Journal of Mathematical Imaging and Vision (2026) 68:14
https://doi.org/10.1007/s10851-026-01292-y

Implicit Reconstruction from Point Cloud: An Adaptive Level Set-Based Semi-Lagrangian Method

Silvia Preda[1] · **Matteo Semplice**[1]

Received: 6 November 2025 / Accepted: 24 March 2026 / Published online: 14 April 2026

Abstract

We propose a level set-based semi-Lagrangian method on graded adaptive Cartesian grids to address the problem of surface reconstruction from point clouds. The goal is to obtain an implicit, high-quality representation of real shapes that can subsequently serve as computational domain for partial differential equation models. The mathematical formulation is variational, incorporating a curvature constraint that minimizes the surface area while being weighted by the distance of the reconstructed surface from the input point cloud. Within the level set framework, this problem is reformulated as an advection–diffusion equation, which we solve using a semi-Lagrangian scheme coupled with a local high-order interpolator. Building on the features of the level set and semi-Lagrangian method, we use quadtree and octree data structures to represent the grid and generate a mesh with the finest resolution near the zero level set and the point cloud data. The complete surface reconstruction workflow is described, including localization and reinitialization techniques, as well as strategies to handle complex and evolving topologies. A broad set of numerical tests in two and three dimensions is presented to assess the effectiveness of the method.

Keywords Surface reconstruction · Point cloud · Level Set Method · Adaptive Mesh Refinement · Semi-Lagrangian schemes · CWENO interpolation

1 Introduction

Many real-world applications share the issue of acquiring and processing 3D digital models of non-synthetic objects. One of the common fundamental problem consists in reconstructing a smooth surface from a given set of unorganized points, called as *point cloud*. This is typical, for instance, in fields like cultural heritage, where one deals with real artistic manufacts presenting complicated geometries and topologies, and whose shape is usually acquired, in the form of a point cloud, from the work of art itself, e.g. via 3D laser scanning or photogrammetry [1–3].

The importance of having a mathematical description of real shapes also lies in the problem of monitoring and predicting damage and degradation of monuments, for which some mathematical models, see for instance [4–6], can be applied. These models require the solution of reaction-diffusion Partial Differential Equations (PDEs) on a computational domain having the exact shape of a work of art. This paper addresses the problem of providing a high fidelity description of a complex object to be later used as domain definition in PDE computations, in particular via ghost-cell methods [7–10]. As a consequence, we seek a watertight representation, having some smoothness properties, which is suitable not only for static operations, but also for dynamic ones. Computations will involve high-order methods and grids finer than the resolution of the point cloud, so that the reconstructed object can then be reliably represented on the extremely fine grids required by the PDE model.

In general, a surface can be represented by following two approaches: the explicit and the implicit one. Regarding the first, we mention mesh-based [11, 12] and parametric techniques, e.g. NURBS [13, 14], which, despite their popularity in a wide variety of applications, are well-known to be difficult to handle when an evolution of the surface occurs. On the other hand, an implicit approach offers a better handling of topological flexibility, while having a very simple data struc-

✉ Silvia Preda
silvia.preda@uninsubria.it

Matteo Semplice
matteo.semplice@uninsubria.it

[1] Department of Science and High Technology, University of Insubria, Via Valleggio 11, Como 22100, Italy

ture that allows simple Boolean operations on the detected surface, too. Radial Basis Functions (RBF) with global and local support have found many applications along this line [15–17] together with some least-squares-based methods [18]. Still, a signed distance function representing a surface can be obtained by using local estimators that associate an oriented plane to each point in the cloud [19]. A different and widespread approach for the processing of evolving surfaces, and point cloud data in particular, is constituted by the application of the Level Set Method (LSM) [20].

Introduced in [20], the LSM has emerged as a powerful and versatile tool in a wide range of applications [7, 21, 22] including image processing and surface reconstruction [23, 24]. In its framework, an n-dimensional object Ω and its boundary Γ are represented by a so-called level set function $\phi : \mathbb{R}^n \to \mathbb{R}$ such that $\Omega = \{\vec{x} \in \mathbb{R}^n : \phi(\vec{x}) < 0\}$ and Γ is the zero isocontour of ϕ, namely $\Gamma = \{\vec{x} \in \mathbb{R}^n : \phi(\vec{x}) = 0\}$. Exploiting their representation via the scalar function ϕ, Ω and Γ can be then evolved in time accordingly to a PDE defined for ϕ, driving the deformation of an initial surface.

Along this line, a level set-based semi-Lagrangian method with local interpolator has been presented in [25] for the problem of surface reconstruction from point cloud, while a first complete workflow from the dataset to PDE computations for the marble sulfation phenomenon can be found in [26, 27].

The aforementioned method numerically solves the model introduced by Zhao et al. [23], which is based on the minimization of the energy functional

$$E_p(\Gamma) = \left(\int_\Gamma d^p(\vec{x}) \, \mathrm{d}s \right)^{1/p}, \quad 1 \le p \le \infty, \tag{1}$$

where $d(\vec{x}) = \min_{\vec{q} \in \mathcal{S}} |\vec{x} - \vec{q}|$ is the distance function of a point $\vec{x} \in \mathbb{R}^n$ from the point cloud $\mathcal{S} = \{\vec{q}_1, \ldots, \vec{q}_M\}$, and Γ is a closed surface of co-dimension one in $\mathbb{R}^n$. The minimum of (1), namely the final shape, is obtained by continuously deforming an initial surface Γ^0, following the gradient descent of (1).

Within the LSM, the evolving surface $\Gamma(t)$ is represented implicitly using a level set function to capture the moving interface, leading to the level set formulation

$$\begin{aligned} \phi_t(\vec{x}, t) = &\left[\frac{d(\vec{x})}{E_p(\phi)} \right]^{p-1} \left(\nabla d(\vec{x}) \cdot \nabla \phi(\vec{x}, t) \right. \\ &\left. + \frac{\mu}{p} d(\vec{x}) \nabla \cdot \left(\frac{\nabla \phi(\vec{x}, t)}{|\nabla \phi(\vec{x}, t)|} \right) |\nabla \phi(\vec{x}, t)| \right), \\ \phi(\vec{x}, 0) = &\phi^0(\vec{x}), \end{aligned} \tag{2}$$

where the energy functional (1) is coherently rewritten as

$$E_p(\phi) = \left(\int_{\mathbb{R}^n} |d(\vec{x})|^p \delta(\phi) |\nabla \phi| \, \mathrm{d}\vec{x} \right)^{1/p}, \tag{3}$$

being δ the Dirac-delta function, and $\phi^0(\vec{x})$ a suitable initial data such that $\Gamma^0 = \{\vec{x} \in \mathbb{R}^n : \phi^0(\vec{x}) = 0\}$. In (2), the evolution of the surface towards the dataset $\mathcal{S}$ is driven by the term $\nabla d(\vec{x}) \cdot \nabla \phi(\vec{x}, t)$, while the second term tempers the maximal curvature of Γ. The balance between these two is tuned by the parameter μ, introduced in [25] for this purpose and to deal with noisy data, too. Finally, among all the possible level set representation of the surface Γ, we aim to work with the signed distance one, thus taking into account suitable reinitializazion procedure that guarantees our function ϕ to stay well-behaved, namely $|\nabla \phi| \approx 1$, during the evolution. For a wide description of the model and the parameters acting in (2) the reader can refer to [23, 25, 28].

The numerical solution of (2) can be addressed in different ways, see for instance [28–31]. In particular, in [30], a semi-Lagrangian (SL) scheme is adopted with the aim of overcoming the prohibitive time step constraint imposed by the diffusion term. In fact, first introduced in [32] for first-order systems of linear equations, SL schemes have gone through an important extension with the main purpose of obtaining methods which are unconditionally stable with respect to the choice of the time step (see [33] for a comprehensive explanation), even in presence of parabolic terms, whose treatment has been studied in [34, 35].

In the level set context, the possibility to overcome the parabolic-type CFL restriction is particularly tempting because it allows one to employ very fine grids for the computations, especially in the vicinity of the zero level set of ϕ, also from an Adaptive Mesh Refinement (AMR) perspective. While the CFL condition will frustrate the adaptive approach when a general explicit scheme is employed, this will not be an issue in the SL framework where one is allowed to work at large Courant numbers, with a hyperbolic CFL restriction of type $\Delta t = \mathcal{O}(\Delta x)$. For more details, we refer the reader to [36], where a first application of SL schemes in the LSM context can be found.

Motivated by the reasons above, here we aim to extend the method presented in [25] to a fully adaptive framework, exploiting both the unconditional stability of the SL approach and the adaptivity criterion offered by the level set function itself. In fact, given a level set function ϕ associated to an interface Γ, constructing a tree-based grid refined around Γ is quite simple since the values of $|\phi|$ naturally give a proper criterion for refinement, yielding a smaller grid size close to Γ and a larger one far away from the interface. Moreover, we will couple this criterion with the proximity to the point cloud $\mathcal{S}$, expressed by the distance function. As a result, the multiple iterations of the method presented in [25] are abandoned, in favour of a single computation in which the resolution of the computational grid increases along the zero level set and as we get closer to $\mathcal{S}$.

On the other side, SL algorithms, compared to Eulerian ones, might not be easy to parallelize, especially on adap-

tive grids, since, depending of the CFL number, the feet of the characteristics may end up outside the halo region, in remote ranks, and may also significantly cluster, affecting load balancing. In order to reduce the computational effort and communications among processors, the SL scheme is coupled with two types of local reconstructions, a multilinear (P1) and a third-order accurate Central Weighted Essentially Non-Oscillatory (CWENO) one, both based on a least-square approach due to the lack of structure in the mesh employed.

First pioneering studies related to central reconstructions can be found in [37], where the authors suggested to blend, in a WENO-like fashion, polynomials of different degrees, allowing to overcome some difficulties of non-existence, non-positivity and dependence on the reconstruction point of the WENO linear weights. The idea has been further developed into the so-called CWENO reconstruction and exploited in multiple spatial dimensions, also in the case of adaptive mesh refinement and non-uniform grids [38–42]. The technique has also been exploited in finite difference schemes for Hamilton-Jacobi (HJ) equations on Cartesian meshes via dimensional splitting [43, 44], as well as on general meshes [45]. General results for establishing the convergence order of a CWENO reconstruction have been presented in [46–48].

In order to design the complete numerical method and achieve the final reconstruction via a signed distance function, some other techniques will be employed, including localization procedure [49], reinitialization [20, 21, 50] and distance function computation [51].

The paper is organized as follows. The SL scheme for the numerical approximation of (2) is described in Sect. 2. All the auxiliary schemes needed for the level set computation are collected in Sect. 3. In Sect. 4, the whole algorithm is summarized and various numerical tests in two and three dimensions are presented. Finally, Sect. 5 collects some conclusions and future perspectives.

2 Numerical Scheme

We consider a background quadtree (resp. octree) mesh having a single tree-based structure which is defined on a cubic domain $\Omega' \subset \mathbb{R}^2$ (resp. $\mathbb{R}^3$) containing the point cloud $\mathcal{S}$ and $\Omega(t)$, at each time $t > 0$. In what follows, the set of quadrants composing Ω' will be denoted by $\mathcal{Q}$ and, for $j \in \mathcal{Q}$, ϕ_j^n indicates the approximate value of ϕ evaluated at the centre $\vec{x}_j$ of the quadrant j at time t^n. Moreover, for each quadrant j, we denote with Δx_j its edge length and, once the maximum level of refinement L is fixed, we define as

$$\Delta x_{\min} = \frac{\Delta\Omega'}{2^L} \tag{4}$$

the minimum edge length of the quadrants composing the tree, where $\Delta\Omega'$ is the edge length of the cubic domain Ω'.

The discretized values of ϕ, namely the numerical evolution of (2), are computed following the SL approach presented in [30], coupled with a suitable local P1 or CWENO reconstruction to handle the quadtree grid, reduce the computational costs and exploit the parallelization. The SL scheme will be briefly described in the next Sect. 2.1, for the two-dimensional case, while the reader can refer to [25, 30] for a complete description in three dimensions. The complementary techniques and the complete algorithm will be described later on, in Sect. 3 and Sect. 4, respectively.

2.1 Semi-Lagrangian Scheme

In two dimensions, we compute the update of ϕ_j^n as

$$\begin{aligned} \phi_j^{n+1} &= \frac{1}{2}\sum_{i=1}^{2} \mathcal{R}[\Phi^n]\left(\vec{x}_{j,i}^{\,*}\right), \\ \vec{x}_{j,i}^{\,*} &= \vec{x}_j + C_j^n \Delta t \nabla d(\vec{x}_j) \\ &\quad + \sqrt{\frac{2C_j^n\, \mu\, d(\vec{x}_j)\Delta t}{p}}\, \vec{\sigma}_j^{\,n} \lambda_i, \end{aligned} \tag{5}$$

where λ_i ranges over $\{-1, +1\}$ and C_j^n is the scale factor $\left[\frac{d(\vec{x}_j)}{E_p(\phi^n)}\right]^{p-1}$. The operator $\mathcal{R}[\Phi^n](\vec{x})$ denotes a reconstruction at point $\vec{x}$ of the data $\Phi^n = \{\phi_j^n : j \in \mathcal{Q}\}$, which will be specified later in Sects. 2.2 and 2.3. The unit vector $\vec{\sigma}_j^{\,n}$ is given by

$$\vec{\sigma}_j^{\,n} = \frac{1}{|\nabla\phi_j^n|}\begin{pmatrix} \partial_y \phi_j^n \\ -\partial_x \phi_j^n \end{pmatrix}, \tag{6}$$

is tangent to the level sets of ϕ and thus orthogonal to its gradient.

One can notice that, in equation (5), the evaluation points of the reconstruction operator are obtained by computing the foot of the characteristic pertaining to the advection term in (2), and adding a further displacement of size $\sqrt{\Delta t}$ that generates a diffusion along the tangent space of the level sets, thus discretizing the curvature term in (2), as described in [52, 53].

Because of the factor $1/|\nabla\phi_j^n|$ inside the definition of $\vec{\sigma}_j^{\,n}$ in (6), when $|\nabla\phi_j^n| < D\Delta t^\alpha$, the scheme is replaced by

$$\phi_j^{n+1} = \frac{1}{|\mathcal{N}_j|}\sum_{i\in\mathcal{N}_j} \mathcal{R}[\Phi^n](\vec{x}_i), \tag{7}$$

where $\mathcal{N}_j$ is the set of the first neighbour indexes of $j \in \mathcal{Q}$ and $|\mathcal{N}_j|$ represents its cardinality. In all computations presented in this work, we set $D = 10^{-3}$ and $\alpha = 1$.

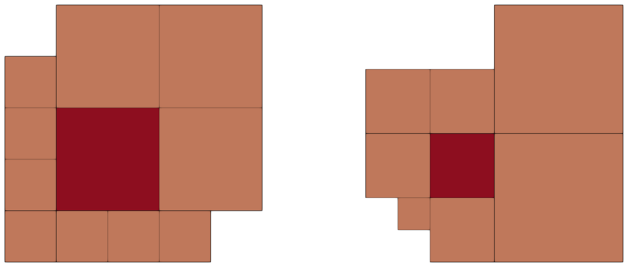

Fig. 1 Example of quadrants with different neighbours. The main quadrants and their neighbours are depicted in red and orange

Analogous computations can be done in order to obtain the 3D version of the scheme presented above. For its complete description we refer the reader to [25, 30].

Remark In order to apply the SL scheme (5), one has to quickly find the quadrant to which the points $\vec{x}^*_{j,i}$ belong. While on Cartesian grids localizing them is a trivial task, on quadtree this is no longer an easy procedure as it involves browsing the tree structure from the root until one finds the leaf containing the point $\vec{x}^*_{j,i}$.

In order to fully detail the scheme (5), a suitable reconstruction operator has to be defined. Considering a space of polynomials $\mathbb{P}$ with a basis $\mathcal{B}$, the key point of interpolation on quadtree is that, since each quadrant $j \in \mathcal{Q}$ may have a different number of neighbours $|\mathcal{N}_j|$, the linear system arising for each quadrant j may have a number of rows other than $|\mathcal{B}|$. Thus, in our scheme we will resort to a P1 constrained least-squares reconstruction $\mathcal{R}_{P1}[\Phi]$ and to a third order CWENO one $\mathcal{R}_{CW}[\Phi]$, based on the same constrained least-squares approach.

2.2 P1 Constrained Least-Squares Reconstruction

In 2D, on a quadrant j of size Δx, we consider the local basis $\mathcal{B}_j = \{\varphi_{j_s}\}_{s=0}^2 = \{1, \hat{x}, \hat{y}\}$, where $\hat{x} = (x - x_j)/\Delta x$ and $\hat{y} = (y - y_j)/\Delta x$. We seek for the vector of coefficients $\{c_{j_s}\}_{s=0}^2$ by imposing the constraint $c_{j_0} = \phi_j$ and the interpolation conditions

$$\sum_{s=1}^{2} c_{j_s} \varphi_{j_s}(\vec{x}_i) = \phi_i - \phi_j, \quad i \in \mathcal{N}_j, \tag{8}$$

such that $\mathcal{R}_{P1}[\Phi](\vec{x}_j) = \phi_j$ holds true.

With this choice one can immediately note that, considering both edge and corner neighbours (see Fig. 1), the case $|\mathcal{N}_j| < |\mathcal{B}_j| - 1$ never occurs. Instead, the system for the c_{j_s} is overdetermined and is solved using a least-squares approach.

In an analogous way, we can compute the P1 reconstruction for a quadrant j in the 3D case, choosing $\mathcal{B}_j = \{1, \hat{x}, \hat{y}, \hat{z}\}$.

Despite its low computational cost, we expect that the P1 operator might lead to less accurate reconstructed surfaces than alternative high-order ones. However, high-degree polynomials might introduce oscillations that severely affect the stability of the scheme. Essentially non-oscillatory techniques are apt for this purpose, producing high-order non-oscillatory operators by introducing a nonlinear dependency from the data. In particular, we resort to a CWENO operator which blends and selects first degree polynomials, having smaller and directionally biased stencils, with the optimal second degree one built on the entire set of neighbours of a quadrant and the quadrant itself, in order to get high-order accuracy. Due to the lack of structure in the adaptive mesh employed, both optimal and lateral polynomials will be obtained with the same constrained least-squares approach presented for the P1 case.

2.3 CWENO Constrained Least-Squares Reconstruction

Let $j \in \mathcal{Q}$ be the quadrant in which we want to compute the reconstruction. We recall now the general definition of the CWENO operator, along the lines of [54].

Definition 1 Given a stencil $\mathcal{S}_{\text{opt}}$, including j, let $P_{\text{opt}} \in \mathbb{P}_n^G$ be the optimal polynomial of degree G, associated to $\mathcal{S}_{\text{opt}}$. Further, let $P_1, P_2, ..., P_m$ be a set of $m \geq 1$ polynomials of degree g with $g < G$, associated to substencil $\mathcal{S}_k$ such that $\{j\} \subset \mathcal{S}_k \subset \mathcal{S}_{\text{opt}}\ \forall k = 1, ..., m$. Let also $\{d_k\}_{k=0}^m$ be a set of strictly positive real coefficients such that $\sum_{k=0}^m d_k = 1$.

The CWENO operator computes a reconstruction polynomial

$$\mathcal{R}_{CW} = \text{CWENO}(P_{\text{opt}}, P_1, ..., P_m) \in \mathbb{P}_n^G, \tag{9}$$

as follows:

1. introduce the polynomial $P_0 \in \mathbb{P}_n^G$ defined as

$$P_0(x) = \frac{1}{d_0}\left(P_{\text{opt}}(x) - \sum_{k=1}^{m} d_k P_k(x)\right); \tag{10}$$

2. compute suitable regularity indicators

$$\mathcal{I}_0 = \mathcal{I}[P_{\text{opt}}], \quad \mathcal{I}_k = \mathcal{I}[P_k], \quad k \geq 1; \tag{11}$$

3. compute the nonlinear coefficients $\{\omega_k^Z\}_{k=0}^m$ as

$$\alpha_k^Z = \frac{d_k}{(\mathcal{I}_k + \epsilon)^l}, \quad \omega_k^{CW} = \frac{\alpha_k^Z}{\sum_{i=0}^m \alpha_i^Z} \tag{12}$$

where ϵ is a small positive quantity and $l \geq 1$;

4. define the reconstruction polynomial as

$$\mathcal{R}_{CW}(x) = \sum_{k=0}^{m} \omega_k^{CW} P_k(x) \in \mathbb{P}_n^G. \tag{13}$$

Compared to the traditional WENO operator, the reconstruction polynomial defined in (13) can be evaluated at any point in the quadrant j at a very low computational cost. This has been proven to be an advantage in SL schemes (see [55]) that require to evaluate the reconstruction in different points laying in the same quadrant, which is, indeed, the case of our scheme (5).

The accuracy and non-oscillatory properties of the CWENO scheme is guaranteed by the dependence of its nonlinear weights (12) on the regularity indicators $\mathcal{I}_k$. On smooth data, the regularity indicators of all polynomials will be close to each others, so that $\mathcal{R} \approx P_{\text{opt}}$ and the reconstruction reaches the optimal order of accuracy $G+1$. On the other hand, when a discontinuity is present in $\mathcal{S}_{\text{opt}}$, both $\mathcal{I}_0 \asymp 1$ and at least one $\mathcal{I}_{\hat{k}} \asymp 1$ for some $\hat{k} \in \{1, ..., m\}$, where with $\asymp 1$ we mean "of the same order of 1, and bounded away from zero". In this case, the reconstruction is given by a linear combination of all lateral polynomials that are not affected by the discontinuity and its accuracy reduces to $g+1$.

2.3.1 Two Spatial Dimensions

Let $j \in \mathcal{Q}$ be the quadrant of the grid containing the reconstruction point x and let $\overline{\mathcal{N}}_j = \mathcal{N}_j \cup \{j\}$ be the stencil for the reconstruction, where $\mathcal{N}_j$ is the set of neighbouring quadrants of j. The setup of the stencils for optimal and lateral polynomials is the following: the optimal polynomial is the one associated to $\overline{\mathcal{N}}_j$, while the substencils for the south-west (sw), south-east (se), north-west (nw) and north-east (ne) polynomials are composed by j itself and corner or edge neighbours in the corresponding directions.

The reconstruction operator compute $\mathcal{R}_{CW} \in \mathbb{P}_2^2$ as

$$\mathcal{R}_{CW} = \text{CWENO}(P_{opt}^{(2)}; P_{\text{sw}}^{(1)}, P_{\text{se}}^{(1)}, P_{\text{nw}}^{(1)}, P_{\text{ne}}^{(1)}), \tag{14}$$

with the optimal and lateral polynomials set to be second and first degree polynomials, respectively, built on the local basis

$$\mathcal{B}_2 = \{1, \hat{x}, \hat{y}, \hat{x}^2, \hat{x}\hat{y}, \hat{y}^2\}, \quad \mathcal{B}_1 = \{1, \hat{x}, \hat{y}\}. \tag{15}$$

For each polynomial we impose the constraint $P(\vec{x}_j) = \phi_j$, such that $c_{j_0} = \phi_j$, and compute the remaining coefficients c_{j_k} similarly to (8), following a least-squares approach. Since we impose the constraint on each polynomial, we can guarantee that the condition $\mathcal{R}_{CW}[\Phi](\vec{x}_j) = \phi_j$ holds true.

Once the coefficients are obtained, the oscillation indicators are defined as

$$\mathcal{I}[P] = \sum_{|\vec{\alpha}|\geq 1} \Delta x_j^{2|\vec{\alpha}|-2} \int_{Q_j} \left[\frac{\partial P}{\partial x^{\alpha_1} \partial y^{\alpha_2}}\right]^2 \mathrm{d}x\mathrm{d}y, \tag{16}$$

according to the classical ones of [56, 57]. This choice might be different to the usual one for HJ equation, whose solutions can be at worst continuous with kinks, see [58, 59], but here is motivated by the degrees of the polynomials employed in the reconstruction. In fact, since our reconstruction employs lateral polynomials of at most first degree, their HJ-type indicators would be all identically equal to zero, providing no information about their regularity and thus the ones to be discarded in case of oscillations, consequently affecting the non-oscillatory properties of the entire reconstruction procedure.

Denoting with $\vec{c}$ the vector of coefficients of a generic polynomial in two variables of degree up to 2, along the basis $\mathcal{B}_2$ in (15), its oscillation indicator can be expressed as

$$\mathcal{I}[P] = \vec{c}^{\,T} M \vec{c}, \tag{17}$$

with

$$M = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 13/3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 7/6 & 0 \\ 0 & 0 & 0 & 0 & 0 & 13/3 \end{pmatrix}. \tag{18}$$

The CWENO reconstruction applied in the numerical tests of this paper is then defined by choosing linear coefficients $d_{\text{sw}} = d_{\text{se}} = d_{\text{nw}} = d_{\text{ne}} = 1/8$ and $d_0 = 3/4$, $l = 2$ and $\epsilon = \Delta x^2$ in the above construction.

2.3.2 Three Spatial Dimensions

The construction for the CWENO operator in three dimensions strictly follows the guidelines seen above for the two-dimensional case.

The reconstruction operator blends the second degree optimal polynomial associated to the stencil $\overline{\mathcal{N}}_j$ with 8 first degree polynomials corresponding to the biased ones. To collect these stencils we consider west, east, north, south, backward and forward directions. The result of the reconstruction operator is $\mathcal{R}_{CW} \in \mathbb{P}_3^2$, built on the basis

$$\begin{aligned} \mathcal{B}_2 = \{1, \hat{x}, \hat{y}, \hat{z}, \hat{x}^2, \hat{x}\hat{y}, \hat{y}^2, \hat{z}^2, \hat{x}\hat{z}, \hat{y}\hat{z}\}, \\ \mathcal{B}_1 = \{1, \hat{x}, \hat{y}, \hat{z}\}. \end{aligned} \tag{19}$$

Due to the dimension, the oscillation indicators need to be properly rescaled and are thus defined as

$$\mathcal{I}[P] = \sum_{|\vec{\alpha}|\geq 1} \Delta x_j^{2|\vec{\alpha}|-3} \int_{Q_j} \left[\frac{\partial P}{\partial x^{\alpha_1} \partial y^{\alpha_2}}\right]^2 \mathrm{d}x\mathrm{d}y\mathrm{d}z, \tag{20}$$

or in a matrix form as

$$\mathcal{I}[P] = \vec{c}^{\,T} M \vec{c}, \tag{21}$$

with

$$M = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 13/3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 7/6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 13/3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 13/3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 7/6 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 7/6 \end{pmatrix}. \tag{22}$$

Remark The scaling factor in (16) and (20) is motivated by the expected behaviour of the indicators. To clarify the idea, let us consider the simpler case of a generic polynomial P_k of degree $r-1$, in one dimension, having a discontinuity in its stencil S_k. In this case, the indicator would read

$$\mathcal{I}[P_k] = \sum_{\alpha \geq 1} \Delta x_j^{2\alpha-1} \int_{Q_j} \left[\frac{\partial P}{\partial x^\alpha} \right]^2 \mathrm{d}x. \tag{23}$$

The α-derivative of P_k is a polynomial of degree $r-\alpha-1$, whose coefficients of degree m are given by the Newton divided differences with $m+\alpha+1$ points, which scale as $\mathcal{O}(\Delta x^{-m-\alpha})$, due to the discontinuity. Computing the integral in (23) contributes another Δx factor and the indicator is thus a sum of terms, all scaling at the same rate $\mathcal{O}(\Delta x^{1-2\alpha})$. It is thus clear that the exponent $2\alpha-1$ in the scaling factor yields $\mathcal{I}_k \asymp 1$ in this non-regular case; on the other hand, in case of smoothness, the Newton divided differences would be $\mathcal{O}(1)$ and one gets $\mathcal{I}_k \to 0$, as expected.

With similar arguments one can deduce the correct scaling factor in the definition of the indicator in the general multidimensional case.

3 Computation of the Level Set Evolution

The overall surface reconstruction procedure requires different schemes to be put together. The SL scheme presented in the previous section constitutes the main ingredient of the method; all the other auxiliary parts will be detailed later on in this section. A summary of the complete algorithm will be given at the beginning of Sect. 4.

3.1 Localization

A typical approach when dealing with LSM is to localize the evolution only in a narrow band around the zero level set of ϕ. Along the lines of [49], the definition of the narrow band, and related computations, is based on the two constants

$$\beta = 2\lambda \Delta x_{\min}, \tag{24}$$

and

$$\gamma = 4\lambda \Delta x_{\min}, \tag{25}$$

where $\Delta x_{\min}$ is the minimum edge length of the refined grid, as defined by (4), and $\lambda = \Delta t / \Delta x_{\min}$ is related to the choice of the time step $\Delta t = \mathcal{O}(\Delta x_{\min})$.

Since it is not important to update ϕ far away from zero, at each time step we choose a subgrid

$$\widetilde{\mathcal{Q}} = \{ j \in \mathcal{Q} : \left| \phi_j^n \right| < \gamma \} \subset \mathcal{Q}, \tag{26}$$

according to [49] and to the stencils for the spatial reconstructions employed in this work, and update ϕ^{n+1} only therein. Outside $\widetilde{\mathcal{Q}}$, we simply cut our level set function as

$$\phi_j = \begin{cases} \gamma & \text{if } \phi_j > \gamma, \\ \phi_j & \text{if } \left| \phi_j \right| \leq \gamma, \\ -\gamma & \text{if } \phi_j < -\gamma. \end{cases} \tag{27}$$

Analogously to [49], to prevent numerical oscillations at the boundary of $\widetilde{\mathcal{Q}}$, we update the solution involving the additional cutoff function

$$c(\phi) = \begin{cases} 1 & \text{if } |\phi| \leq \beta, \\ \frac{(|\phi|-\gamma)^2 (2|\phi|+\gamma-3\beta)}{(\gamma-\beta)^3} & \text{if } \beta < |\phi| \leq \gamma, \\ 0 & \text{if } |\phi| > \gamma, \end{cases} \tag{28}$$

considering the modified equation

$$\begin{aligned} \phi_t(\vec{x},t) = c(\phi) \left[\frac{d(\vec{x})}{E_p(\phi)} \right]^{p-1} \Big(& \nabla d(\vec{x}) \cdot \nabla \phi(\vec{x},t) \\ & + \frac{\mu}{p} d(\vec{x}) \nabla \cdot \left(\frac{\nabla \phi(\vec{x},t)}{|\nabla \phi(\vec{x},t)|} \right) |\nabla \phi(\vec{x},t)| \Big) \end{aligned} \tag{29}$$

and the proper modifications to the semi-Lagrangian scheme (see [25]).

Finally, even if we consider the modified equation (29), keeping the level set function close to the signed distance one during its evolution constitutes an important issue and a reinitialization procedure will be required. This reinitialization procedure must be performed on a larger narrow band $\overline{\mathcal{Q}}$ obtained by considering $\widetilde{\mathcal{Q}}$ and a number of neighbouring layers proportional to λ. The fact that $\overline{\mathcal{Q}} \supset \widetilde{\mathcal{Q}}$ is essential if one starts the algorithm with an initial data so far from the point cloud that the initial computational band (26) does not contain $\mathcal{S}$. In such a case, without the enlarged reinitialization band, the evolution of ϕ would remain confined to the first computational band $\widetilde{\mathcal{Q}}^0$, while with this choice, the successive bands $\widetilde{\mathcal{Q}}^n$ will be able to move contextually with the zero level set of ϕ^n.

3.2 Adaptivity

We consider an adaptive framework in which the computational domain is subdivided into quadrants (resp. octants). Each of them can be refined by splitting into 4 (resp. 8) equal pieces, each group of them can be coarsened and replaced by the quadrant (octant) they were obtained from. Each quadrant has a level $l \geq 0$, which dictates its edge length as being $1/2^l$ that of the computational domain. The minimum and maximum level control the size of the largest and smallest cell in the grid. Typically one also enforces that the levels of two adjacent quadrants do not differ by more than 1. The grids can be represented by a quadtree (resp. octree), in which any refined quadrant is a node with 4 (resp. 8) children and the active computational quadrants are the leaves of the tree.

In what follows, unless otherwise specified, we will use the terms cell, quadrant, octant and leaf, to refer to the elements of the computational grid.

In the AMR context the typical adaptivity procedure follows the steps below:

- first of all, the leaves are marked for refinement, coarsening or to be left unchanged, depending on the chosen criteria;
- the refinement and coarsening is then applied to each marked leaf in a recursive way and consistently with the minimum and maximum level of refinement set for the mesh;
- if the grid is graded, once it has been roughly adapted, a 2:1 balancing is performed in order to guarantee that the level difference between a quadrant and each of its neighbours is at most 1;
- finally, as far as parallelization is concerned, load distribution is operated between processes by an equal division of the new array of leaves.

In our level set application solid adaptivity criteria need to be defined to ensure the grid to be properly refined in the vicinity of the zero level set, i.e. the surface interface Γ, and in the proximity of the cloud data $\mathcal{S}$. A progressive decreasing resolution is expected when moving further away, in particular where our level set function ϕ is flattened to a prescribed value $\pm\gamma$, according to the localization of the method, described in Sect. 3.1.

3.2.1 Refinement

Let us indicate with $h_{\mathcal{S}}$ a suitable estimation of the resolution of the point cloud $\mathcal{S}$, i.e. the average distance between two closest points in $\mathcal{S}$, and let us recall that l and L denote the current level of a quadrant and the maximum level of refinement, respectively. In order to refine the computational grid we proceed as follows:

- a quadrant $j \in \mathcal{Q}$ is refined up to level L if

$$|\phi_j| < \gamma \quad \text{and} \quad d < 2h_{\mathcal{S}}; \tag{30}$$

- a quadrant $j \in \mathcal{Q}$ is refined up to level $L-1$ if

$$|\phi_j| < \gamma \quad \text{and} \quad 2h_{\mathcal{S}} \leq d < 4h_{\mathcal{S}}; \tag{31}$$

- a quadrant $j \in \mathcal{Q}$ is refined up to level $L-2$ if

$$|\phi_j| < \gamma; \tag{32}$$

where γ is defined as in Sect. 3.1.

Once the quadrant j is marked, the procedure for the refinement is trivial thanks to the local reconstruction $\mathcal{R}_j$ defined on the quadrant itself. The quadrant j of centre $\vec{x}_j$ originates 4 children in 2D, 8 in 3D, by dividing by two each of its edges. The new quadrants j_r have centres $\vec{x}_{j_r}$ and their corresponding level set values are $\phi_{j_r} = \mathcal{R}_j[\Phi](\vec{x}_{j_r})$.

3.2.2 Coarsening

On the other side, a set of 4 in 2D (resp. 8 in 3D) children $C \subset \mathcal{Q}$ is coarsened if

$$|\{c \in C \,|\, |\phi_c| < \gamma\}| = 0, \tag{33}$$

where γ is again defined as in Sect. 3.1.

The coarsening procedure consists in gluing together quadrants of the same level originated from the same branch: the new centre $\vec{x}_j$ and the corresponding level set value is obtained by averages from the outgoing quadrants of centres $\{\vec{x}_c\}_{c\in C}$, namely

$$\vec{x}_j = \frac{1}{|C|}\sum_{c\in C}\vec{x}_c \quad \text{and} \quad \phi_j = \frac{1}{|C|}\sum_{c\in C}\phi_c. \tag{34}$$

3.3 Distance Function Computation

The evolution of the level set function ϕ requires, as a fundamental step, to compute the distance function d from the point cloud $\mathcal{S}$, which we recall to be given by $d(\vec{x}) = \min_{\vec{q}\in\mathcal{S}} |\vec{x} - \vec{q}|$. The distance function appears itself in the governing equation (2), in the definition of the energy functional, and its gradient constitutes the velocity field driving the evolution of the level set towards the data. Clearly, one is not interested in computing the exact values of d on the whole grid, but just in the vicinity of the cloud.

Thus, in a first step, d_j can be computed exactly in the quadrants that contain at least a point of $\mathcal{S}$ as the minimum distance of the centre $\vec{x}_j$ from the cloud points located in the quadrant j. Then, this information can be spread across the

whole grid, all over the processors, by solving the Eikonal equation

$$|\nabla d(\vec{x})| = 1, \quad d(\vec{q} \in \mathcal{S}) = 0. \tag{35}$$

The most popular algorithms for solving (35) are the Fast Marching Method (FMM) [60–62] and the Fast Sweeping Method (FSM) [23, 63]. However, in our framework, these methods should be applied carefully since they might suffer both from the adaptive discretization of the grid and the parallelization of the scheme, due to the causality across processes. Some of the earliest attempts in parallelizing the FMM were proposed in [64, 65], where one could notice how the number of iterations needed to get convergence greatly depends on the complexity of the interface and on the parallel partitioning and, in general, fewer iterations are required if the domains are aligned with the normals to the interface. In [66] a parallel FSM method was presented for the first time, while a hybrid FMM-FSM was presented in [67]. A parallel Fast Iterative Method (FIM) has been also proposed in [68]. The case of unstructured meshes has been investigated in [69, 70].

Due to the complexity of the aforementioned methods on AMR grids, here we prefer to compute the exact values of d on the entire grid, starting from the set of cells containing the points of $\mathcal{S}$ and then proceeding by layers of neighbours, as described below in the so-called *propagation* procedure (see Sect. 3.5).

3.4 Reinitialization

A delicate issue when dealing with LSM is to guarantee that the level set function ϕ stays well-behaved during its evolution, hopefully close to a signed distance function, so that

$$|\nabla \phi| \approx 1. \tag{36}$$

This is fundamental in order to get a watertight representation of the evolving surface and to exploit its geometrical properties in a reliable and stable way, since $|\nabla \phi|$ is well-defined and further away from zero. Also, for PDE models to be solved in $\Omega = \{\phi \leq 0\}$, it is often crucial to be able to compute the normals to Γ (i.e. by evaluating the gradient of ϕ), at least close to the surface.

In this regard, even if we choose as initial data a signed distance function, the evolution governed by (2) will produce steep gradients in the solution and a so-called *reinitialization* procedure needs to be performed. This constitutes a classical ingredient when dealing with LSM even if we point out that some alternatives to avoid the reinitialization step can be found in literature (see for example [71–74]).

The most common way to reinitialize a level set function is by solving the steady-state HJ-type equation introduced in [75] given by

$$\begin{cases} \phi_\tau + S(\tilde{\phi})(|\nabla \phi| - 1) = 0, \\ \phi(\vec{x}, 0) = \phi_0(\vec{x}) = \tilde{\phi}(\vec{x}, t), \end{cases} \tag{37}$$

where $\tilde{\phi}$ represents the possibly perturbed level set function to be reinitialized. In [76] the authors solve the above equation following the explicit finite differences scheme described in [77], to get a second-order accurate LSM on non-graded adaptive Cartesian grids.

Here instead we refer to the work of Saye [50] in which the author proposes an efficient method for calculating high-order approximations of closest points on implicit surfaces that can be applied also on a general unstructured grid and in any number of spatial dimensions.

Actually, the procedure introduced in [50] is applied just in the vicinity of the zero level set of ϕ and then, once we have reinitialized ϕ in these quadrants, we propagate the information all over $\mathcal{Q}$, as described in Sect. 3.5.

Let us consider a fixed time n of the evolution of the level set ϕ for which we have obtained the updated level set function $\tilde{\phi}$ that needs to be reinitialized. Along the lines of [50], we start by detecting the quadrants that contain the zero level set of $\tilde{\phi}$, namely the set $\mathcal{Q}_0 \subset \mathcal{Q}$, as in Sect. 3.1. For each quadrant $j \in \mathcal{Q}_0$, a high-order closest point algorithm via Newton's method is applied as follows:

- for each quadrant j we create 4 points $\vec{x}_{j_s}$ in 2D (resp. 8 points in 3D) located at the centres of a subgrid 2×2 (resp $2 \times 2 \times 2$) of the quadrant j;
- each point $\vec{x}_{j_s}$ is projected onto the zero level set of the polynomial $\mathcal{R}_j[\tilde{\Phi}]$, namely the reconstruction defined on the quadrant j from the updated values $\{\tilde{\phi}_j\}_{j \in \mathcal{Q}}$;
- for each quadrant $j \in \mathcal{Q}_0$ we find its closest point $\vec{x}_j^*$ in $X = \{\vec{x}_{j_s}\}_{j \in \mathcal{Q}_0}$;
- the point $\vec{x}_j^* \in X$ is then used as the initial guess to compute the minimum distance of $\vec{x}_j$ from the zero level set of $\tilde{\phi}$ via a Newton's method that relies on the reconstruction associated to the point $\vec{x}_j^*$;
- finally, for each $\vec{x}_j$, the reinitialized value ϕ_j is set equal to this minimum distance, multiplied by the sign of $\tilde{\phi}_j$, namely preserving the sign of the level set function before the reinitialization.

In practice, the method described in [50] is based on a two steps procedure to compute the closest point to the interface $\tilde{\phi} = 0$ from a point $\vec{x}_j$ located in its vicinity, considering the zero level set of the polynomials $\mathcal{R}_j$. In what follows, the points $\vec{x}_j^* \in X$ are referred to as *seed points*.

3.5 Propagation

Let us consider a subset $\mathcal{D}^0 \subset \mathcal{Q}$, on whose quadrants a certain quantity g^0 to be minimized is defined. In our specific

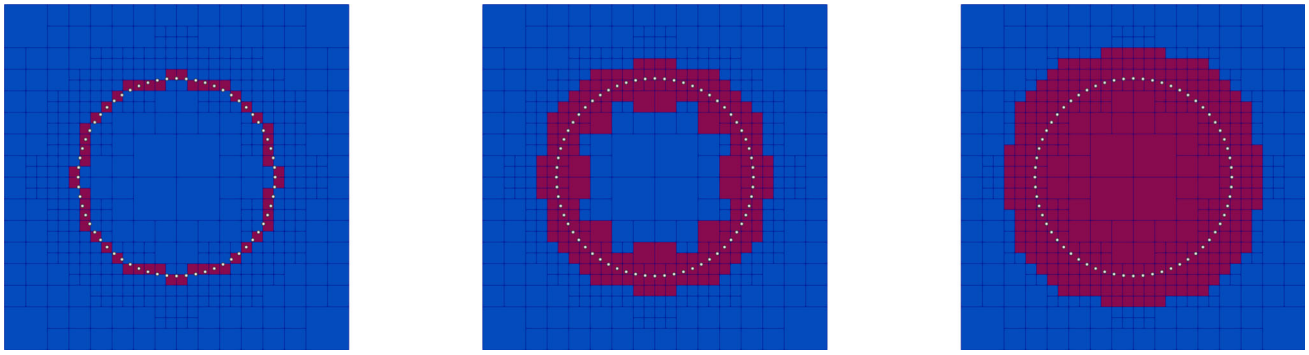

Fig. 2 Propagation of the distance function. Left: the initialized quadrants before the propagation step are the one containing one or some of the points in $\mathcal{S}$. Center: quadrants with updated distance after one step of propagation. Right: quadrants with updated distance after two steps of propagation

case g^0 is the distance (resp. signed distance) of the quadrant centre $\vec{x}$ from a point $\vec{q}$, which is a member of $\mathcal{S}$ (resp. a reinitialization seed point). We store this reference point as $\vec{q}_j$, attached to the quadrant j.

At the beginning of the propagation, data of quadrants in $\mathcal{Q} \setminus \mathcal{D}^0$ are initialized to a high enough value. At the $m+1$-th iteration, let $\mathcal{D}^m$ be the set of quadrants on which the value of g has changed in the previous iteration, and g^m be the current update of the function g on the set $\cup_{k=0}^{m} \mathcal{D}^k$. We will compute g^{m+1} and $\mathcal{D}^{m+1}$ as follows:

- for each $j \in \mathcal{D}^m$ we consider $\overline{\mathcal{N}}_j^m$ the set of j itself and its first neighbours, and we define the set

$$\mathcal{N}^m := \bigcup_{j \in \mathcal{D}^m} \overline{\mathcal{N}}_j^m;$$

  note that in general $\mathcal{N}^m \cap \mathcal{D}^m \neq \emptyset$, as we may need to further minimize the values updated in the previous propagation step, too;
- then, for each quadrant $i \in \mathcal{N}^m$ we compute the temporary value $\tilde{g}_i^{m+1}$ as

$$\tilde{g}_i^{m+1} = \min_{k \in \mathcal{D}^m} |\vec{x}_i - \vec{q}_k|, \tag{38}$$

  where the point $\vec{q}_k$ is associated to $\vec{x}_k$;
- we define the set of updated quadrants as

$$\mathcal{D}^{m+1} = \{i \in \mathcal{N}^m : \tilde{g}_i^{m+1} < g_i^m\} \tag{39}$$

  and set

$$g_i^{m+1} = \begin{cases} \tilde{g}_i^{m+1} & \text{if } \ i \in \mathcal{D}^{m+1}, \\ g_i^m & \text{elsewhere;} \end{cases} \tag{40}$$

- if $\mathcal{D}^{m+1} \neq \emptyset$ we iterate, otherwise we stop.

The procedure described above works exactly as it has been described, for the propagation of the distance function from the point cloud $\mathcal{S}$. The quadrants containing the cloud points are initialized and then their information is propagated, including the already initialized quadrants in the propagation procedure. In Fig. 2 three different phases of the propagation are shown ($m = 0, 1, 2$). Red quadrants are the ones that have been initialized with the distance to a point $\vec{q} \in \mathcal{S}$, if $m = 0$, or updated lowering their value of the distance to $\mathcal{S}$, for $m = 1, 2$; the blue ones are the ones in which the information has not arrived yet.

Regarding reinitialization, the procedure is done analogously, but in the definition of $\mathcal{N}^m$, $m \geq 0$, we exclude the quadrants in $\mathcal{D}^0$ since we do not want their values to be changed by the propagation process. Also, the condition in (39) and (40) are modified respectively as

$$\mathcal{D}^{m+1} = \{i \in \mathcal{N}^m : \left|\tilde{g}_i^{m+1}\right| < \left|g_i^m\right|\} \tag{41}$$

and

$$g_i^{m+1} = \begin{cases} S(g_i^0)\, \tilde{g}_i^{m+1} & \text{if } \ i \in \mathcal{D}^{m+1}, \\ g_i^m & \text{elsewhere,} \end{cases} \tag{42}$$

where S is the sign function.

Remark Of course, the propagation algorithm must be implemented within the parallel setup of the rest of the code. This requires to iterate the scheme above more than one time in order to allow the information to reach all the processors, namely the whole computational domain. Once the local set $\mathcal{D}^m$ is empty, local boundary quadrants spread their information to the halo region in order to minimize the function g in ghost quadrants. At this point, a communication occurs. Each processor collects possible minimizing values from its neighbours and checks if the function g should be actually updated in its local boundary quadrants. If so, the set $\mathcal{D}^m$ would be filled by the updated quadrants, being not any more

empty, causing the local propagation iterations to start again. Otherwise, if there are no local updates, the corresponding processor will wait for the others to finish the current round of local propagation. When all the $\mathcal{D}^m$ remain empty, the overall propagation procedure stops.

3.6 Energy Functional

To apply the SL scheme (5) with $p > 1$ we finally need to approximate, at each time step, the value of the energy functional $E_p(\phi)$ defined in (3).

We approximate the Dirac δ function by restricting the integration domain to the subset

$$\mathcal{Q}_0 \coloneqq \{j \in \mathcal{Q} : \left|\{i \in \mathcal{N}_j : \phi_i \phi_j \leq 0\}\right| > 0\}, \tag{43}$$

namely the set composed by the cells in $\mathcal{Q}$ where, at a specific time step, the front is located. Further, thanks to the reinitialization procedure, we assume that the function ϕ is, at least locally around $\mathcal{Q}_0$, a signed distance so that $|\nabla\phi| = 1$.

We thus approximate the energy as

$$E_p(\phi) \approx \left(\sum_{j \in \mathcal{Q}_0} |d_j|^p (\Delta x_{\min})^{n-1} \right)^{1/p}, \tag{44}$$

where n in (44) states for the dimension of the problem and we assume that the refinement provides the maximum resolution for the quadrants in $\mathcal{Q}_0$.

4 Numerical Tests

In this section, we present some representative numerical results in 2D and 3D, showing different features of the reconstruction process. Point clouds data will be both synthetic and non-synthetic ones coming from laser scanning of real objects.

The overall computation of the level set reconstruction is summarized in Algorithm 1. In a preliminary phase, one creates the computational grid, computes the distance function, sets the initial data, and adapts the grid to the first zero level set. Then, the evolution starts and the level set function is evolved until a final stopping criterion is satisfied. Differently from [25], thanks to the fully adaptive framework, Algorithm 1 is here performed just once, setting the desired maximum level of refinement L from the beginning. Once convergence is achieved, some parameters and the reconstruction operator are changed in order to perform five last regularizing iterations to smoothen the final data. Moreover, in these final steps the adaptivity criterion is only based on the level set function ϕ, and not on the distance function.

Algorithm 1 Adaptive level set reconstruction from a point cloud $\mathcal{S}$

1. Define the computational domain Ω' and create a quadtree/octree grid $\mathcal{Q}$ encompassing the dataset $\mathcal{S}$.
2. Compute the distance function $d(\vec{x})$ at all grid points, see § 3.3.
3. Set initial data $\{\phi_j^0\}_{\vec{x}_j \in \mathcal{Q}}$ enclosing $\mathcal{S}$.
4. Cut and adapt using ϕ^0 according to § 3.1 and § 3.2, respectively.
5. Loop:
   (a) choose computational subgrid $\widetilde{\mathcal{Q}} \subset \mathcal{Q}$ as in § 3.1;
   (b) compute the energy functional (3) as in § 3.6;
   (c) compute $\{\phi_j^{n+1}\}_{j \in \widetilde{\mathcal{Q}}}$ using the scheme defined in § 2 coupled with the reconstruction operator from § 2.2;
   (d) reinitialize $\{\phi_j^{n+1}\}_{j \in \mathcal{Q}}$ as in § 3.4 and cut according to (27);
   (e) adapt using ϕ^{n+1} according to § 3.2.
6. The previous loop is repeated for five iterations with the reconstruction operator from § 2.3 and different values of the parameters.

The datasets used for the tests were made available in the Digital Shape WorkBench of the AIM@SHAPE and VISIONAIR projects [78], in the 3D Scanning Repository of the Stanford University [79] or by the work of Hoppe et al. [19]. All the codes for the simulations have been implemented in C++ using the P4EST library [80] for grid management and MPI parallelization. The 3D tests have been performed on the cluster Galileo 100 hosted at CINECA,[1] exploiting the resources assigned to ISCRA-C Projects[2].

Spatio-temporal discretization To define the computational domain Ω' we first rescaled the cloud $\mathcal{S}$ in a box $[-1, 1]^n$ and then set Ω' as an enlarged cube $[-M, M]^n$, $M > 1$ that encompasses all the points in $\mathcal{S}$. The constant M and the maximum level of refinement L are computed in order to have

$$\Delta x_{\min} = \frac{\Delta\Omega'}{2^L} \approx C_{\mathcal{S}} h_{\mathcal{S}}, \tag{45}$$

where $h_{\mathcal{S}}$ denotes an estimate of the resolution of the cloud $\mathcal{S}$ and $C_{\mathcal{S}}$ is a constant set by the user. The value of $h_{\mathcal{S}}$ is approximated by randomly choosing a sample made up of the 10% of the points in $\mathcal{S}$ and then computing the average of their distances from the cloud itself. Regarding $C_{\mathcal{S}}$, having in mind the further applications to PDE models, we usually choose values between 0.25 and 0.5 in order to get a final reconstruction defined on a finer grid, compared to the cloud resolution.

Concerning the time step, for each quadrant $j \in \mathcal{Q}$, we set

$$\Delta t_j = \frac{3}{2} \Delta x_j, \tag{46}$$

[1] https://www.hpc.cineca.it/systems/hardware/galileo100/

[2] *Adaptive Mesh Refinement in Level Set Methods* (HP10C7HWOL) *High-order schemes on Adaptive Meshes* (HP10CO8NC7)

which clearly states the advantages of using the SL approach: to update the solution we are allowed to set $\Delta t = \mathcal{O}(\Delta x)$, not being prohibitively constrained by the parabolic term. In (46), note that Δx_j corresponds to the edge length of the actual active quadrant j.

Initial data Regarding the initial data, differently from [23], this is simply set to be a signed distance function associated to a circle or a sphere containing all the data points. While this choice has the effect of slowing down the reconstruction process, it reduces the dependence on other parameters (see [25]), and prevents the risks arising from not evenly distributed datasets. In fact, especially in 3D, it is common to find datasets that present fake holes due to the supports used during the laser scanning phase. These missing data could be misleading and need to be properly distinguished from actual topological features. That is why, exploiting the potential of adaptivity, in this work, we totally avoid these kind of issues choosing the simplest geometry and topology for the initial data. Moreover, in Sect. 4.8, we propose a modification to the level set equation (2) in order to capture diverse topologies.

Stopping criterion To stop the evolution, as we are looking for a minimum of the energy functional (1), we resort to the following criteria: at time n, the algorithm stops if

$$\Delta_E^n = \frac{|\bar{e}_{n-1}^k - \bar{e}_n^k|}{\bar{e}_n^k} < 10^{-4}, \tag{47}$$

where

$$\bar{e}_n^k = \frac{1}{k} \sum_{i=n-k+1}^{n} E_2(\phi^i) \tag{48}$$

or after a maximum of 100 iterations. Note that in (47), the energy functional (1) is computed with $p = 2$ and we force the algorithm to do at least 10 iterations by setting $k = \min(n, 10)$. Condition (47) is the one proposed in [31] and is in practice a way to detect stationary points or flat areas of the energy functional. The evaluation of $|\phi(\vec{q})|$, for any $\vec{q} \in \mathcal{S}$, or $|\phi_t|$ could constitute possible alternatives, as suggested in [23, 81].

Parameters setting The parameters p and μ appearing in (2) need to be chosen. According to [25], we set them initially as

$$p = 1 \quad \text{and} \quad \mu \in [0.05, 0.2], \tag{49}$$

and perform the loop 5 in Algorithm 1 until convergence is achieved, namely when (47) is satisfied. Then, we update them as

$$p = 2 \quad \text{and} \quad \mu = 1, \tag{50}$$

and perform five more iterations of the loop 6 in Algorithm 1. We recall that $p = 1$ grants for a faster evolution and it thus useful given our choice for the initial data, while $p = 2$ is more effective in reaching a steady state for the evolution. Moreover, setting $\mu = 0$ simplifies the update formulas (5) by nullifying the parabolic term while also preventing from losing too much details, especially on coarse grids, at a price of having a more rugged surface as final result. On the other hand, higher values are suggested to smoothen the solution and to handle possible changes of topology. Unless otherwise specified we initially choose $\mu = 0.2$. While in [25, 81] smaller values of μ were strongly required to handle the coarser Cartesian grids, they are no longer needed now. In the final iterations, we set $\mu = 1$ in order to reproduce the original model and to regularize the final reconstruction. An example of the effect of the parameter μ will be given in Sect.4.4.

Reconstruction operator Following the same spirit of parameter selection, we choose different reconstruction operators for the loops 5 and 6 in Algorithm 1. In particular, we expect the P1 operator to be cheaper, less accurate, but closer to a polygonal reconstruction faithfull to the points in $\mathcal{S}$. On the other side, a high-order reconstruction should provide more accurate and regular results, being also capable to reproduce complex features of the shape. According to these considerations and to our focus on PDE computations, referring to Algorithm 1, we use the P1 operator of Sect. 2.2 in loop 5, while we switch to the CWENO one of Sect. 2.3 for the regularizing iterations of loop 6, to get a smoother final result. More details about this choice will be give in the square test in Sect. 4.1.

Results evaluation Finally, in order to evaluate the quality of the reconstruction we compute, alongside (47), the L^1-norm of the error Err_1^n, when the exact signed distance function ϕ^* is given, and the average of the error on the points of the cloud

$$Err_{\mathcal{S}}^n = \frac{\sum_{\vec{q} \in \mathcal{S}} |\mathcal{R}[\Phi^n](\vec{q})|}{|\mathcal{S}|}, \tag{51}$$

to evaluate how much the final reconstruction is attached to the data.

4.1 Square 2D Test

We first consider the benchmark case of a square sampled by a point cloud $\mathcal{S}$ made of 24 points whose resolution is $h_{\mathcal{S}} \approx 0.23$ and for which the exact signed distance function ϕ^* is known. The initial data and the final reconstruction, obtained with the setting described above ($C_{\mathcal{S}} = 0.125$, $\Delta x_{\min} = 0.03$ and initial value $\mu = 0.05$) are depicted in Fig. 3, while in Table 1 we show a comparison between strate-

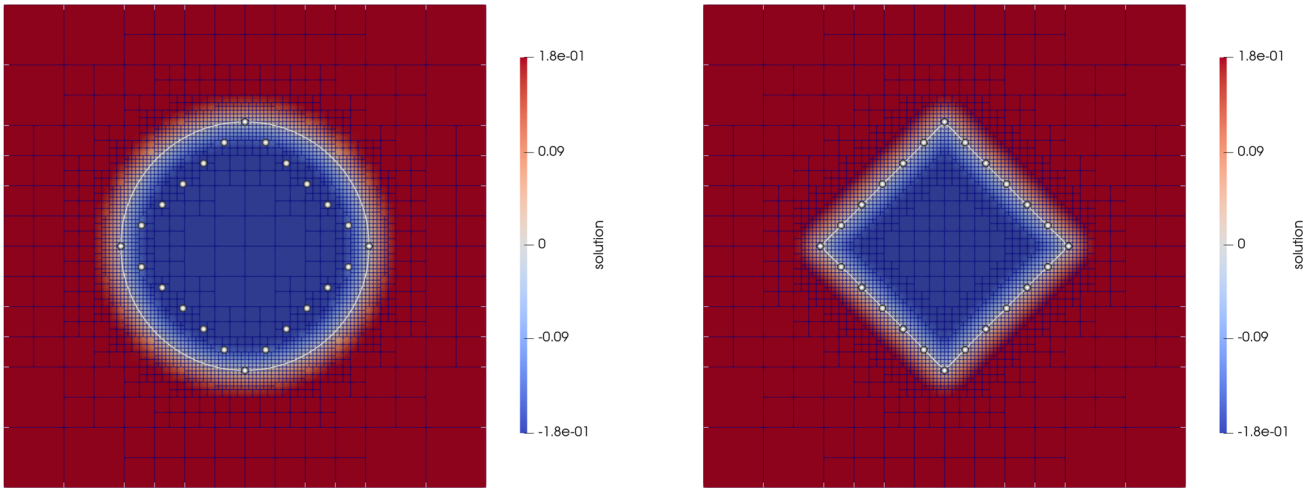

Fig. 3 2D square test: from left to right, the initial data and the final reconstruction. The zero isocontour of the level set function ϕ is always represented with a white line

Table 1 Comparison between the use of different reconstruction operators. The P1+CW strategy shows the best result in terms of errors versus computational time

	L1-error	Cloud error	CPU time (secs)
P1	9.57×10^{-3}	4.21×10^{-3}	2.62
CWENO	7.52×10^{-3}	4.66×10^{-3}	15.52
P1 + CWENO	8.63×10^{-3}	5.33×10^{-3}	4.30

gies that employ different reconstruction operators: only P1, only CWENO and the usual setup P1+CWENO, taking respectively 39, 38 and 39 iterations to get the final result. One can notice that mixing the two operators, P1 and CWENO, gives the best result in terms of accuracy versus computational time, even if we pay something in terms of fidelity to the dataset $\mathcal{S}$. Similar considerations can be deduced from Fig. 4 where some isocontours of the final level set are depicted. We point out that using only the P1 operator the isocontours are less orthogonal around the vertices. On the other side, resorting to the CWENO reconstruction leads to a more accurate result, but Fig. 4 shows that there is no significant gain in applying it from the beginning.

4.2 Heart 2D Test

In the second test we consider a cloud of 24 points representing a heart and having a resolution approximately equal to 2.13×10^{-1}. Starting with $\Delta x_{\min} = 3.24 \times 10^{-2}$, the reconstruction takes 43 iterations and is capable to capture both the tips of the heart and its rounded parts. The initial data and the final reconstruction, with their zero isocontour, are shown in Fig. 5.

4.3 Teapot 2D Test

Here we consider a cloud of 189 points representing a teapot that presents different features, including rounded parts and fine details. In Fig. 6 we present three snapshots of the evolution of the level set and of the grid. One can appreciate that in the adaptive framework the resolution of

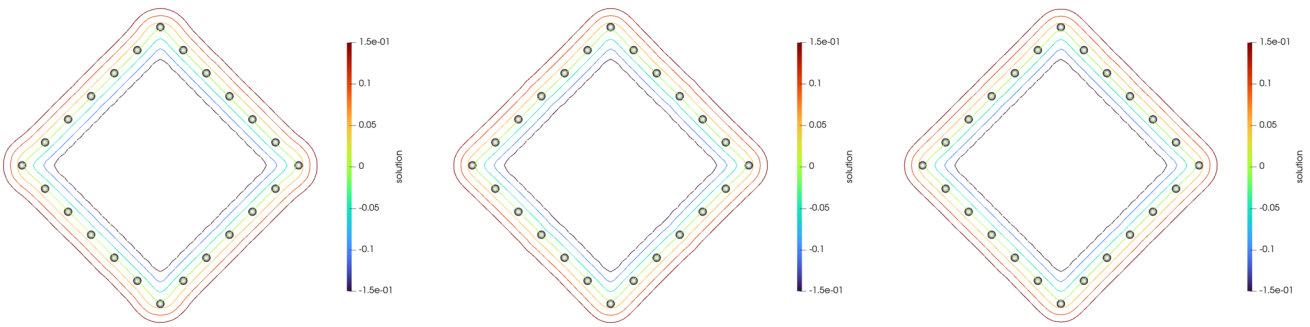

Fig. 4 Final isocontours obtained by setting different reconstruction operators in the square test. From left to right: only P1, only CWENO, the usual setting P1 + CWENO

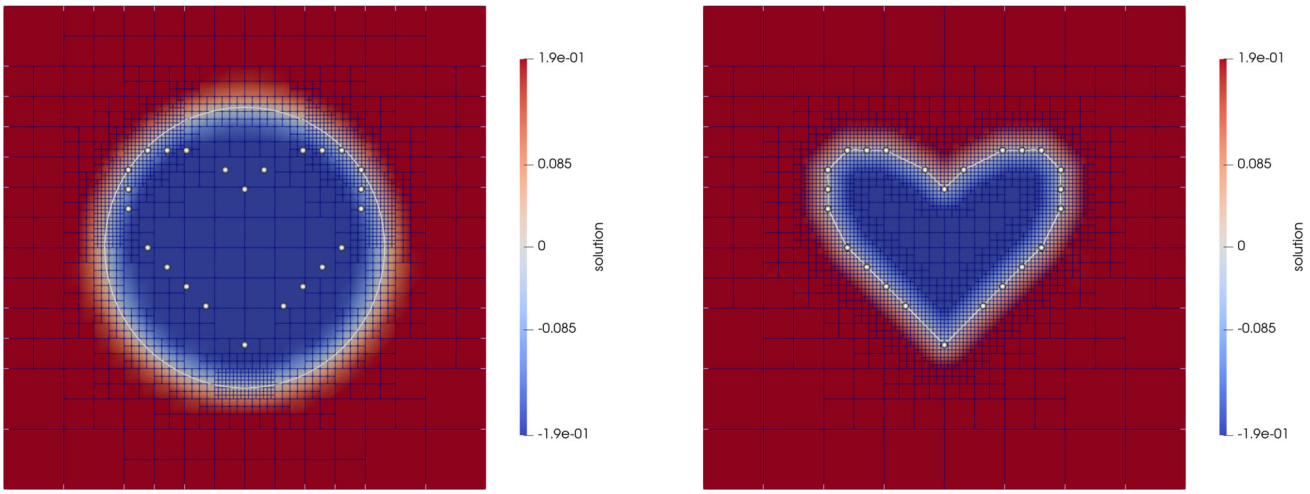

Fig. 5 Reconstruction from a heart-shaped point cloud. From left to right: the initial data and the final signed distance function. The zero isocontour of ϕ is depicted by the white line

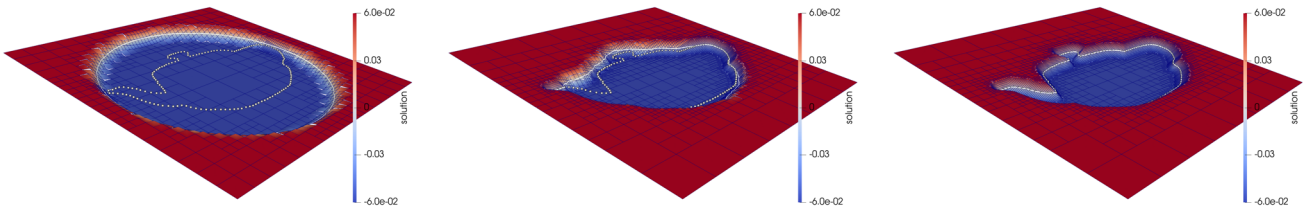

Fig. 6 Reconstruction from a teapot-shaped point cloud. From left to right: the initial data, an intermediate state and the final signed distance function. The zero isocontour of ϕ is depicted by the white line. Note that the grid around the white line is finer where the level set is closed to the point cloud

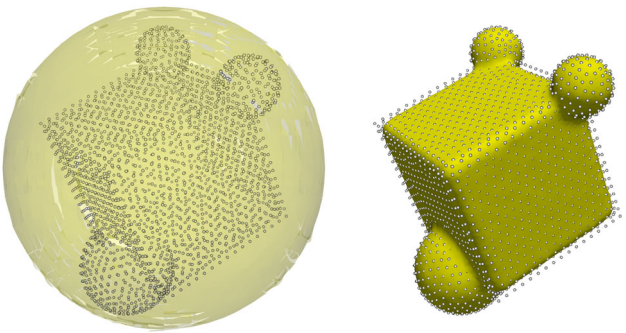

Fig. 7 The initial surface and the reconstructed one of the "Cube&Spheres"

the grid gradually increases as we get close to the data. The coarser grid allows to speed up the evolution when we are far from $\mathcal{S}$, while the maximum level of refinement is effectively employed to capture all the details.

4.4 "Cube&Spheres" Synthetic 3D Test

We start the 3D session of tests with a domain composed by a cube joined with three spheres, sampled by 2346 points, which we refer to as "Cube&Spheres". The cube edge length is 0.8, the first sphere has radius 0.25 and centre at the middle of an edge of the cube, while the other two had radius 0.15 and were centred onto the two vertices of the opposite edge of the cube. The geometrical object was rotated in such a way that no face nor edge were aligned with the background grid. We find $h_{\mathcal{S}} \approx 5.60 \times 10^{-2}$ and start the evolution with $\Delta x_{\min} = 2.38 \times 10^{-2}$, by setting $C_{\mathcal{S}} = 0.5$. After 28 iterations we get the reconstruction shown in Fig. 7 and, since the exact solution is given, we can evaluate our approximation both in terms of L^1-error and cloud fidelity equal to 4.32×10^{-3} and 4.45×10^{-3}, respectively. As we expected, higher values of μ lead to more regular solutions. This is clear from Fig. 8, where we show some isocontours of the final level set computed by setting initial values of μ equal to 0.2 and 0.05, respectively. At the expense of exact vanishing on the point cloud ($Err_{\mathcal{S}} = 3.57 \times 10^{-3}$ for $\mu = 0.05$), one can appreciate the improved regularity of the final reconstruction.

4.5 Real Data

Here we test our method using data sets coming from laser scanning of real objects, namely the "Frog" and the "Head" point clouds coming respectively from [78] and [19]. Their reconstructions are shown in Fig. 9.

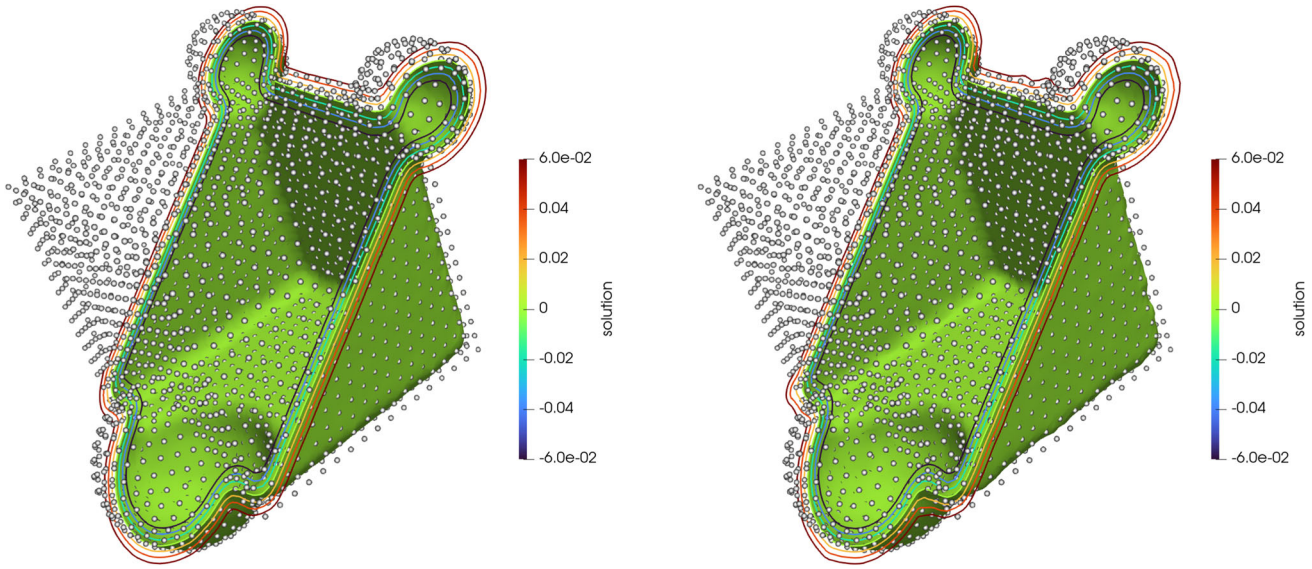

Fig. 8 Evenly spaced isocontours computed from the "Cube&Spheres" final level set for different initial values of μ. On the left, $\mu = 0.2$ gives a smoother result, whereas on the right, $\mu = 0.05$ gives a more rugged surface

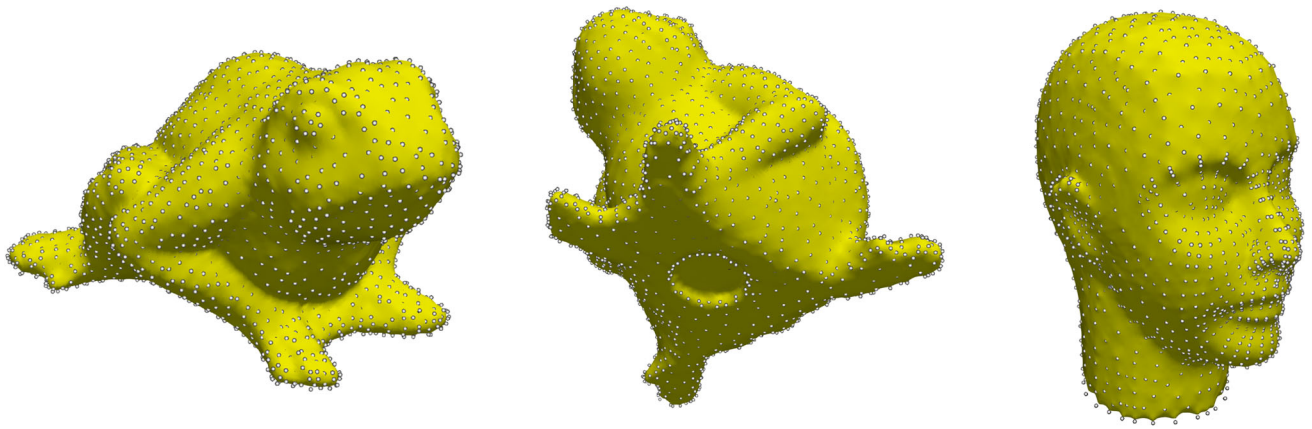

Fig. 9 Reconstructed surfaces from the "Frog" and the "Head" datasets. Note how the level set is able to handle missing data at the bottom of the "Frog" cloud

In the "Frog" test, we elaborate a cloud made up of 2512 points with $h_{\mathcal{S}} \approx 4.11 \times 10^{-2}$ and use $\Delta x_{\min} = 1.43 \times 10^{-2}$, by setting $C_{\mathcal{S}} = 0.25$. The reconstruction takes 44 iterations with a final $Err_{\mathcal{S}} = 2.37 \times 10^{-3}$. This error is mostly affected by some parts of the reconstruction where the cross section of the shape is comparable to the resolution of the cloud, namely where the curvature is high or the thickness is very thin. Among the important advantages, we point out that this type of cloud has missing data that we would like to be plugged by our level set function, as it actually does (see Fig. 9).

On the other side, the "Head" cloud consists of 1881 points having an estimated resolution $h_{\mathcal{S}} \approx 3.83 \times 10^{-2}$. Setting $C_{\mathcal{S}} = 0.25$ we start with $\Delta x_{\min} = 1.09 \times 10^{-2}$ and compute the reconstruction in 37 total iterations getting a final error on the cloud $Err_{\mathcal{S}} = 1.80 \times 10^{-3}$.

4.6 Complex Shapes

In what follows we want to test the ability of our method to handle possible changes in topology. We thus consider a complex shape representing a knot, sampled by a point cloud of 10000 points ([19]), having a resolution $h_{\mathcal{S}} \approx 1.39 \times 10^{-2}$. From Fig. 10 one can notice that, even if we start the evolution with the simplest spherical initial data, we are able to capture all the features of the shape.

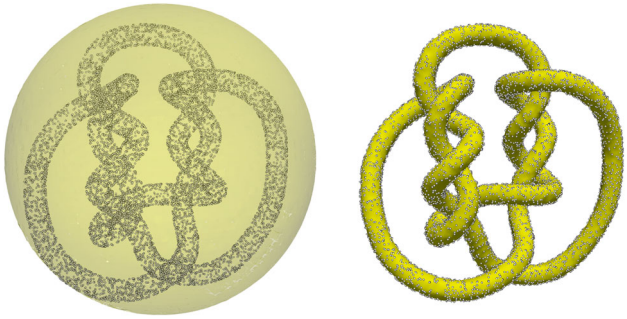

Fig. 10 From left to right: the initial data and the final reconstruction for the "Knot" dataset. Note how the evolving level set is able to handle changes not only in geometry, but especially in topology

4.7 Comparison with the Cartesian Framework

In this section we want to investigate the improvements achieved by this novel adaptive approach, compared to the previous Cartesian one, presented in [25]. The comparison is made by considering the "Frog" dataset of Sect. 4.5 and performing the simulation on a laptop equipped with an Intel Core Ultra 9 185 H CPU and 32 GB of RAM, employing 14 ranks for the computations.

The adaptive reconstruction has been obtained with $\Delta x_{\min} = 3.42 \times 10^{-2}$, with the usual setting described in this work and it is depicted in the left panel of Fig. 11. When applying the Cartesian algorithm of [25] with a number of runs $R > 1$, one is exposed to the risk of missing details in the coarse runs, causing a high error and a long time to converge. This is illustrated for $R = 2$ in the right panel of Fig. 11 and can of course be avoided by employing finer grids from the beginning. Next, we have applied the Cartesian algorithm with $R = 1$ and $\Delta x = 3.42 \times 10^{-2}$, mimicking the same changes of parameters and of reconstruction operators of the adaptive version. The central panel of Fig. 11 depicts the solution in this case and from Table 2 one can appreciate that the error is comparable with the one of the novel adaptive approach, which is slightly faster; moreover, the Cartesian setup employs ten times more degrees of freedom thus requiring a much larger memory usage.

Table 2 Comparison between the novel adaptive framework versus the Cartesian one of [25], for the "Frog" dataset. The error on the cloud, the number of iterations, the total CPU time, and the number of degrees of freedom are considered

	Cloud error	Iterations	CPU time (secs)	Degrees of freedom
Adaptive	7.65×10^{-3}	28	138	59 235
Cartesian (one grid)	6.75×10^{-3}	29	154	599 326
Cartesian (two grids)	2.16×10^{-2}	66	372	409 536

4.8 Detecting Cavities

The last part of the numerical tests is devoted to particular shapes that present tunnel-type features corresponding to real cavities, which we would like to distinguish from fake ones due to missing data. A 2D example is depicted in Fig. 12, where one can notice that if a portion of the cloud $\mathcal{S}$ is distributed along two close parallel lines, the evolution would be stopped by the vanishing of the transport term, namely by the orthogonality between $\nabla\phi$ and ∇d, even if the distance function is still large. This is topologically analogous to the case of holes in the "Knot" test, but metrically different, since by "tunnels" we mean "holes" that are sufficiently long with respect to the size of their cross section.

To address this issue we propose to modify the governing PDE (2) if the following conditions hold: on an active quadrant $j \in \tilde{\mathcal{Q}}$, if $d_j > 4h_{\mathcal{S}}$ and $\left|\nabla d_j\right| < 0.9$, we replace the velocity field ∇d with $\nabla\phi$, which is in practice a way of letting the level set ϕ proceed in the same direction followed so far. Note that a condition on the scalar product $\nabla\phi \cdot \nabla d$ might

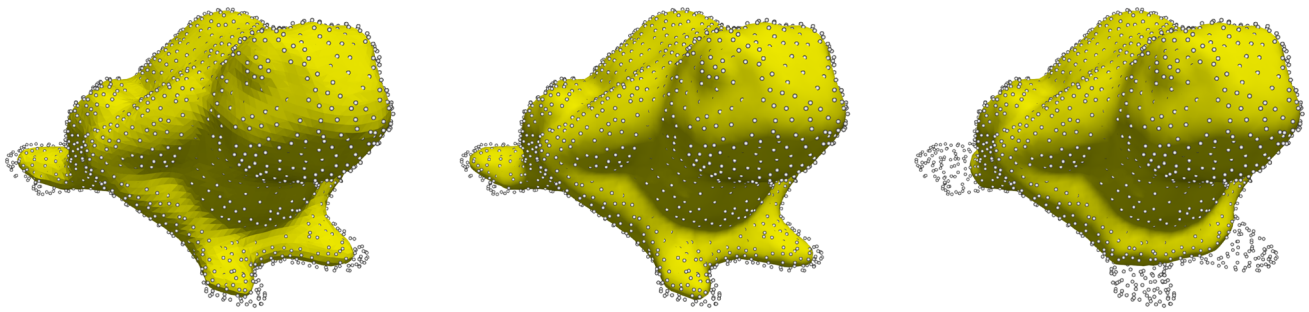

Fig. 11 Comparison between the novel adaptive framework versus the Cartesian one of [25], for the "Frog" dataset. From left to right: the final zero level set obtained with the adaptive method, the Cartesian one employing a single fine grid, and the Cartesian one employing two grids. While the first two results are comparable, in the last case, the evolution on the coarser grid is faster but completely fails to resolve the paws

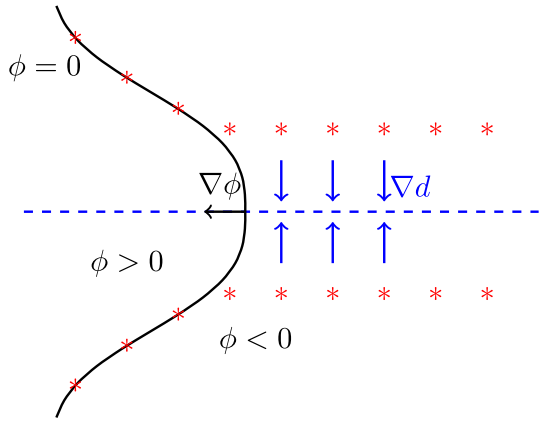

Fig. 12 Configuration of a tunnel in a point cloud $\mathcal{S}$, represented by the red star points. The evolution of the ϕ initially pushes its zero level set inside of the corridor, but it is then slowed down until it stops due to the vanishing of the transport term. In fact, once inside the tunnel, the gradient of ϕ and the velocity field ∇d become orthogonal

be too strong and might expose to the risk of perturbing the evolution even in well resolved areas.

We first consider a synthetic object constituted by a cylinder with a hole connecting the two parallel faces. The cloud has 16000 points, its resolution is $h_{\mathcal{S}} \approx 2.40 \times 10^{-2}$ and we start with $\Delta x_{\min} = 3.19 \times 10^{-2}$, by setting $C_{\mathcal{S}} = 1$. The results obtained with the original PDE (2) and with the modified one are depicted in Fig. 13, clearly showing the difference between the two algorithms in properly detecting the cavity at the centre.

Similar considerations can be done for the "Mechanical Part" test: we process a point cloud of 4102 points, with $h_{\mathcal{S}} \approx 1.84 \times 10^{-2}$, presenting various features like sharp edges and corners, rounded parts, and a cavity in the centre. We set $C_{\mathcal{S}} = 0.5$ and thus set $\Delta x_{\min} = 1.04 \times 10^{-2}$ allowing the modified velocity to come into play during the evolution. The result of the reconstruction procedure is shown in Fig. 14.

Remark We point out that the modification of the governing PDE proposed in this section can be applied for all the test cases presented in this work, even in presence of missing data, which is the case of the "Frog" of Sect. 4.5. Indeed, in such cases, the condition $|\nabla d_j| < 0.9$ would not be satisfied and the computed level set will not protrude inside the object.

5 Conclusions

We presented a fully adaptive method for the reconstruction of surfaces from a set of unorganized points, having no information about their connection nor orientation. The LSM is used to detect and evolve these surfaces in an implicit way. In particular, we aim to keep the level set function close to the signed distance function, at least in the vicinity of its zero isocontour, in order to preserve numerical accuracy and to allow end users to reliably compute surface normals and curvature from the level set itself. The numerical method is based on a

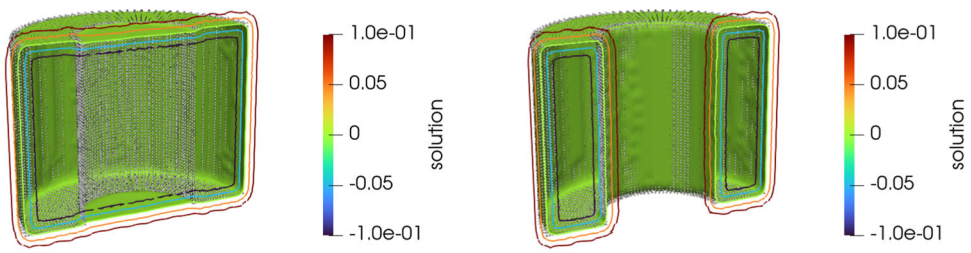

Fig. 13 From left to right: evenly spaced isocontours computed for the "Cylinder" test at the end of the evolution, using the original velocity and the modified one, respectively. Modifying the velocity field as in Sect. 4.8, the level set manages to detect the cavity

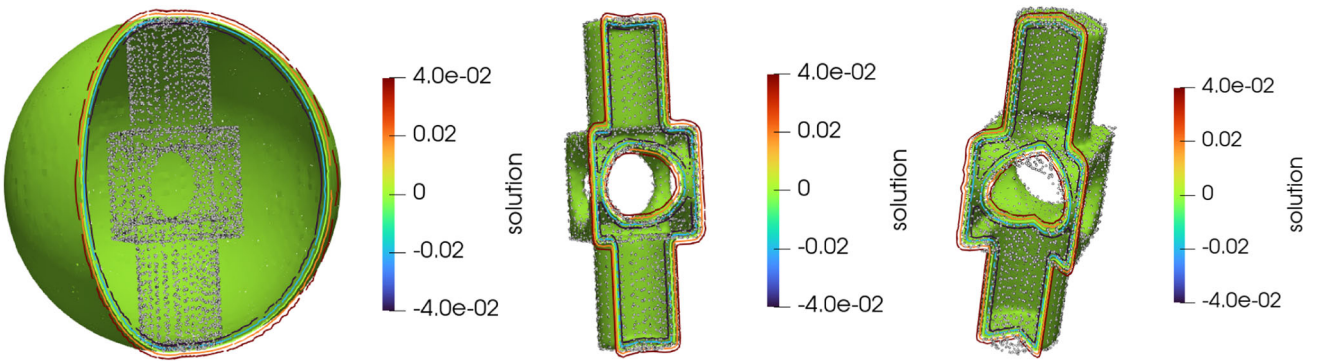

Fig. 14 From left to right: evenly spaced isocontours computed for the "Mechaninal Part" test at beginning and from two different perspectives at the end of the evolution. For this test, the modified velocity field, as described in Sect. 4.8, is considered and it allows the level set to detect the cavity in the centre

SL scheme on a quadtree (octree in 3D) grid and is coupled with a P1 or CWENO reconstruction operator based on the least-squares approach. We made use of the grid and parallel implementation of the P4EST library, and presented all the complementary procedures needed to design the algorithm.

Future research will be directed towards a finer and local estimation of the parameters employed in the method, taking into account the curvature of the level set function ϕ and a local estimation of the cloud size $h_{\mathcal{S}}$, aiming to save computational effort, without compromising the quality of the results. Also, employing a forest of trees, instead of a single octree, would be more efficient.

Acknowledgements The authors acknowledge the CINECA award under the ISCRA initiative, for the availability of high-performance computing resources and support (ISCRA Projects ID: HP10C7HWOL, HP10CO8NC7, HP10COTGT4). Both authors are members of the Gruppo Nazionale Calcolo Scientifico-Istituto Nazionale di Alta Matematica (GNCS-INdAM). This research has been partly funded by the PRIN-PNRR project "MATHematical tools for predictive maintenance and PROtection of CULTtural heritage (MATHPROCULT)" (code P20228HZWR).

Author Contributions Both authors contributed equally to the conceptualization of the method. Silvia Preda was the primary implementer of the method, and designed and performed the numerical tests.

Funding Open access funding provided by Università degli Studi dell'Insubria within the CRUI-CARE Agreement.

Data Availability No datasets were generated or analysed during the current study.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Conflict of interest The authors declare no conflict of interest.

References

1. Wang, Q., Tan, Y., Mei, Z.: Computational methods of acquisition and processing of 3D point cloud data for construction applications. Arch. Comput. Methods Eng. **27**, 479–499 (2020)
2. Remondino, F.: Heritage recording and 3D modeling with photogrammetry and 3D scanning. Remote Sens. **3**, 1104–1138 (2011)
3. Remondino, F., El-Hakim, S.: Image-based 3D modelling: a review. Photogram. Rec. **21**, 269–291 (2006)
4. Aregba Driollet, D., Diele, F., Natalini, R.A.: mathematical model for the SO_2 aggression to calcium carbonate stones: numerical approximation and asymptotic analysis. *SIAM J. Appl. Math.***64**, 1636–1667 (2004)
5. Clarelli, F., Fasano, A., Natalini, R.: Mathematics and monument conservation: free boundary models of marble sulfation. SIAM J. Appl. Math. **69**, 149–168 (2008)
6. Clarelli, F., De Filippo, B., Natalini, R.: Mathematical model of copper corrosion. Appl. Math. Model. **38**, 4804–4816 (2014)
7. Osher, S., Fedkiw, R.: Level set methods and dynamic implicit surfaces, Vol. 153 of Applied Mathematical Sciences, XIII, 273 (Springer New York, 2003)
8. Gibou, F., Fedkiw, R.P., Cheng, L.-T., Kang, M.: A second-order-accurate symmetric discretization of the poisson equation on irregular domains. J. Comput. Phys. **176**, 205–227 (2002)
9. Coco, A., Russo, G.: Second order finite-difference ghost-point multigrid methods for elliptic problems with discontinuous coefficients on an arbitrary interface. J. Comput. Phys. **361**, 299–330 (2018)
10. Coco, A., Russo, G.: High-order finite-difference ghost-point methods for elliptic problems in domains with curved boundaries. Open Math. **22**, 20240072 (2024). https://doi.org/10.1515/math-2024-0072
11. Amenta, N., Bern, M., Kamvysselis, M., Cunningham, S., Bransford, W., Cohen, M. F.: A new Voronoi-based surface reconstruction algorithm. In: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1998, 415 – 422 (1998)
12. Cazals, F., Giesen, J.: Delaunay triangulation based surface reconstruction. Springer, Berlin Heidelberg (2006)
13. Park, I. K., Yun, I. D., Lee, S. U.: Constructing NURBS surface model from scattered and unorganized range data. In: Werner, B. (ed.) Second International Conference on 3-D Digital Imaging and Modeling (Cat. No.PR00062), 312–320 (1999)
14. Gregorski, B., Hamann, B., Joy, K.: Reconstruction of B-spline surfaces from scattered data points. In: Rawlinson, A. (ed.) Proceedings Computer Graphics International 2000, 163–170 (2000)
15. Carr, J., et al.: Reconstruction and representation of 3D objects with radial basis functions. ACM SIGGRAPH (2001)
16. Carr, J., et al.: Smooth surface reconstruction from noisy range data. In: Adcock, M., Gwilt, I., Tsui, L. Y. (eds.) Proceedings of the 1st International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia, GRAPHITE '03, 119–297 (2003)
17. Zeng, Y., Zhu, Y.: Implicit surface reconstruction based on a new interpolation/approximation radial basis function. Comput. Aided Geom. Des. **92**, 102062 (2021)
18. Cheng, Z. Q., et al.: A survey of methods for moving least squares surfaces. In: Hege, H. C., Laidlaw, D., Pajarola, R., Staadt, O. (eds.) IEEE/EG Symposium on Volume and Point-Based Graphics, 9–23 (2008)
19. Hoppe, H., Derose, T., Duchamp, T., Mcdonald, J., Stuet-zle, W.: Surface reconstruction from unorganized point clouds. Comput. Graph. 71–78 (1992)
20. Osher, S., Sethian, J.A.: Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations. J. Comput. Phys. **79**, 12–49 (1988)
21. Sussman, M., Smereka, P., Osher, S.: A level set approach for computing solutions to incompressible two-phase flow. J. Comput. Phys. **114**, 146–159 (1994)

22. Sethian, J.: Level set methods and fast marching methods. Evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science (Cambridge University Press, 1999)
23. Zhao, H.K., Osher, S., Merriman, B., Kang, M.: Implicit and non-parametric shape reconstruction from unorganized data using a variational level set method. Comput. Vis. Image Underst. **80**, 295–314 (2000)
24. Marcon, M., Piccarreta, L., Sarti, A., Tubaro, S.: Fast point-cloud wrapping through level-set evolution. In: Grau, O. (ed.) 1st European Conference on Visual Media Production (CVMP) 2004, 119–125 (2004)
25. Preda, S., Semplice, M.: Surface reconstruction from point cloud using a semi-Lagrangian scheme with local interpolator. SIAM J. Sci. Comput. 103 (2025)
26. Coco, A., Preda, S., Semplice, M.: From point clouds to 3D simulations of marble sulfation. In: Bretti, G., Cavaterra, C., Solci, M., Spagnuolo, M. (eds.) Mathematical Modeling in Cultural Heritage, 153–174 (Springer Nature Singapore, 2023)
27. Coco, A., Semplice, M., Serra Capizzano, S.: A level-set multigrid technique for nonlinear diffusion in the numerical simulation of marble degradation under chemical pollutants. Appl. Math. Comput. **386**, 125503 (2020)
28. Kósa, B., Haličková-Brehovská, J.: New efficient numerical method for 3D point cloud surface reconstruction by using level set methods. In: Mikula, K., Ševčovič, D., Urbán, J. (eds.) Proceedings of Equadiff 2017 Conference, 387–396 (2017)
29. Zhao, H. K., Osher, S., Fedkiw, R.: Fast surface reconstruction using the level set method. In: Williams, A. D. (ed.) Proceedings of IEEE Workshop on Variational and Level Set Methods in Computer Vision, 194–201 (2001)
30. Carlini, E., Ferretti, R.: A Semi-Lagrangian scheme with radial basis approximation for surface reconstruction. Comput. Vis. Sci. **18**, 103–112 (2017)
31. He, Y., Huska, M., Kang, S. H., Liu, H.: Fast algorithms for surface reconstruction from point cloud. In: Li, M., Han, X. Cheng, J. (eds.) Springer Proceedings in Mathematics and Statistics, Vol. 360, 61–80 (2021)
32. Courant, R., Isaacson, E., Rees, M.: On the solution of nonlinear hyperbolic differential equations by finite differences. Comm. Pure Appl. Math. **5**, 243–255 (1952)
33. Falcone., M., Ferretti, R.: Semi-Lagrangian approximation schemes for linear and Hamilton-Jacobi equations. In: Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, (2014)
34. Bonaventura, L., Ferretti, R.: Semi-Lagrangian methods for parabolic problems in divergence form. SIAM J. Sci. Comput. **36**, A2458–A2477 (2014)
35. Bonaventura, L., Calzola, E., Carlini, E., Ferretti, R.: Second order Fully Semi-Lagrangian discretizations of advection-diffusion-reaction systems. J. Sci. Comput. **88** (2021)
36. Strain, J.: Semi-Lagrangian methods for level set equations. J. Comput. Phys. **151**, 498–533 (1999)
37. Levy, D., Puppo, G., Russo, G.: Compact central WENO schemes for multidimensional conservation laws. SIAM J. Sci. Comput. **22**, 656–672 (2000)
38. Balsara, D.S., Garain, S., Florinski, V., Boscheri, W.: An efficient class of WENO schemes with adaptive order for unstructured meshes. J. Comput. Phys. **404**, 109062 (2020)
39. Zhu, J., Qiu, J.: A new fifth order finite difference WENO scheme for solving hyperbolic conservation laws. J. Comput. Phys. **318**, 110–121 (2016)
40. Baeza, A., Bürger, R., Mulet, P., Zorío, D.: Central WENO schemes through a global average weight. J. Sci. Comput. **78**, 499–530 (2019)
41. Zhou, J., Cai, L., Zhou, F.-Q.: New high-resolution scheme for three-dimensional nonlinear hyperbolic conservation laws. Appl. Math. Comp. **198**, 770–786 (2008)
42. Dumbser, M., Boscheri, W., Semplice, M., Russo, G.: Central weighted ENO schemes for hyperbolic conservation laws on fixed and moving unstructured meshes. SIAM J. Sci. Comput. **39**, A2564–A2591 (2017)
43. Zhu, J., Qiu, J.: A new fifth order finite difference WENO scheme for Hamilton-Jacobi equations. Numer. Meth. for PDEs **33**, 1095–1113 (2017)
44. Zheng, F., Shu, C.W., Qiu, J.: High order finite difference hermite WENO schemes for the Hamilton-Jacobi equations on unstructured meshes. Comp. Fluids **183**, 53–65 (2019)
45. Zhu, J., Qiu, J.: A New Type of High-Order WENO Schemes for Hamilton-Jacobi Equations on Triangular Meshes. Comm. Comput. Phys. **27**, 897–920 (2020)
46. Cravero, I., Puppo, G., Semplice, M., Visconti, G.: CWENO: uniformly accurate reconstructions for balance laws. Math. Comp. **87**, 1689–1719 (2018)
47. Cravero, I., Semplice, M., Visconti, G.: Optimal definition of the nonlinear weights in multidimensional Central WENOZ reconstructions. SIAM J. Numer. Anal. **57**, 2328–2358 (2019)
48. Semplice, M., Visconti, G.: Efficient implementation of adaptive order reconstructions. J. Sci. Comput. **83** (2020)
49. Peng, D., Merriman, B., Osher, S., Zhao, H.K., Kang, M.: A PDE-based fast local level set method. J. Comput. Phys. **155**, 410–438 (1999)
50. Saye, R.: High-order methods for computing distances to implicitly defined surfaces. Commun. Appl. Math. Comput. Sci. **9**, 107–141 (2014)
51. Zhao, H.K.: A fast sweeping method for Eikonal equations. Math. Comput. **74**, 603–627 (2005)
52. Falcone, M., Ferretti, R.: Consistency of a large time-step scheme for mean curvature motion. In: Brezzi, F., Buffa, A., Corsaro, S., Murli, A. (eds.) Numerical Mathematics and Advanced Applications, 495–502 (Springer Milan, 2003)
53. Carlini, E., Falcone, M., Ferretti, R.: Convergence of a large time-step scheme for mean curvature motion. Interfaces Free Bound. **12**, 409–411 (2010)
54. Cravero, I., Semplice, M., Visconti, G.: Optimal definition of the nonlinear weights in multidimensional Central WENOZ reconstructions. SIAM J. Numer. Anal. **57**, 2328–2358 (2019)
55. Carlini, E., Ferretti, R., Preda, S., Semplice, M.: A CWENO Large Time-Step Scheme for Hamilton-Jacobi Equations. Commun. Appl. Math, Comput (2025)
56. Jiang, G.S., Shu, C.W.: Efficient implementation of weighted ENO schemes. J. Comput. Phys. **126**, 202–228 (1996)
57. Hu, C., Shu, C.W.: Weighted essentially non-oscillatory schemes on triangular meshes. J. Comput. Phys. **150**, 97–127 (1999)
58. Jiang, G.-S., Peng, D.: Weighted ENO schemes for Hamilton-Jacobi equations. SIAM J. Sci. Comput. **21**, 2126–2143 (2000)
59. Falcone, M., Paolucci, G., Tozza, S.: Multidimensional smoothness indicators for first-order Hamilton-Jacobi equations. J. Comput. Phys. **409** (2020)
60. Tsitsiklis, J.N.: Efficient algorithms for globally optimal trajectories. IEEE Trans. Autom. Control **40**, 1528–1538 (1995)
61. Sethian, J.A.: A fast marching level set method for monotonically advancing fronts. Proc. Natl. Acad. Sci. **93**, 1591–1595 (1996)
62. Chopp, D.L.: Some Improvements of the Fast Marching Method. SIAM J. Sci. Comput. **23**, 230–244 (2001)
63. Tsai, Y.-H.R., Cheng, L.-T., Osher, S., Zhao, H.-K.: Fast Sweeping Algorithms for a Class of Hamilton-Jacobi Equations. SIAM J. Numer. Anal. **41**, 673–694 (2003)
64. Herrmann, M.: A Domain Decomposition Parallelization of the Fast Marching Method. Tech. Rep, DTIC Document (2003)

65. Tugurlan, M. C.: Fast marching methods—parallel implementation and analysis. Ph.D. thesis (2008)
66. Zhao, H.: Parallel implementations of the fast sweeping method. J. Comput. Math. **25**, 421–429 (2007)
67. Chacon, A., Vladimirsky, A.: A parallel two-scale method for Eikonal equations. SIAM J. Sci. Comput. **37**, A156–A180 (2015)
68. Jeong, W.K., Whitaker, R.T.: A fast iterative method for Eikonal equations. SIAM J. Sci. Comput. **30**, 2512–2534 (2008)
69. Qian, J., Zhang, Y., Zhao, H.: Fast sweeping methods for Eikonal equations on triangular meshes. SIAM J. Numer. Anal. **45**, 83–107 (2007)
70. Chen, X., Cao, D., Fu, X.: A fast sweeping method based on coordinate transformation for eikonal equation on unstructured triangular meshes. In: Fourth International Meeting for Applied Geoscience & Energy of SEG Technical Program Expanded Abstracts (2024)
71. Zhao, H.-K., Chan, T., Merriman, B., Osher, S.: A variational level set approach to multiphase motion. J. Comput. Phys. **127**, 179–195 (1996)
72. Li, C., Xu, C., Gui, C., Fox, M. D.: Level set evolution without re-initialization: a new variational formulation. In: Schmid, C., Soatto, S., Tomasi, C. (eds.) 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), Vol. 1, 430–436 (2005)
73. Wu, W., Wu, Y., Huang, Q.: An improved distance regularized level set evolution without re-initialization. In: IEEE 2012 IEEE Fifth International Conference on Advanced Computational Intelligence (ICACI), 631–636 (2012)
74. Falcone, M., Paolucci, G., Tozza, S.: A high-order scheme for image segmentation via a modified level-set method. SIAM J. Imaging Sci. **13**, 497–534 (2020)
75. Sussman, M., Smereka, P., Osher, S.: A level set approach for computing solutions to incompressible two-phase flow. J. Comput. Phys. **114**, 146–159 (1994)
76. Mirzadeh, M., Guittet, A., Burstedde, C., Gibou, F.: Parallel level-set methods on adaptive tree-based grids. J. Comput. Phys. **322**, 345–364 (2016)
77. Min, C., Gibou, F.: A second order accurate level set method on non-graded adaptive cartesian grids. J. Comput. Phys. **225**, 300–321 (2007)
78. AIM@SHAPE & VISIONAIR. VISIONAIR shape repository. http://visionair.ge.imati.cnr.it/ontologies/shapes/ (2022). http://visionair.ge.imati.cnr.it/ontologies/shapes/view.jsp?id=268-frog_-_merged. Accessed March 2022
79. Laboratory, S. U. C. G.: The Stanford 3D Scanning Repository. http://graphics.stanford.edu/data/3Dscanrep/. Accessed March 2022
80. Burstedde, C., Wilcox, L.C., Ghattas, O.: `p4est`: Scalable Algorithms for Parallel Adaptive Mesh Refinement on Forests of Octrees. SIAM J. Sci. Comput. **33**, 1103–1133 (2011)
81. Haličková, J., Mikula, K.: Level set method for surface reconstruction and its application in surveying. J. Surveying Engrg. **142**, 04016007 (2016)

Silvia Preda earned a M.Sc. in Mathematics from Università degli Studi di Pavia, in 2021, and a PhD in Computer Science and Mathematics of Computation from Università degli Studi dell'Insubria (Como, Italy), in 2025. She is currently a Postdoctoral Researcher at the Università degli Studi dell'Insubria, at the Department of Science and High Technology. Her research interests include numerical methods for Partial Differential Equations, with a focus on semi-Lagrangian and finite volume schemes, as well as level set methods for surface reconstruction and manipulation. She also works on adaptive and unstructured grids and parallel computing.

Matteo Semplice graduated in Mathematics at Università Statale di Milano and obtained a PhD at Oxford University in 2002. He is currently Associate Professor of Numerical Analysis at Università degli Studi dell'Insubruia (Como, Italy). His research interests revolve around high order numerical methods for hyperbolic conservation laws, and more broadly on the numerics for Partial Differential Equations in scientific computing applications also beyond computational fluid dynamics.