Press, 1999.

[20] S. Osher, R. Fedkiw, Level set methods: an overview and some recent results, J. Comput. Phys. 169 (2001) 463.

[21] J. Sethian, Evolution, implementation, and application of level set and fast marching methods for advancing fronts, J. Comput. Phys. 169 (2001) 503.

[22] D. Enright, F. Losasso, R. Fedkiw, A fast and accurate semi-lagrangian particle level set method, Comput. Struct. 83 (2005) 479–490.

[23] S. Osher, R. Fedkiw, Level Set Methods and Dynamic Implicit Surfaces, Springer, 2002.

[24] S. Ilic, P. Fua, Implicit meshes for surface reconstruction, IEEE Trans. Pattern Anal. Mach. Intell. 28 (2) (2006) 328–333.

[25] H. Zhao, S. Osher, R. Fedkiw, Fast surface reconstruction using the level set method, in: Proceedings of IEEE Workshop on Variational and Level Set Methods in Computer Vision (VLSM 2001), 2001.

[26] M. Marcon, L. Piccarreta, A. Sarti, S. Tubaro, A fast level-set approach to surface modeling from unorganized sample points, in: Proceedings of 2nd IEEE Workshop on Variational and Level Set Methods in Computer Vision (VLSM'03), vol. a, 2003, pp. 191–196.

[27] M. Marcon, L. Piccarreta, A. Sarti, S. Tubaro, Fast point-cloud wrapping through level-set evolution, in: Proceedings of 1st European Conference on Visual. Media Prod., 2004, pp. 119–125.

[28] D. Gueyffier, J. Li, A. Nadim, R. Scardovelli, S. Zaleski, Volume-of-fluid interface tracking with smoothed surface stress methods for three-dimensional flows, J. Comput. Phys. 152 (1999) 423–456.

[29] D. Youngs, Time-dependent Multi Material Flow with Large Fluid Distortion, K.W. Morton, M.J. Baines, 1982, pp. 273–285.

[30] W. Noh, P. Woodward, A simple line interface calculation, in: A. vn de Vooran, P. Zandberger (Eds.), Proceedings of 5th International Conference on Fluid Dynamics, Springer-Verlag, 1976.

[31] R. Whitaker, A level-set approach to 3D reconstruction from range data, Int. J. Comput. Vis. 29 (3) (1998) 203–231.

[32] H. Zhao, S. Osher, B. Merriman, M. Kang, Implicit and non-parametric shape reconstruction from unorganized points using variational level set method, Comput. Vis. Image Understanding 80 (3) (2000) 295–319.

[33] D. Peng, B. Merriman, S. Osher, H. Zhao, M. Kang, A pde based fast local level set method, J. Comput. Phys. 155 (1999) 410–438.

[34] M. Sussman, P. Smereka, S. Osher, A level set approach for computing solutions to incompressible two-phase flows, J. Comput. Phys. 119 (1994) 146–159.

[35] J. Gomes, O. Faugeras, Reconciling distance functions and level sets, Tech. Rep. 3666, INRIA Sophia Antipolis, 1999.

[36] F.S.L.F.E. Ham, A.B. Strong, A cartesian grid method with transient anisotropic adaptation, J. Comput. Phys. 179 (2002) 469–494.

[37] B.M.R. Fedkiw, T. Aslam, S. Osher, A non-oscillatory eulerian approach to interfaces in multimaterial flows, J. Comput. Phys. 152 (1999) 457–492.

[38] S.X.R. Fedkiw, T. Aslam, The ghost fluid method for deflagration and detonation discontinuities, J. Comput. Phys. 154 (1999) 393–427.

[39] T. Aslam, A level set algorithm for tracking discontinuities in hyperbolic conservation laws i: scalar equations, J. Comput. Phys. 167 (2001) 413–438.

[40] J.C. Carr, R.K. Beatson, B.C. McCallum, W.R. Fright, T.J. McLennan, T.J. Mitchell 1, Smooth surface reconstruction from noisy range data, in: ACM GRAPHITE, 2003, pp. 119–126.

[41] C. Hirsh, Computation of internal and external flows, Series in Numerical Methods in Engineering, Wiley Interscience, 2001.

[42] C. Wylie, Advanced Engineering Mathematics, fourth ed., McGraw Hill, 1975.

[43] A. Quarteroni, R. Sacco, F. Saleri, Numerical Mathematics, Springer-Verlag, 2000.

[44] G. Aubert, P. Kornprobst, Mathematical problems in image processing: partial differential equations and the calculus of variations, Applied Mathematical Sciences, no. 147, Springer-Verlag, 2000.

[45] S.L.T.C.L. Jackins, Oct-trees and their use in representing three-dimensional objects, Comput. Graph. Image Process. 14 (3) (1980) 249–270.

[46] W.E. Lorensen, H.E. Cline, Marching cubes: a high resolution 3D surface construction algorithm, in: SIGGRAPH'87 Proceedings, vol. 21, 1987, pp. 163–169.

[47] H.K. Versteeg, W. Malalasekera, An Introduction to Computational Fluid Dynamics, The Finite Volume Method, Longman Scientific and Technical, 1995.

[48] H.K. Zhao, S. Osher, Analysis, shape recontruction and visualization from sparse data, in: S. Osher, N. Paragios (Eds.), In Geometric Level Set Methods in Imaging, Vision and Graphics, Springer, 2003.