Journal of Scientific Computing (2025) 103:38
https://doi.org/10.1007/s10915-025-02843-y

Surface Reconstruction From Point Cloud using a Semi-Lagrangian Scheme with Local Interpolator

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

Received: 25 March 2024 / Revised: 17 December 2024 / Accepted: 6 February 2025 /
Published online: 18 March 2025

Abstract
We propose a level set method to reconstruct unknown surfaces from point clouds, without assuming that the connections between points are known. We consider a variational formulation with a curvature constraint that minimizes the surface area weighted by the distance of the surface from the point cloud. More precisely we solve an equivalent advection–diffusion equation that governs the evolution of an initial surface described implicitly by a level set function. Among all the possible representations, we aim to compute the signed distance function at least in the vicinity of the reconstructed surface. The numerical method for the approximation of the solution is based on a semi-Lagrangian scheme coupled with a local interpolator. In particular, we resort to a multi-linear interpolator and to a Weighted Essentially Non-oscillatory one, to improve the accuracy of the reconstruction. An analysis of the parameters employed in the model is given, focusing in particular on the effect of the curvature regularization, and on the presence of noisy data. Special attention has been paid to the localization of the method and to the development of fast algorithms that run in parallel, resulting in faster reconstruction and thus the opportunity to easily improve the resolution. Numerical tests in two and three dimensions are presented to evaluate the quality of the reconstruction and the efficiency of the algorithm in terms of computational time.

Keywords Curvature regularized surface reconstruction · Point cloud · Level set methods · Semi-Lagrangian schemes · WENO interpolation

1 Introduction

The problem of acquiring, creating and processing 3d digital models of real objects has gained importance in numerous applications across diverse industries and areas of research including physics, computer graphics, medical imaging, urban planning and cultural heritage. Especially in the latter, real artistic manufacts usually present complicated geometries and topologies and they are not built from precise (e.g. CAD) drawings. Therefore, their shape

✉ Silvia Preda
silvia.preda@uninsubria.it

Matteo Semplice
matteo.semplice@uninsubria.it

[1] Dipartimento di Scienza e Alta Tecnologia, Università dell'Insubria, Via Valleggio 11, 22100 Como, Italy

must be acquired from the real object itself, e.g. via 3d laser scanning or photogrammetry [43, 44, 50].

In the field of cultural heritage, predictive mathematical models for the deterioration like [3, 18, 19] require the solution of reaction-diffusion Partial Differential Equations (PDEs) on a computational domain having the exact shape of a work of art. Applications in this sense like the workflow from point cloud to PDE solution presented in [20, 22] are the main goal of the present work. In particular, our main interest is to produce high fidelity level set descriptions of complex objects, which can be later used as domain definition in PDE computations via ghost-cell methods [21, 28, 39]. To this end we will employ 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 many cases, the only information available for the reconstruction of the original shape is a set of unorganized and non-uniformly spaced points that does not contain any information about the ordering or the connection between them. This makes surface reconstruction from point clouds complex and time-consuming. Additionally, as an infinite number of surfaces may pass through or near the data points, this problem turns out to be ill-posed, with no unique solution. The challenge is thus to get a good approximation of the data set that should have some smoothness properties while being able to retain as many details and features of the real object as possible. Moreover, the reconstruction should be also useful for dynamic operations and not only for static representations.

In general, there are two approaches for representing a surface: the explicit one and the implicit one. The former includes mesh-based techniques, such as Delaunay triagulations and Voronoi diagrams [2, 16, 25] and parametric techniques, e.g. NURBS [29, 41]. Although they represent some of the most explored techniques in this topic, the main problem with this approach is to ensure a watertight reconstruction in cases when an evolution of the surface occurs. Recent works, for example [24], propose to overcome this difficulty by considering an additional mesh adjustment in order to control the quality of the mesh during the evolution. However, the state of the art agrees that explicit reconstructions are difficult to handle when dealing with moving boundaries or complex and changing topologies.

On the other hand, using an implicit approach, one has the advantage of better handling topological flexibility, while having a very simple data structure that also allows very simple Boolean operations on the detected surface. Among implicit methods, traditional ones enclose interpolation techniques to approximate the desired implicit function as a combination of some smooth basis functions. The problem here is obviously related to the large linear systems that need to be solved, resulting in high computational costs and, even worse, in ill-conditioned matrices when the number of interpolated points is very large. Radial Basis Functions (RBF) with global and local support have found many applications along this line [14, 15, 51] and some least-square-based methods [17] belong to this family too. Other methods use local estimators to associate an oriented plane to each point in the cloud and thus approximate the signed distance function representing the surface [33]. A different and widespread approach for the processing of point cloud data is constituted by the application of the Level Set Method (LSM) [40]. This last line of research constitutes the focus of this work.

For a broader overview about the reconstruction of real objects starting from point clouds, the reader can refer to [6, 7]. In these surveys various algorithms have been compared and classified according to their ability to handle imperfections in the point clouds, their input requirements, the class of shape that they are able to recover and the type and the quality of the final reconstruction. Significant advancements in surface reconstruction are also being driven by the integration of machine learning techniques, especially deep learning models,

which have shown remarkable results in learning implicit surface representations directly from unorganized point clouds [37, 46, 48].

Since its introduction in [40], the LSM has emerged as a powerful and versatile tool in a wide range of applications [39, 45, 49] including image processing and surface reconstruction [38, 54]. The central idea is to represent both an n-dimensional object Ξ and its boundary by a so called level set function $\phi : \mathbb{R}^n \to \mathbb{R}$ such that $\Xi = \{\vec{x} \in \mathbb{R}^n : \phi(\vec{x}) < 0\}$ and its boundary Γ is the zero isocontour of ϕ. Ξ and Γ can be then evolved in time using PDEs for ϕ, driving the deformation of an initial surface. Following this approach, one can formulate the reconstruction problem as an evolutive problem and easily handle noisy and incomplete point clouds, since the level set function evolves smoothly and naturally adapts to the input data, enjoying all the geometrical and topological advantages of working with implicit representations.

Our work thus delves into the approximation of the solution of the level set evolution PDE presented in [54] that leads the evolution of an initial guess to get the final, steady-state shape. The model is based on the minimization of the energy functional

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

where $d(\vec{x})$ is the distance of $\vec{x}$ from the point cloud. The curvature term present in the corresponding evolutive PDE for the level set function allows us to trade off the exact vanishing of the level set function on the points of the cloud for a control on the maximum curvature of the resulting zero level set surface. This curvature term is also expected to be especially useful in counteracting noise in the data.

The model introduced in [54] has been first improved in [53] and more recently adopted in [12, 30, 32, 35]. In particular, in [30, 35] the authors present a second order explicit finite difference scheme and change the original $d(\vec{x})$ factor in front of the curvature regularization into a constant tunable parameter. In [32] two approaches based respectively on a Semi-Implicit Method and an Augmented Lagrangian Method, discretized with spectral methods but adding stabilizing diffusion terms, are explored. Finally, in [12] a semi-Lagrangian (SL) scheme is adopted with the aim of overcoming the prohibitive time-step constraint imposed by the diffusive term.

In fact, first introduced in [23] for first-order systems of linear equations, SL schemes have been then extended to Hamilton-Jacobi (HJ) equations and to the treatment of parabolic terms with the main purpose of obtaining methods which are unconditionally stable with respect to the choice of the time step (see [27] for a comprehensive explanation). The possibility to overcome the parabolic-type CFL restriction is useful in the LSM context since this method usually focuses just on what is happening in the vicinity of the zero level set of ϕ, and one could thus employ very fine grids there, including Adaptive Mesh Refinement (AMR) techniques. While the CFL condition will frustrate the adaptive approach when a general explicit scheme is employed, this will not be such an issue in the SL framework where one is allowed to work at large Courant numbers with a hyperbolic CFL restriction of type $\Delta t = \mathcal{O}(\Delta x)$. In the context of LSM, a first application of SL schemes can be found in [47].

Motivated by the reasons above, here we apply the SL scheme of [12], first presented in [26] for curvature-related equations, but we use a local interpolator for space reconstructions. We employ both a multilinear interpolation and a third order accurate Weighted Essentially Non-oscillatory (WENO) reconstruction [13]. We find that both are faster than the RBF approach of [12], with the second one yielding more satisfactory results. Of course, we employ a constrained reinitialization scheme [31] to compute a signed distance function and we employ

distributed memory parallelization and localization strategies to reduce the computational cost.

With respect to previous works based on the aforementioned energy functional, here we adopt the recommended value of $p = 2$ [54], we study how to handle noise in the data and we perform a thorough study of the errors committed by the methods, measuring them in various metrics.

The paper is organized as follows. In Sect. 2 we briefly recall the mathematical model [54] and the level set formulation [40] that lays behind our governing PDE. In Sect. 3 all the numerical schemes used in the algorithm are detailed, together with our strategies to save computational costs. In Sect. 4 two- and three-dimensional tests of increasing complexity are presented. Finally Sect. 5 draws some conclusions and highlights future perspectives.

2 Mathematical Model

Let us assume that we are given a set of points $\mathcal{S} = \{Q_1, \ldots, Q_M\}$ in a bounded region of $\mathbb{R}^n$ and let us define $d(\vec{x}) = \min_{Q \in \mathcal{S}} |\vec{x} - Q|$ to be the distance function to $\mathcal{S}$. Our model for surface reconstruction from point clouds is based on the minimization of the surface energy

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

where Γ is a closed surface of co-dimension one in $\mathbb{R}^n$.

To achieve the minimum and get the final shape, we continuously deform an initial surface Γ^0 following the gradient descent of the energy functional(1). In this process, the evolving surface $\Gamma(t)$ is represented implicitly using a level set function to capture the moving interface (see [54] for full details).

Let $\Xi(t)$ be the region enclosed by $\Gamma(t)$. The LSM, first introduced by Osher and Sethian [40], considers a level set function $\phi(\vec{x}, t)$ associated with $\Xi(t)$ such that

$$\begin{aligned} &\phi(\vec{x}, t) < 0 \quad \text{inside} \quad \Xi(t), \\ &\phi(\vec{x}, t) = 0 \quad \text{on} \quad \Gamma(t), \\ &\phi(\vec{x}, t) > 0 \quad \text{outside} \quad \Xi(t). \end{aligned} \tag{2}$$

In terms of the level set function, the energy functional (1) can be rewritten as

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

where δ denotes the Dirac-delta function. Following [54], we minimize the energy (3) by computing the solution of the following evolutionary problem

$$\begin{aligned} &\phi_t(\vec{x}, t) = \left[\frac{d(\vec{x})}{E_p(\phi)} \right]^{p-1} \left(\nabla d(\vec{x}) \cdot \nabla \phi(\vec{x}, t) + \frac{1}{p} d(\vec{x}) \nabla \cdot \left(\frac{\nabla \phi(\vec{x}, t)}{|\nabla \phi(\vec{x}, t)|} \right) |\nabla \phi(\vec{x}, t)| \right), \\ &\phi(\vec{x}, 0) = \phi^0(\vec{x}), \end{aligned} \tag{4}$$

where $\phi^0(\vec{x})$ is a suitable initial data such that $\Gamma^0 = \{\vec{x} \in \mathbb{R}^n : \phi^0(\vec{x}) = 0\}$.

In the above equation, the term $\nabla d(\vec{x}) \cdot \nabla \phi(\vec{x}, t)$ drives the surface towards the dataset $\mathcal{S}$, while the second term tempers the maximal curvature of Γ. Changing the balance between

them can lead to surfaces closer to the point cloud but with sharper edges or to more rounded surfaces that are a little further away from $\mathcal{S}$.

The parameter p controls the relative influence of the curvature term and of the global scaling factor $\left[\frac{d(\vec{x})}{E_p(\phi)}\right]^{p-1}$ which makes the most remote points move more quickly than the ones closer to $\mathcal{S}$, especially when the zero level set ϕ is close to the data set. We point out that, except for the degenerate case, $E_p(\phi)$ evolves towards a non-zero local minimum. On the other hand, for $p > 1$, this term slows down the entire evolution when the functional is large; this can be an issue when the initial surface is very far from the cloud.

In [35], a model similar to (4) is employed, wherein the factor $d(\vec{x})$ in front of the curvature term is replaced by a constant $\mu \in [0, 1]$ to control the balance between the two terms. While we prefer to keep the $d(\vec{x})$ factor in the diffusion term which will slow down the evolution in the vicinity of the point cloud, we also introduce an additional parameter $\mu \geq 0$ multiplying $d(\vec{x})$ in the curvature term. Our final formulation then reads

$$\begin{aligned} \phi_t(\vec{x}, t) &= \left[\frac{d(\vec{x})}{E_p(\phi)}\right]^{p-1} \left(\nabla d(\vec{x}) \cdot \nabla \phi(\vec{x}, t) + \frac{\mu}{p} d(\vec{x}) \nabla \cdot \left(\frac{\nabla \phi(\vec{x}, t)}{|\nabla \phi(\vec{x}, t)|}\right) |\nabla \phi(\vec{x}, t)|\right), \\ \phi(\vec{x}, 0) &= \phi^0(\vec{x}). \end{aligned} \tag{5}$$

In this way our numerical algorithm can be used in both regimes. If we set $\mu = 0$, the model becomes purely convective and leads quickly to a reconstructed surface close to a piecewise linear approximation. In contrast, the case with $\mu = 1$ corresponds to the original formulation which contains a weighted curvature regularization effect, is more computationally expensive, but leads to a smoother reconstructed surface. Higher values of μ are also useful in the case of noisy data.

Finally, we observe that for visualization purposes it is often sufficient to compute a function ϕ such that its zero level set is close to the data set $\mathcal{S}$. Equation (5) is apt for this purpose, but its transport term tends to produce high gradients in the computed function ϕ. However, for computing normals of the reconstructed surface or to use the level set function as a mathematical description of a domain into which a PDE solver has to be applied, it is more robust to compute a level set function with moderate gradients. In particular, a level set function with $|\nabla \phi| = 1$ is called a signed-distance function and our algorithm will be designed to ensure that the computed solution is a signed distance at least in the vicinity of the point cloud.

3 Numerical Scheme

The numerical evolution of (5) is computed following the semi-Lagrangian approach presented in [12]. As such, the scheme is explicit, but not constrained by a parabolic-type CFL condition. However, the interpolation operator proposed in [12] involves the solution of a large linear system at each timestep. For this reason, we will propose the use of a local interpolant, which is more efficient, especially in view of a parallel implementation.

The scheme considers a background Cartesian mesh $\mathcal{G}$ with uniform mesh width Δx on a bounded region of $\mathbb{R}^2$ or $\mathbb{R}^3$, containing the point cloud $\mathcal{S}$. Let $\vec{x}_j \in \mathcal{G}$ be a point in the Cartesian mesh and let us denote by ϕ_j^n the approximate value of ϕ at $\vec{x}_j$ and time t^n.

The semi-Lagrangian scheme will be recalled in Sect. 3.1 and the interpolation operator will be described in Sect. 3.2. The later subsections will describe the other algorithms used in our method: the computation of the distance function $d(\vec{x})$ from the point cloud, the evaluation

Algorithm 1 Computing the levelset ϕ from a point cloud $\mathcal{S}$

1. Create a Cartesian grid $\mathcal{G}$ encompassing all of $\mathcal{S}$
2. Compute the distance function $d(\vec{x})$ at all grid points, see §3.4
3. Set initial data $\{\phi_j^0\}_{\vec{x}_j \in \mathcal{G}}$ as in §3.6
4. Loop:
   (a) Choose computational subgrid $\widetilde{\mathcal{G}} \subset \mathcal{G}$ as in §3.8
   (b) Compute the energy functional (3) with one of the methods of §3.5
   (c) Compute $\{\phi_j^{n+1}\}_{\vec{x}_j \in \widetilde{\mathcal{G}}}$ using (6) or (8) from §3.1 and the interpolator (11) or (16) from §3.2
   (d) Reinitialize $\{\phi_j^{n+1}\}_{j \in \overline{\mathcal{G}}}$ as in §3.7 and cut with (29)

of the energy functional $E_p(\phi)$, the choice of the initial data ϕ^0, the reinitialization procedure and the use of cut-off functions to reduce the computational effort. The complete algorithm is summarized in Algorithm 1. Unless otherwise specified, all numerical gradients needed in the algorithms are computed by centered finite differences.

3.1 Semi-Lagrangian Scheme

The scheme follows a SL approach both for the advection and for the diffusion term. In particular, the diffusion term is discretized by averaging the data in a region of size $\sqrt{\Delta t}$ around the foot of the characteristic of the advection term, as first proposed in [8, 9].

2d Case

We compute the update of ϕ_j^n as

$$\begin{aligned} \phi_j^{n+1} &= \frac{1}{2}\sum_{i=1}^{2} I[\Phi^n]\left(\vec{x}_{j,i}^*\right), \\ \vec{x}_{j,i}^* &= \vec{x}_j + C_j^n \Delta t \nabla d(\vec{x}_j) + \sqrt{\frac{2C_j^n \mu\, d(\vec{x}_j)\Delta t}{p}}\, \vec{\sigma}_j^n \lambda_i, \end{aligned} \tag{6}$$

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 $I[\Phi^n](\vec{x})$ denotes an interpolation at point $\vec{x}$ of the data $\Phi^n = \{\phi_j^n : \vec{x}_j \in \mathcal{G}\}$. This operator will be specified later. $\vec{\sigma}_j^n$ denotes the unit vector tangent to the level sets of ϕ: it is thus orthogonal to the gradient of the level set function and is given by

$$\vec{\sigma}_j^n = \frac{1}{|\nabla\phi|}\begin{pmatrix} \partial_2\phi \\ -\partial_1\phi \end{pmatrix}. \tag{7}$$

In equation (6), the interpolation points are obtained by adding two terms: the first is the foot of the characteristic pertaining to the advection term in (5), while the second is a further displacement that generates a diffusion along the tangent space of the level sets, thus discretizing the curvature term in (5). This discretization of the second order operator has been described in [10, 26].

3d Case

Similarly to the previous case, the update of ϕ_j^n is computed as

$$\begin{aligned} \phi_j^{n+1} &= \frac{1}{4}\sum_{i=1}^{4} I[\Phi^n]\left(\vec{x}_{j,i}^{\,*}\right), \\ \vec{x}_{j,i}^{\,*} &= \vec{x}_j + C_j^n \Delta t \nabla d(\vec{x}_j) + \sqrt{\frac{2C_j^n \mu\, d(\vec{x}_j)\Delta t}{p}}\, \Sigma_j^n \vec{\lambda}_i, \end{aligned} \tag{8}$$

where again we treat the scale factor C_j^n as a constant. In the above equation, $\Sigma_j^n = \left[\nu_1(\nabla\phi_j^n), \nu_2(\nabla\phi_j^n)\right]$ is a 3×2 matrix whose columns span the $2d$ space orthogonal to $\nabla\phi_j^n$ and the column vector $\vec{\lambda}_i$ ranges over $\{(\pm 1, \pm 1)^T\}$, so that $\Sigma_j^n \vec{\lambda}_i$ represents four points in the local tangent plane.

In order to avoid numerical singularities, one regularizes the two orthonormal eigenvectors of the projection onto the plane tangent to the level sets of ϕ as

$$\nu_1 = \begin{cases} \tilde{\nu}_1 & \text{if } \sqrt{(\partial_1\phi)^2 + (\partial_3\phi)^2} \neq 0 \\ (1,0,0)^T & \text{otherwise} \end{cases}$$

$$\nu_2 = \begin{cases} \tilde{\nu}_2 & \text{if } \sqrt{(\partial_1\phi)^2 + (\partial_3\phi)^2} \neq 0 \\ (0,0,1)^T & \text{otherwise} \end{cases}$$

with $\tilde{\nu}_k$ denoting the exact eigenvectors, which are given by

$$\tilde{\nu}_1 = \begin{pmatrix} \frac{-\partial_3\phi}{\sqrt{(\partial_1\phi)^2+(\partial_3\phi)^2}} \\ 0 \\ \frac{\partial_1\phi}{\sqrt{(\partial_1\phi)^2+(\partial_3\phi)^2}} \end{pmatrix}, \quad \tilde{\nu}_2 = \frac{1}{|\nabla\phi|} \begin{pmatrix} \frac{-\partial_1\phi\,\partial_2\phi}{\sqrt{(\partial_1\phi)^2+(\partial_3\phi)^2}} \\ \sqrt{(\partial_1\phi)^2 + (\partial_3\phi)^2} \\ \frac{-\partial_2\phi\,\partial_3\phi}{\sqrt{(\partial_1\phi)^2+(\partial_3\phi)^2}} \end{pmatrix}. \tag{9}$$

Because of the factor in front of $\vec{\sigma}_j^n$ in (7), respectively in front of $\tilde{\nu}_2$ in (9), the numerical schemes (6) and (8) would be singular in cases where $|\nabla\phi_j^n|$ is close to zero. Thus, when $|\nabla\phi_j^n| < D\Delta t^\alpha$, the schemes are replaced by

$$\phi_j^{n+1} = \frac{1}{|\mathcal{N}_j|} \sum_{\vec{x}_i \in \mathcal{N}_j} I[\Phi^n](\vec{x}_i) \tag{10}$$

where $\mathcal{N}_j$ is the set of the first neighbours of $\vec{x}_j$ in the Cartesian grid $\mathcal{G}$ and $|\mathcal{N}_j|$ represents its cardinality. In all computations presented in this work, we set $D = 10^{-3}$ and $\alpha = 1$.

In order to fully detail the numerical algorithm, a suitable interpolation operator has to be specified.

3.2 Interpolation

An important issue for the accuracy and the efficiency of the numerical scheme is clearly represented by the choice of the interpolation operator $I[\Phi^n]$. In [12], the authors employ a

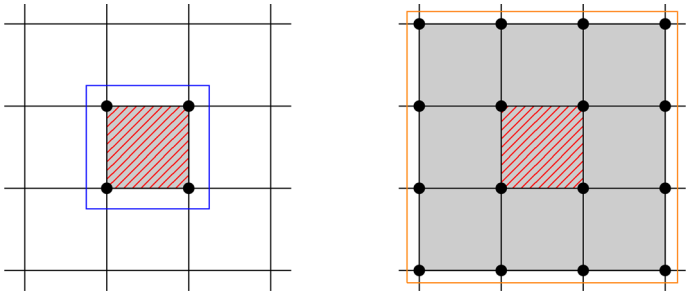

Fig. 1 Stencils of the two-dimensional Q1 and WENO reconstructions. The red hatched region represents the cell Ω in which we compute the reconstruction. The multi-linear interpolator only requires the vertices of the cell Ω, enclosed by the blue square on the left. On the right, the WENO interpolator involves the cell vertices and their first neighbours, enclosed in the orange square (Color figure online)

global RBF interpolation of the data on $\mathcal{G}$ defined as

$$I_{\mathsf{RBF}}[\Phi^n](\vec{x}) = c_0(\Phi^n) + \vec{c}(\Phi^n)\cdot\vec{x} + \sum_{\vec{x}_i\in\mathcal{G}} \xi_i(\Phi^n)\psi(|\vec{x}-\vec{x}_i|),$$

where the function $\psi : \mathbb{R}^+ \to \mathbb{R}$ provides the radial term. The coefficients $c_0 \in \mathbb{R}, \vec{c} \in \mathbb{R}^n$ and ξ_i of the RBF interpolator are computed by imposing interpolation conditions at all (or a subset of) points of the grid ($I[\Phi](\vec{x}_k) = \phi_k$ for all $\vec{x}_k \in \mathcal{G}$) and the additional conditions $\sum_{\vec{x}_i\in\mathcal{G}} \xi_i = 0$, $\sum_{\vec{x}_i\in\mathcal{G}} \xi_i x_i = 0$, $\sum_{\vec{x}_i\in\mathcal{G}} \xi_i y_i = 0$, $\sum_{\vec{x}_i\in\mathcal{G}} \xi_i z_i = 0$. The interpolation is thus computed by solving, at each time step, a linear system with matrix of the form $\begin{bmatrix} B & P \\ P^T & 0 \end{bmatrix}$, where B is an $N \times N$ block ($N = |\mathcal{G}|$) and P is $N \times 4$ in the 3d case (resp. $N \times 3$ in the 2d case).

However, the computation of the RBF interpolator then becomes a bottleneck of the algorithm due to the solution of the linear system involved. Furthermore, the global linear term $c_0(\Phi^n) + \vec{c}(\Phi^n)\cdot\vec{x}$, i.e. the blocks P and P^T in the system, forms a strong coupling of all the equations which is difficult to handle for parallel runs as it requires a lot of inter-processor communications.

Here instead we resort to two different types of local interpolator with the aim of making the algorithm faster and practical for surface reconstructions on a finer grid, using parallel computing. The first one is a multilinear interpolator, while the second is a WENO interpolator. The stencils involved in the reconstruction procedures for the 2d case are shown in Fig. 1.

In this way, we can guarantee that the only communications in the SL scheme occur when the two (resp. four) interpolation points needed for the update (6) (respectively (8)) belong to a different processor than the one owning the point $\vec{x}_j$ (see Sect. 3.3) and not during the reconstruction procedure itself.

Multilinear Interpolator

This interpolator is simply the Q1 Finite Element interpolation on the grid $\mathcal{G}$. Let $\{\varphi_j\}_{\vec{x}_j\in\mathcal{G}}$ be the shape functions that are, in each cell, a tensor product of degree 1 polynomials in each space direction and such that $\varphi_j(\vec{x}_k) = \delta_{j,k}$. Then, for any function $v(\vec{x})$, we consider the interpolator

$$I_{\mathrm{Q1}}[v](\vec{x}) = \sum_{\vec{x}_j \in \mathcal{G}} v_j \varphi_j(\vec{x}) \tag{11}$$

where v_j denotes the point value of the function v at the point $\vec{x}_j$.

We remark that, once the voxel of the grid containing $\vec{x}$ is located, the summation above in the 2d case (respectively in the 3d case) involves only four (resp. eight) terms and can be locally computed.

We expect that this kind of interpolator will be computationally very cheap, but that it might lead to less accurate reconstructed surfaces since it is piecewise linear and cannot faithfully represent the curvature of the surface, thus all its detailed features. On the other hand, considering multiquadric or multicubic operators, while trying to improve the quality of the reconstruction, could pose serious stability issues in non-smooth regions. This is why we completely avoided to investigate these techniques and moved to ENO interpolators which are apt to prevent spurious oscillations by introducing a non-linear dependency from the data.

WENO Interpolator

In order to have a more accurate interpolation operator, we resort to higher order polynomials; the larger stencils involved, however, expose to the risk of introducing oscillations due to the presence of corner points in the data. We thus choose WENO techniques. With respect to classical WENO reconstructions employed in conservation laws, here the reconstruction point is arbitrary and not necessarily on a face of the grid.

The procedure for the construction of a high-order WENO interpolator follows the one described in [13]. For practical purposes, here we will give some details of this procedure focusing on the case of a third-order one-dimensional reconstruction in the cell $\Omega_j = [x_j, x_{j+1}]$, on a uniform Cartesian mesh of width Δx. Additionally, the description will be limited to the one-dimensional case, since, in the multi-dimensional case, one only needs to iterate this procedure by dimension.

When constructing a WENO interpolator on the interval $[x_j, x_{j+1}]$, one searches for a convex combination of low-degree polynomials, designed in such a way that a high-order reconstruction is computed in regions associated to smooth data, while non-oscillatory properties are guaranteed in presence of a discontinuity. The selection or blending of such polynomials P is performed in a non-linear way relying on oscillation indicators $\mathrm{OSC}[P]$, which are in general scalar quantities associated to a polynomial P, designed in such a way that $\mathrm{OSC}[P] \to 0$ under grid refinement, if P is associated to smooth data, and $\mathrm{OSC}[P] \asymp 1$ (i.e., the indicator is asymptotically a non-zero constant), in presence of a discontinuity within the stencil of P.

More in details, to construct a WENO interpolation of degree $2r-1$ on the interval $[x_j, x_{j+1}]$, we start considering the Lagrange polynomial built on the stencil $S = \{x_{j-r+1}, \ldots, x_{j+r}\}$, written in the form

$$Q(x) = \sum_{k=1}^{r} C_k(x) P_k(x), \tag{12}$$

where the C_k are polynomials of degree $r-1$ and the P_k are polynomials of degree r interpolating the point values v_l of a function v on the stencil $S_k = \{x_{j-r+k}, \ldots, x_{j+k}\}$, $k = 1, \ldots, r$. Then, to compute the reconstruction in a point $\hat{x} \in [x_j, x_{j+1}]$, we proceed as follows:

1. compute suitable regularity indicators

$$\mathrm{OSC}_k = \mathrm{OSC}[P_k], \quad k = 1, \dots, r; \tag{13}$$

2. define the coefficients

$$\alpha_k = \frac{C_k(\hat{x})}{(\mathrm{OSC}_k + \epsilon)^2}, \tag{14}$$

where $\epsilon = \Delta x^2$ prevents small denominators;

3. compute the nonlinear weights $\{w_k\}_{k=1}^r$ as

$$w_k = \frac{\alpha_k}{\sum_l \alpha_l}; \tag{15}$$

4. finally, define the reconstruction polynomial as

$$I[v](\hat{x}) = \sum_{k=1}^{n} w_k P_k(\hat{x}). \tag{16}$$

Differently from [13], we consider regularity indicators defined as

$$\mathrm{OSC}_k = \mathrm{OSC}[P_k] = \sum_{\beta \geq 2} \Delta x^{2\beta-3} \int_{x_j}^{x_{j+1}} \left(\frac{d^{(\beta)} P_k}{dx^\beta} \right)^2 \mathrm{d}x. \tag{17}$$

We point out that the above is the classical definition of the oscillation indicators for the WENO reconstruction as given in [34], except that the first derivative is not included in the sum. This choice is justified by the fact that the function that we want to interpolate can be at worst continuous with kinks.

In fact, let us consider a generic polynomial P_k of degree $r-1$ having a discontinuity in its first derivative in the stencil $\mathcal{S}_k$. When we consider its β-derivative, for $\beta \geq 2$, we get a polynomial of degree $r-\beta-1$ whose coefficients of degree $m = 0, \dots, r-\beta-1$ are given by the Newton divided differences with $m+\beta+1$ points that scale as $\mathcal{O}(\Delta x^{1-m-\beta})$, due to the discontinuity. Now, computing the integral in (17), we get a sum of terms, each one scaling at the same rate, given by $\mathcal{O}(\Delta x^{3-2\beta})$. It is thus clear that the exponent $2\beta - 3$ is apt to get $\mathrm{OSC}[P_k] \asymp 1$ in this non-regular case. In case of smoothness, instead, the Newton divided differences would be $\mathcal{O}(1)$ and $\mathrm{OSC}[P_k] \to 0$ as expected.

Now, turning back to our specific case, we need to find a proper expression for the linear weights C_k in a one-dimensional framework and considering $r = 2$ to get, at least in the best case, a third-order reconstruction. Since $r = 2$, the whole stencil is given by $S = \{x_{j-1}, x_j, x_{j+1}, x_{j+2}\}$ and the only two substencils to consider are $S_L = \{x_{j-1}, x_j, x_{j+1}\}$ and $S_R = \{x_j, x_{j+1}, x_{j+2}\}$. Since the polynomials P_L and P_R interpolate the function v respectively on S_L and S_R, each C_k should vanishes outside S_k, namely $C_L(x_{j+2}) = 0$ and $C_R(x_{j-1}) = 0$, and the condition $\sum_{k=L,R} C_k(x_i) = 1$ must hold for every node $x_i \in S$. Thus, we have two conditions for each polynomial C_k, which, in our case leads to a unique definition of these one degree polynomials that we can write in the form

$$C_k(x) = \gamma_k \frac{x - x_k}{\Delta x}, \tag{18}$$

where $k = L, R$ and $x_k \in S \setminus S_k$.

To get the coefficients γ_L and γ_R we start considering the first node of the stencil x_{j-1}, on which we have $C_L(x_{j-1}) = 1$ and $C_R(x_{j-1}) = 0$, so that

$$Q(x_{j-1}) = C_L(x_{j-1}) P_L(x_{j-1}) = C_L(x_{j-1}) v_{j-1}, \tag{19}$$

thus inferring that $C_L(x_{j-1}) = 1$ and therefore, using (18) with $x_k = x_{j+2}$,

$$\gamma_L = -\frac{1}{3}. \tag{20}$$

Analogously, we can proceed considering the node x_j for which we must have $C_L(x_j) + C_R(x_j) = 1$. Since the expression for C_L is known, we can compute

$$C_R(x_j) = 1 - C_L(x_j) = 1 - \frac{2}{3} = \frac{1}{3} \tag{21}$$

and therefore, from $\frac{1}{3} = \gamma_R[(x_j - x_{j-1})/\Delta x]$ in (18), we get

$$\gamma_R = \frac{1}{3}. \tag{22}$$

Summarizing, we have the following expressions:

$$C_L(x) = \frac{x_{j+2} - x}{3\Delta x}, \quad C_R(x) = \frac{x - x_{j-1}}{3\Delta x}. \tag{23}$$

It only remains to find a proper expression of the regularity indicators OSC_L and OSC_R for the two degree polynomials P_L and P_R. Let us consider a polynomial of degree at most two written in the form $P(x) = \sum_{i=0}^{2} a_i[(x - x_j)/\Delta x]^i$. Its indicator is a quadratic form of its coefficients and is given by

$$\mathrm{OSC}[P] = \frac{1}{\Delta x^2} 4a_2^2$$

or equivalently, denoting by V the vector of data $(v_{j-1}, v_j, v_{j+1}, v_{j+2})^T$, since the coefficients of the polynomial linearly depends on V, we can express the regularity indicators as a quadratic form of the data being interpolated. For our specific case we obtain

$$\mathrm{OSC}[P_k] = \frac{1}{\Delta x^2} V^T A_k V \quad (k = L, R),$$

with

$$A_L = \begin{pmatrix} 1 & -2 & 1 & 0 \\ -2 & 4 & -2 & 0 \\ 1 & -2 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad \text{and} \quad A_R = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 \\ 0 & -2 & 4 & -2 \\ 0 & 1 & -2 & 1 \end{pmatrix}.$$

Finally, we illustrate in some details the two-dimensional reconstruction. Let $(\hat{x}, \hat{y})$ be the reconstruction point, located in the cell $[x_i, x_{i+1}] \times [y_j, y_{j+1}]$. One first performs four one-dimensional WENO interpolations to compute auxiliary data $w_{i-1}, w_i, w_{i+1}, w_{i+2}$. Each w_k is the interpolation in the y direction of the data $v_{k,j-1}, \ldots, v_{k,j+2}$ evaluated at $\hat{y}$. Finally a WENO interpolation in the x direction of $w_{i-1}, w_i, w_{i+1}, w_{i+2}$ evaluated at $\hat{x}$ will be the reconstructed value. The three-dimensional case can be treated analogously.

Note that, due to the dimensional splitting employed in the two and three dimensional procedures, the WENO interpolator, as the RBF and the Q1 ones, is globally continuous on the edges of the cell in which the reconstruction is performed. Moreover, it is more local than RBF and it is expected to represent curved surfaces more faithfully than Q1.

3.3 Domain Decomposition for Parallel Runs

Especially three-dimensional computations suggest the use of parallel computing for reducing the execution time. Here we recall how the computational grid has been organized for parallel

implementation and what are the main communications between workers that might occur during the execution.

As usual for Cartesian grids, in our parallel implementation, each rank m owns a rectangular portion G_m of the computational grid $\mathcal{G}$ such that $\mathcal{G} = \cup_m G_m$, and is responsible to update the solution on it. Since the computation of the reconstruction and of the time update might require knowledge of the solution at the previous time step in a neighbourhood of G_m, each rank also keeps a local copy of the data in a halo region $\widetilde{G}_m$ of at least one ghost point per direction around G_m. The width of this halo region depends on which information are needed to perform all the computations in the algorithm. The halo data in $\widetilde{G}_m$ have to be synchronized between workers during the computations through inter-process communications.

In the case of a SL scheme, the main sources of communication arise from: first, the interpolation of Φ^n at feet of the characteristics emanating backwards from the nodes $\vec{x}_j$ and specified in equations (6) and (8); second, the stencil width required for the computation of the interpolant. This latter is mitigated by our choice of local interpolation techniques with very small stencils, like those of Sect. 3.2: in both cases we are able to run with one ghost-cell per direction for the Q1 reconstruction and two ghosts per direction in the WENO case.

The former task is facilitated by the Cartesian structure of the grid and of its partitioning scheme. Consider any of the points $\vec{x}^*_{j,i}$ appearing in (6) or (8) associated to a node $\vec{x}_j$ owned by the processor m. First $\vec{x}^*_{j,i}$ is located on the grid partition and let m^* be the processor owning the cell into which it falls. If $m^* = m$, the interpolation $I[\Phi^n](\vec{x}^*_{j,i})$ is computed without the need of any communication. Otherwise, a request to the processor m^* is sent, communicating the point $\vec{x}^*_{j,i}$ and receiving in response the value of $I[\Phi^n](\vec{x}^*_{j,i})$. Of course all these communications are gathered in a single push.

In our implementation also the point cloud $\mathcal{S}$ is distributed among the workers. Some more communications are needed in the auxiliary routines described in the next sections and will be highlighted therein.

3.4 Distance Function

In the SL schemes (6) and (8) one needs to evaluate the distance $d(\vec{x}) = \min_{Q\in\mathcal{S}} |\vec{x} - Q|$ from the dataset $\mathcal{S}$ at every grid point. We remark that a direct computation would have computational complexity proportional to $N \times |\mathcal{S}|$, where N is the number of points in the computational grid and $|\mathcal{S}|$ the size of the point cloud. It would also require an overwhelming amount of communications in parallel runs, when both the grid and the point cloud are distributed among different workers.

However, one acknowledges that accurate values of the distance function are needed only close to the object surface to which $\mathcal{S}$ belongs. In practice we initially set $d(\vec{x}_j)$ to the exact distance from $\mathcal{S}$ on all nodes of $\mathcal{G}$ that are in a box of 4×4 or $4 \times 4 \times 4$ grid points around each point $Q \in \mathcal{S}$. Then, we apply the fast sweeping method of [52] to compute approximate values for $d(\vec{x})$ at all other grid points in $\mathcal{G}$. These approximate values are still good enough to drive the evolution of the surface Γ towards $\mathcal{S}$, while the final stages of the evolution will be guided by the exact values of the distance set in the neighbourhood of the points in $\mathcal{S}$.

3.5 Energy Functional

Applying the schemes (6) and (8) with $p > 1$ also requires 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 $\mathcal{G}_0$ composed by the cells in $\mathcal{G}$

where, at a specific time step, the front is located. Further, we assume that the function ϕ is, at least locally around $\mathcal{G}_0$, a signed distance so that $|\nabla\phi| = 1$. This latter point is ensured by the reinitialization procedure described in Sect. 3.7. We thus compute the energy as

$$E_p(\phi) \approx \left(\sum_{\Omega \in \mathcal{G}_0} \int_{\Omega} |d(\vec{x})|^p \mathrm{d}\vec{x} \right)^{1/p}. \tag{24}$$

To detect the cells in $\mathcal{G}_0$, we check the values of ϕ on their vertexes and consider only the cells across which the function ϕ changes sign. Once the front is located in a cell Ω, different strategies can be considered to compute (24).

- In two space dimensions, we identify two points intercepted by the front of ϕ on the cell boundary, by the linear interpolation of ϕ on the edges. Then the trapezoidal rule is used to approximate the integral over the segment connecting them. This approach would be much more involved in the three-dimensional case since one would need to identify the (two-dimensional) intersection of the zero level set surface with the cube Ω and this could be placed in a general position.
- In three space dimensions, to avoid the aforementioned complications, we further approximate the Dirac δ function within a single volume $\Omega \in \mathcal{G}_0$ by considering a local refinement of the grid element with $\Delta x' = \Delta x / R$ to detect the smaller subcells $\Omega' \in \mathcal{G}_0'$ containing the front, and consider the approximation

$$E_p(\phi) \approx \left(\sum_{\Omega' \in \mathcal{G}_0'} |d(\vec{x}')|^p (\Delta x')^2 \right)^{1/p}, \tag{25}$$

  where $d(\vec{x}')$ is the interpolated value of the distance function at the center of the subcell Ω'. Instead of detecting the subcells in $\mathcal{G}_0'$ by interpolating ϕ on the vertices, we avoid the huge amount of evaluations checking if

$$|\phi(\vec{x}')| < \frac{\sqrt{3}}{2} \Delta x', \tag{26}$$

  where x' and $\Delta x'$ are the center and the edge length of the subcell Ω', respectively. We employ this approach with $R = 5$.

3.6 Initial Data

The initial data should be chosen as an approximation of the signed distance function from the data set $\mathcal{S}$. Obviously, the better the initial data, the more efficient the method will be, but one has to take into account also the computational effort spent in the computation of the initial datum itself. In practice, we compute ϕ^0 from the distance function, similarly to [54], to obtain a signed distance function whose zero level surface encompasses the point cloud and is as close as possible to it.

In particular we start by the approximate distance function computed as in Sect. 3.4. First, the cells on the outer boundary of $\mathcal{G}$ are marked as external. Then, moving inwards from all boundaries in all Cartesian directions, we propagate the external point marking to nearby cells until we find a point for which $d(\vec{x}_j) < \gamma_{\mathcal{S}}$, where $\gamma_{\mathcal{S}}$ is a suitable threshold related to the resolution of the point cloud.

After external regions have been so identified, ϕ^0 is provisionally set to $d(\vec{x}_j) - \gamma_{\mathcal{S}}$ on external points and to an artificially high value otherwise. Next, the fast sweeping method

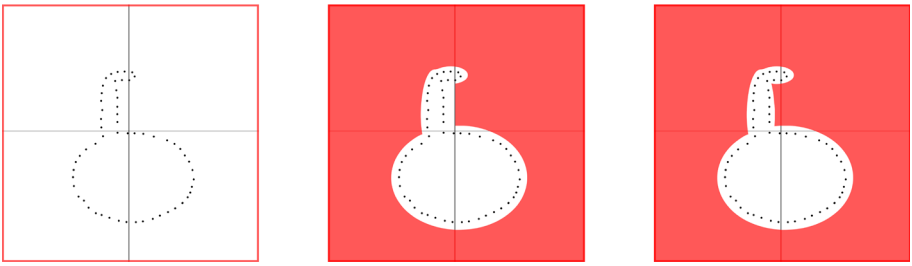

Fig. 2 Illustration of marking external points in parallel run with a 2×2 domain decomposition (grey lines). Left: initial state. Center: after the first sweep. Right: after the second sweep

of [52] is applied to ϕ^0, recomputing its values at internal points; finally, the sign of ϕ^0 is reversed on internal points.

We point out that in parallel runs the procedure for initial marking of external points is similar to the serial run, but is iterated more times and interleaved with communications of point marking in the halo regions. This is illustrated in Fig. 2. The top-left processor, in the first run (middle panel), expands the marking in the right and bottom direction from the outer border of the computational grid and is thus unable to mark the small area under the hook. In the second sweep (right panel), the external marking is propagated from the marking done by the top-right processor. A number of iterations of at least half the number of processors per direction in the decomposition ensures that the initial marking obtained is the same that would have been computed in a serial run.

The one described above, is a good strategy to find an initial data and start the evolution having only the information carried by the point cloud. As a matter of fact, one could also choose to evolve a less accurate initial guess until steady-state on a coarse grid and then use this final level set function as the initial data for a finer evolution.

3.7 Reinitialization

One of the nice feature of using a signed distance function to capture interfaces is that some geometric quantities, such as normals and curvatures, are easier to compute in terms of ϕ. In particular this is of paramount importance when the level set is used in ghost-fluid algorithms for the discretization of PDEs. The evolution determined by (5) will push the solution towards the data set, but will create sharp gradients in the level set function ϕ around it. To keep $|\nabla\phi|$ close to 1, we employ a reinitialization procedure, first introduced in [49], which consists in evolving until steady-state the equation

$$\begin{cases} \phi_\tau + \text{sign}(\tilde{\phi})\,(|\nabla\phi| - 1) = 0 \\ \phi(\vec{x}, 0) = \tilde{\phi}(\vec{x}) \end{cases} \tag{27}$$

where the initial value $\tilde{\phi}$ corresponds to the solution of (5) at a fixed time step and τ is a pseudo-time.

The delicate issue with the reinitialization is to guarantee that the interface of $\tilde{\phi}$ is displaced as little as possible, ideally not at all, while the level set function is modified to ensure that $|\nabla\phi| = 1$. To this aim, we have followed the constrained reinitialization (CR) scheme introduced by Hartmann et al. [31] which consists in a modification of the first scheme by Sussman et al. [40] aiming to locally preserve the location of the interface during the

reinitialization procedure. The scheme proposed in [31] has been demonstrated to perform better than the previous ones and we prefer to adopt it in order to prevent the zero level set to be modified when solving (27).

With this approach a one step procedure is performed to reinitialize the level set function in the first neighbours of the interface, which can be easily detected from the change of sign of $\tilde{\phi}$. After that, an iterative procedure is performed to reinitialize the values of $\tilde{\phi}$ in a proper tube around the already reinitialized region; to this end a first-order spatial discretization and forward Euler integration in pseudo-time τ is used to approximate the solution of the original equation (27). In parallel runs, each step of this algorithm also requires an update of the ϕ data in the halo region.

3.8 Localizing the Computational Effort

A typical approach when dealing with level set methods is to localize the evolution only in a narrow band around the zero level set of ϕ. Since it is not important to update ϕ far away from that, at each time step we choose a subgrid

$$\widetilde{\mathcal{G}} = \{\vec{x}_j \in \mathcal{G} : \phi_j^n < \gamma\} \subset \mathcal{G}, \tag{28}$$

with $\gamma = 4\Delta x$, which, according to [42], is apt for the spatial reconstructions employed in this work (multilinear and third-order WENO), and update ϕ^{n+1} only therein. Outside $\widetilde{\mathcal{G}}$ we simply cut our level set function as

$$\phi(\vec{x}) = \begin{cases} \gamma & \text{if } \phi(\vec{x}) > \gamma, \\ \phi(\vec{x}) & \text{if } |\phi(\vec{x})| \le \gamma, \\ -\gamma & \text{if } \phi(\vec{x}) < -\gamma. \end{cases} \tag{29}$$

To prevent numerical oscillations at the boundary of $\widetilde{\mathcal{G}}$, we also update the solution involving an additional cut-off function $c(\phi)$, the same described in [42], which is given by

$$c(\phi) = \begin{cases} 1 & \text{if } |\phi| \le \beta, \\ (|\phi| - \gamma)^2(2|\phi| + \gamma - 3\beta)/(\gamma - \beta)^3 & \text{if } \beta < |\phi| \le \gamma, \\ 0 & \text{if } |\phi| > \gamma, \end{cases} \tag{30}$$

with $\beta = 2\Delta x$ ([42]), thus considering the modified equation

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

and properly modifying the semi-Lagrangian schemes (6) and (8) to incorporate the additional factor $c(\phi)$.

Finally, the reinitialization procedure is performed on a different narrow band $\overline{\mathcal{G}}$ obtained by considering $\widetilde{\mathcal{G}}$ and its first neighbours. The fact that $\overline{\mathcal{G}} \supset \widetilde{\mathcal{G}}$ is essential if one starts the algorithm with an initial data so far from the point cloud that the initial computational band (28) 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 $\widetilde{\mathcal{G}}^0$, while with this choice, the successive bands $\widetilde{\mathcal{G}}^n$ will be able to move contextually with the zero level set of ϕ^n. For more clarity, the bands involved in the different parts of the scheme are depicted in a 2d case in Fig. 3 and a typical evolution of a front of the level set function in 1d is depicted in Fig. 4.

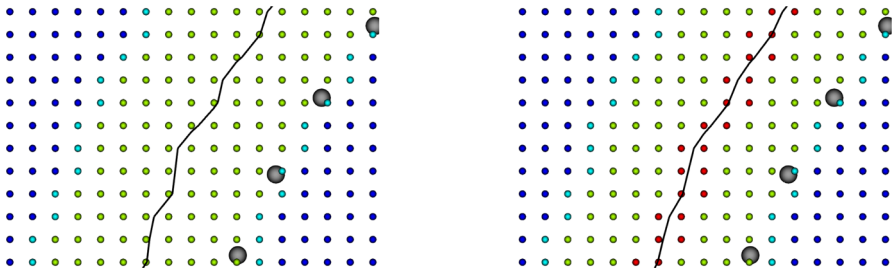

Fig. 3 On the left, the mask computed to detect the computational subgrid $\widetilde{\mathcal{G}}$. Active nodes are depicted in green, their first neighbours are depicted in light blue (they are inactive during the update, while they are active during the reinitialization step) and remaining inactive nodes are depicted in blue. On the right, the mask computed to detect the computational subgrid $\overline{\mathcal{G}}$ of the reinitialization. Colors are used in the same way, with in addition red nodes representing the nodes immediately close to the interface on which the signed distance function is computed explicitly with the one step procedure (Color figure online)

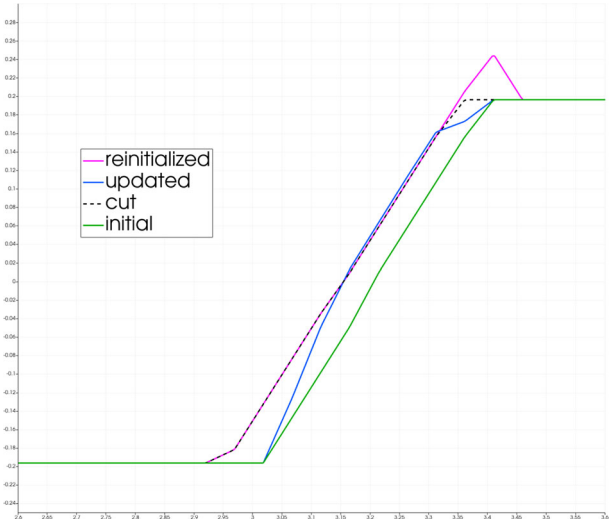

Fig. 4 Typical evolution of a front in 1d. Starting from the green line, the update makes the front move towards the data resulting in the blue line, sharper than the green one. Reinitialization (pink line) makes the computational narrow band move contextually with the front, fixing the zero level set. The dotted black line represents the last cutting step necessary to obtain the final update (Color figure online)

4 Numerical Experiments

In this section we present some representative numerical results obtained with the algorithm described in the previous sections. In order to compare easily the different $\mathcal{S}$, we have rescaled all the point clouds in the box $[-1, 1]^n$.

As we are looking for a minimum of the energy functional (1), we will consider the same stopping criterion used in [32]: at time n, the algorithm stops if

$$\Delta_E^n = \frac{|\bar{e}_{n-1}^k - \bar{e}_n^k|}{\bar{e}_n^k} < 10^{-4}, \quad \text{where} \quad \bar{e}_n^k = \frac{1}{k} \sum_{i=n-k+1}^{n} E_2(\phi^i) \tag{31}$$

Table 1 Standard values for p and μ in each run r of Algorithm 1

r	p	μ
1	1	0.05
2	2	0.05
≥ 3	2	1

or after a maximum of 100 iterations. Note that in (31), the energy functional (1) is computed with $p = 2$. This condition is in practice a way to detect stationary points or flat areas of the energy functional. In all the tests we set $k = \min(n, 10)$ and forced the algorithm to do at least 10 iterations.

We point out that some other stopping criteria can be considered. In [54] the author propose to stop the evolution either all data points are close enough to the zero level set, i.e. $|\phi(Q)| < tol$ for any $Q \in \mathcal{S}$ or $|\phi_t|$ is small enough, meaning that we are close to an equilibrium state. Alternatively, the mean of squared differences of two subsequent time steps is tested in [30] is order to get a right choice of the number of time steps for the model creation.

4.1 Quantitative Evaluation

In order to get a quantitative insight into the quality of the reconstruction achieved with our scheme and compare it with already existent methods, different quantities are considered. In all the tests we compute, alongside (31), the normalized L^1-norm of the update between two successive iterates, namely

$$\epsilon_1^n = \frac{\sum_{\vec{x}_j \in \mathcal{G}} |\phi_j^{n+1} - \phi_j^n|}{\sum_{\vec{x}_j \in \mathcal{G}} |\phi_j^n|}, \tag{32}$$

as in [12], and and the L^1-norm of the error, when the exact signed distance function ϕ^* is given.

Furthermore, we compute the average of the error on the points of the cloud

$$Err_{\mathcal{S}}^n = \frac{\sum_{Q \in \mathcal{S}} |I[\Phi^n](Q)|}{|\mathcal{S}|}, \tag{33}$$

to make some considerations on the role of the curvature regularization and to evaluate how much the final reconstruction is attached to the data set.

4.2 Choice of Parameters

It only remains now to detail the choices for the parameters p and μ and for the spatial and temporal discretization.

In [12], the authors consider the case with $p = 1$, which grants for a faster evolution of the initial data towards the cloud and also does not require to compute the factor $[d(\vec{x})/E_p]^{p-1}$, while in [54] the authors suggest to set $p = 2$, which is more effective in reaching a steady state for the evolution. Also, setting $\mu = 0$ simplifies the update formulas (6) and (8) since all $\vec{x}_{j,i}^*$ coincide. Furthermore, disregarding the curvature effect, it prevents from loosing too

much details, especially on coarse grids. On the other hand, increasing values of $\mu \geq 0$ will smoothen the solution and will help handling changes of topology, at the price of having a zero level set slightly off from the cloud $\mathcal{S}$.

Our goal is to compute a levelset with $p = 2$ on a fine grid. In order to save computational time, we combine different choices of the parameters in more than one run of Algorithm 1, gradually increasing the resolution of the grid.

Let r represent the number of the run of Algorithm 1, Table 1 summarizes the usual approach. We point out that $\mu = 0.05$ is chosen accordingly to the analysis made in [30] and also with the aim of better handling possible changes in topology, while $\mu = 1$ is chosen in order to reproduce the original model. On the other hand, we will see that the choice $\mu > 1$ might be useful when the dataset is affected by noise in order to further smoothen the solution.

Regarding the spatial and temporal discretization we set

$$\Delta x^{(r)} = \frac{C_{\Delta x}}{2^{r-1}} h_{\mathcal{S}}, \quad \Delta t^{(r)} = \Delta x^{(r)}, \tag{34}$$

where r is the number of the current run, $h_{\mathcal{S}}$ is specified below, and, unless otherwise specified, $C_{\Delta x} = 1$. The above spatio temporal discretization clearly states the advantages of using the SL approach since in (34) we set $\Delta t = \mathcal{O}(\Delta x)$, not being prohibitively constrained by the parabolic term. In the usual procedure three runs of Algorithm 1 are performed.

In (34) $h_{\mathcal{S}}$ represents an estimate of the resolution of the cloud $\mathcal{S}$; its value is approximated by randomly choosing a sample made up of the 10% of the points in $\mathcal{S}$ and then computing the average of the distances between each of these points and their nearest neighbour in $\mathcal{S}$.

In the first and coarsest run ($r = 1$), we compute the initial data as described in subsection 3.6. The required threshold is set as $\gamma_{\mathcal{S}} = K_{\mathcal{S}} h_{\mathcal{S}}$ with $K_{\mathcal{S}}$ set to 2.0, unless otherwise specified. The role of $K_{\mathcal{S}}$ is relevant when the points in $\mathcal{S}$ are not evenly distributed on the sampled surface. It is in fact common, especially in 3d, due to the supports used for the object during laser scanning, to find data sets that have piggy bank-like shapes, which have fake holes that could distort the result. In such cases, the good practice consists in choosing a larger $K_{\mathcal{S}}$, to start with level set that is further away from the data, but encloses the point cloud without entering in the fake cavities. For the later runs ($r > 1$), we use as initial data the final solution computed by the previous run and interpolated on the current grid.

All the codes have been written in C++ language with the support of the MPI and of the PETSc library [4, 5]. In particular, the communications in the SL step 4c of Algorithm 1 have been performed with the help of a DMSWARM object. The tests have been performed on the Galileo100 cluster hosted at CINECA.[1]

4.3 2d Data Sets

4.3.1 Circle Test

We first consider a data set $\mathcal{S}$ made up of 64 points uniformly chosen on a circle of radius 1, thus having a cloud size approximately equal to 9.81×10^{-2}.

The main steps of the reconstruction of the circle are depicted in Fig 5, while Fig 6 collects significant graphs to evaluate the accuracy of the method. The three runs of the algorithm required respectively 18, 20 and 12 iterations using the Q1 reconstruction, while 20, 13 and

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

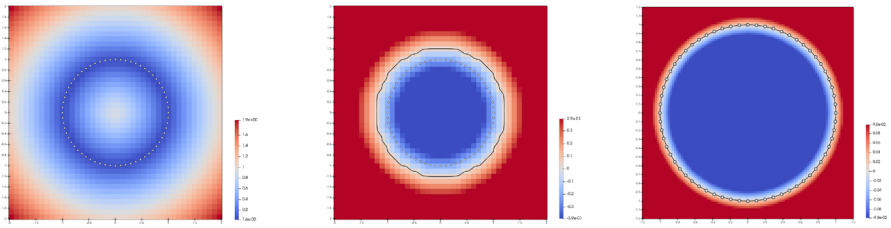

Fig. 5 Steps of the algorithm for the 2d circle case: on the left, the distance function is represented; the central panel shows the initial data and its zero level set (black line); the final data and its contour (black line) are represented on the right. Data represented here has been obtained with WENO reconstruction

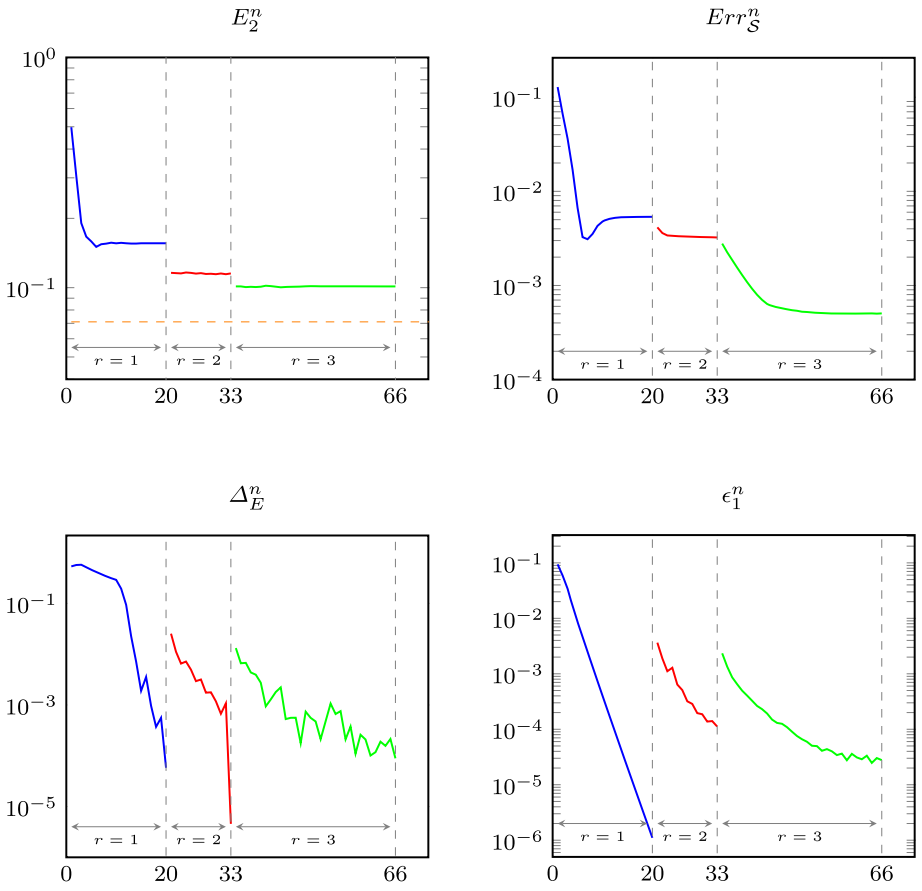

Fig. 6 Significant quantities computed for the circle test using WENO interpolator. In the top-left panel the energy functional is depicted: the drop between two successive runs is due to the change of the grid size and the consequent new computation of the distance function. In the top-right panel the error computed on the cloud is shown: as expected, its profile is not monotone since during the evolution the interface can even pass the cloud and then come back. The bottom-left panel represents the running average of the energy functional: note that in the first 10 iterations of each run the moving average is still forming up. The bottom-right panel represents the normalized L^1-norm of the update between two successive iterates

Table 2 Errors and energy computed for the circle test at the end of each run. To better appreciate the results Algorithm 1 has been performed five times, fixing the parameters according to Table 1. Note that, the better results on WENO, especially in terms of the error computed on the cloud $\mathcal{S}$, are linked to a greater ability to be faithful to the data set and to capture details

		Q1			WENO		
r	Grid	L^1-err	Energy	Error on $\mathcal{S}$	L^1-err	Energy	Error on $\mathcal{S}$
1	42×42	$8.01e-02$	$1.76e-01$	$1.29e-02$	$7.49e-02$	$1.57e-01$	$1.20e-02$
2	58×58	$9.74e-03$	$1.19e-01$	$3.10e-03$	$1.01e-02$	$1.15e-01$	$3.86e-03$
3	99×99	$1.71e-03$	$1.04e-01$	$6.43e-04$	$1.48e-03$	$1.01e-01$	$5.05e-04$
4	181×181	$6.80e-04$	$1.04e-01$	$5.06e-04$	$6.29e-04$	$1.03e-01$	$4.61e-04$
5	344×344	$2.73e-04$	$1.04e-01$	$3.77e-04$	$2.48e-04$	$1.04e-01$	$3.19e-04$

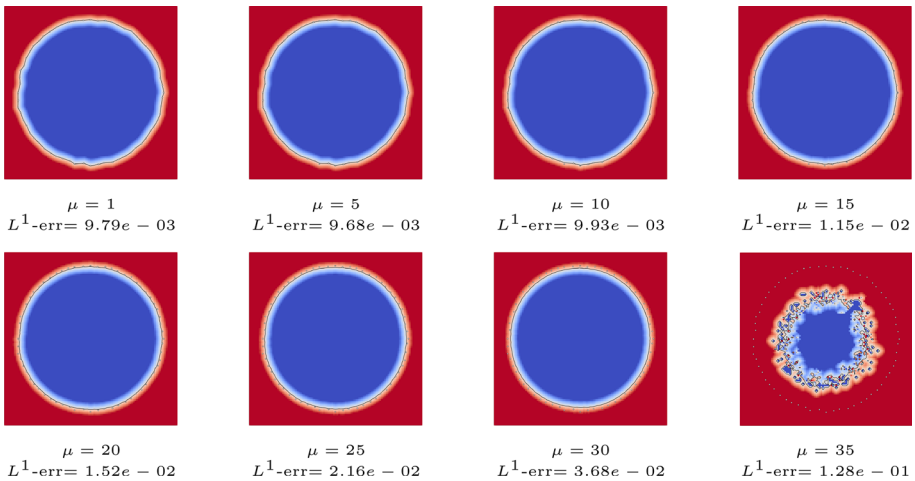

Fig. 7 Reconstructions form noisy data. A 10% of normally distributed noise is added to $\mathcal{S}$ and increasing values of μ in the final run are considered. The solution keeps smoothing, until it degenerates ($\mu = 35$)

33 iterations are required in the WENO case, with an amount of computational time equal to $9.40e-02$ and $1.31e-01$ seconds, respectively.

Looking at Fig. 6 note how the zero level set approaches the cloud: as expected, the normalized L^1-norm of the update does not show a monotone profile and the error on the cloud $Err^n_{\mathcal{S}}$ is not vanishing due to the finite grid size and to the curvature term. During the first run, the interface quickly moves towards the data, it can even pass through the cloud, and then stabilizes. During the successive runs the reconstruction becomes more accurate: the energy gets closer to the exact value of the energy computed for a circle, represented by the dashed orange line ($\approx 7.10 \times 10^{-2}$), and the error made on the cloud decreases. In this case, the exact signed distance function from the circle is known and we can compute the L^1-norm of the error (see Table 2).

Finally, we perform the circle test adding a normally distributed noise to the data (10%). Figure 7 shows the WENO reconstructions corresponding to increasing values of μ in the last run: for $\mu = 5$ we have the best result in term of L^1 error, while for larger values of μ we get a smoother solution, but a larger L^1 error due to the shrinking of the level sets, until, for $\mu = 35$, the zero level set completely degenerates.

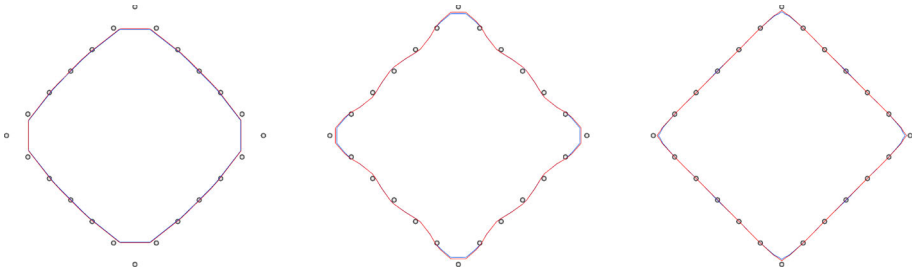

Fig. 8 From left to right: a comparison between the curves reconstructed with Q1 (blue line) and WENO (red line) interpolators at each run (Color figure online)

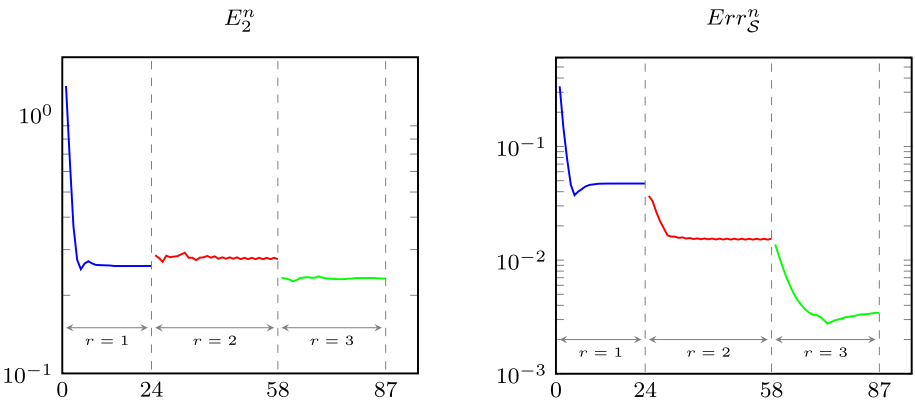

Fig. 9 Energy and error on cloud computed for the square test using WENO interpolator

4.3.2 Square Test

In the second test we consider a cloud made up of 24 points representing a square rotated by 45 degrees with respect to the Cartesian axis. The reconstruction takes respectively 23, 26, 14 iterations and $1.08e-01$ seconds in the Q1 case, and 24, 34, 29 iterations and $1.50e-01$ seconds in the WENO case. The final curves of each run are depicted in Fig. 8. Graphs of the energy functional and of the error on the cloud are reported in Fig. 9, exclusively for the WENO case. It is worth highlighting the role of μ: the curvature regularization in the last run penalizes the sharp corners, thus providing a final contour with a controlled maximum curvature, which, although the higher resolution of the grid, is not passing exactly through the corresponding points of the cloud.

Table 3 shows the errors computed with respect to the exact data. We point out that in this test the WENO based algorithm leads to significantly lower errors, due to the differences near the corners.

4.4 Synthetic 3d Data Sets

In this subsection we present numerical tests for the reconstruction of 3d shapes. Here the data sets are synthetic and are made up by sampling points on simple geometrical forms for which the exact signed distance function is known.

Table 3 Errors and energy computed for the square test at the end of each run. To better appreciate the results, Algorithm 1 has been performed five times

		Q1			WENO		
r	Grid	L^1-err	Energy	Error on $\mathcal{S}$	L^1-err	Energy	Error on $\mathcal{S}$
1	30×30	$1.01e+00$	$3.76e-01$	$9.44e-02$	$9.09e-01$	$3.33e-01$	$8.07e-02$
2	34×34	$1.18e-01$	$2.65e-01$	$3.08e-02$	$7.71e-02$	$2.96e-01$	$1.69e-02$
3	51×51	$1.27e-02$	$2.39e-01$	$4.71e-03$	$9.69e-03$	$2.32e-01$	$3.43e-03$
4	85×85	$4.30e-03$	$2.38e-01$	$7.10e-03$	$3.01e-03$	$2.37e-01$	$4.37e-02$
5	153×153	$1.45e-03$	$2.42e-01$	$5.95e-02$	$1.04e-03$	$2.42e-01$	$4.16e-02$

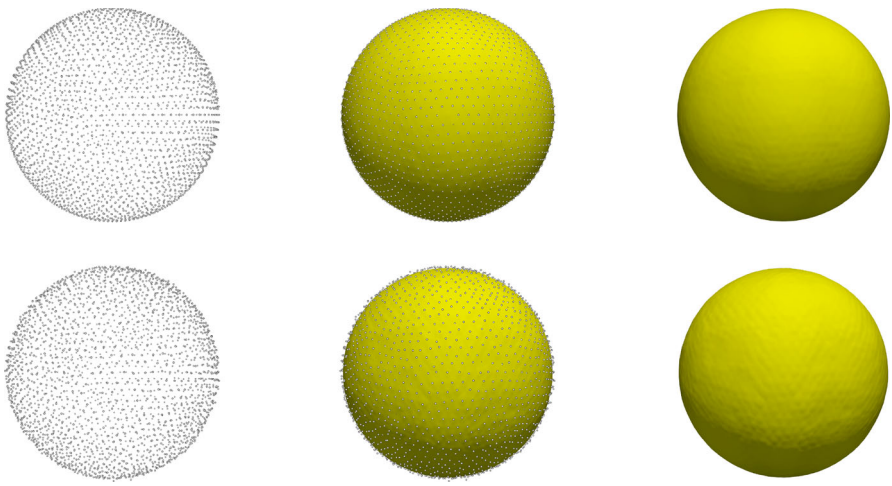

Fig. 10 Reconstruction steps of a sphere for the WENO case. In the first row the point cloud and the final zero surface with and without the point cloud are depicted for the unperturbed case. In the second row the same figures are produced adding a 10% of noise to the dataset and setting $\mu = 10$ in the last run in order to smoothen the solution

4.4.1 Sphere

The first 3d test has been performed on a point cloud made up of 2562 points chosen on a sphere of radius 1. The reconstruction procedure is illustrated in Fig. 10 for the WENO case with the original data set (first row) and a perturbed one (second row, adding 10% of normally distributed noise). In this simple case it is worth emphasizing that the role of the curvature regularization is to provide a smoother final surface than the ones provided by the previous two runs. In fact, setting $\mu = 10$ in the last run of the noisy case, we aimed at smoothing the final surface that would have been sensibly rough with the usual setting of the parameter. Looking a Table 4 one can also compare the results obtained with different interpolators and observe the better performance of the WENO interpolator.

4.4.2 Cube & Spheres

The second test concerns a three-dimensional domain composed by a cube joined with three spheres, sampled by 2346 points. The cube edge length was 0.8, the first sphere has radius

Table 4 Errors and energy computed for the sphere at the end of each run

	Q1			WENO		
r	L^1-err	Energy	Error on $\mathcal{S}$	L^1-err	Energy	Error on $\mathcal{S}$
1	$5.56e-02$	$2.12e-01$	$4.60e-03$	$4.97e-02$	$1.90e-01$	$4.05e-03$
2	$9.02e-03$	$1.74e-01$	$1.81e-03$	$8.73e-03$	$1.71e-01$	$1.93e-03$
3	$3.65e-03$	$1.61e-01$	$1.56e-03$	$3.27e-03$	$1.57e-01$	$1.37e-03$

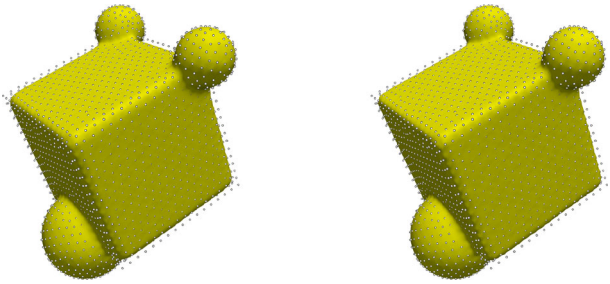

Fig. 11 The reconstructed surfaces of the "Cube&Spheres". On the left, the result obtained with the Q1 interpolant, while on the right, the result obtained using the WENO interpolant

Table 5 Errors and energy computed for the Cube&Spheres at the end of each run

	Q1			WENO		
r	L^1-err	Energy	Error on $\mathcal{S}$	L^1-err	Energy	Error on $\mathcal{S}$
1	$9.86e-02$	$1.82e-01$	$1.05e-02$	$9.34e-02$	$1.66e-01$	$8.92e-03$
2	$2.90e-02$	$1.55e-01$	$4.64e-03$	$2.85e-02$	$1.53e-01$	$4.18e-03$
3	$1.34e-03$	$1.46e-01$	$4.91e-03$	$1.30e-02$	$1.42e-01$	$3.95e-03$

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 Cartesian grid.

Results are shown in Fig. 11 and in Table 5. As for the 2d case of the square, the curvature regularization provides rounded edges and corners in the final reconstruction, especially in the Q1 case. Here too the WENO reconstruction shows better performances according to all three measures and especially to the error on the point cloud. Plot of the energy functional and of the error on the cloud are shown in Fig. 12 for the WENO case. Note in particular that the minimum of the error on $\mathcal{S}$ is reached at the end of the second run and is about $4.64e-03$, while at the end of the third run the error is about $4.91e-03$, which is an effect of the curvature regularization.

4.4.3 Mechanical Part and Knot

In the following tests we consider two point clouds that have been tested also in reference works as [33, 54].

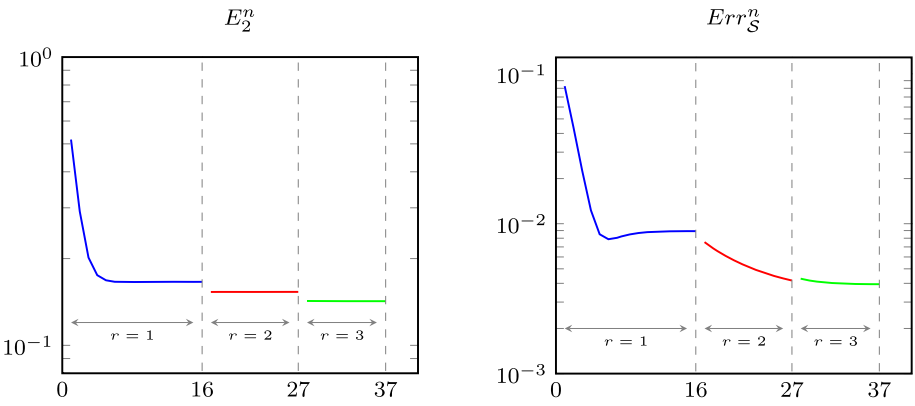

Fig. 12 Energy and error on cloud computed for the "Cube&Spheres" using WENO interpolator

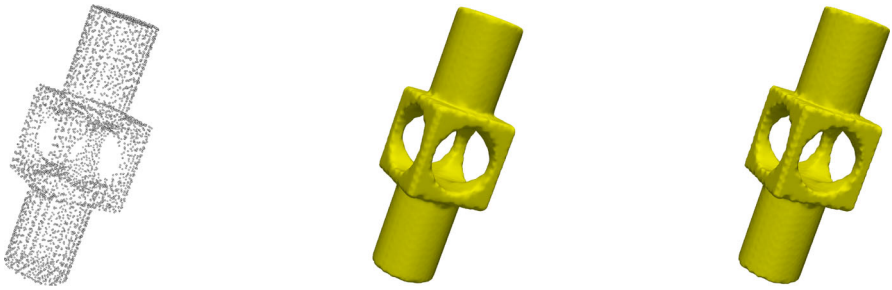

Fig. 13 Mechanical part: the dataset, and the zero level set obtained with Q1 (center) and WENO (right) interpolator

The first one is made up of 4102 points representing a mechanical part which presents both rounded and sharp features. The results are depicted in Fig. 13 where one can notice how the WENO reconstruction fits more faithfully the dataset; in fact, the final errors computed on $\mathcal{S}$ are equal to $7.59e-03$ and $3.74e-03$, respectively for the Q1 interpolator and the WENO one.

The second dataset consists of 10000 points and represents a knot. Motivated by its intriguing topological features, we explored the capability of our method to accurately reconstruct its shape, even when we start from an initial data with a completely different topology. If Fig. 14 we show the initial and final data of the reconstruction procedure using the WENO interpolator setting $C_{\Delta x} = 2$: in the first row the initial data is computed as usual, choosing $K_{\mathcal{S}} = 10$, while in the second row the initial data is chosen to be an ellipsoid encompassing the dataset (the maximum number of iterations in this latter case has been set to 200). Of course, we observe a difference in the number of iterations: the first case requires 61, 23 and 10 iterations, while the second one requires 120, 24 and 14. The computational times are reported in Tab. 9.

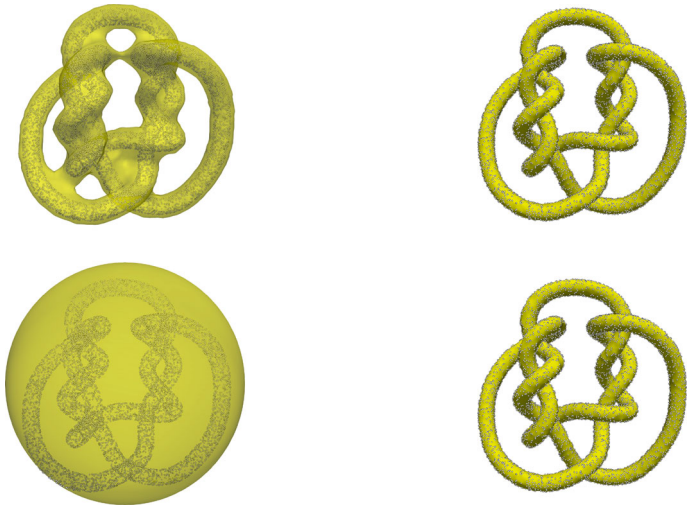

Fig. 14 The reconstruction procedure for the knot point cloud, using WENO interpolator. First row: usual initial guess obtained as a proper isocontour of the distance function from $\mathcal{S}$. Second row: ellipsoidal initial guess

4.5 Data Sets From Laser Scans

Here we test our method using data sets coming from laser-scanning of real objects. These point clouds are made available in the Digital Shape WorkBench of the AIM@SHAPE and VISIONAIR projects [1] or in the 3D Scanning Repository of the Stanford University [36].

4.5.1 The "Frog" and the "Bunny"

We first consider a point cloud named "Frog" [1] and one named "Bunny" [36], made up of 2512 and 35947 points, respectively. The results are shown in Fig. 15 and in Tables 6, 7 and we point out that for the "Bunny" we set $C_{\Delta x} = 2$. In both these two cases the data sets have holes in the bottom and that's why we had to increase the threshold $\gamma_{\mathcal{S}}$ for the initial data computation, thus we set $K_{\mathcal{S}} = 10$. This obviously produced an enlarged initial contour and therefore more iterations were needed during the first run, but this simple strategy secured us from getting distorted results deriving from a bad initial data. Of course, alternatively, one could also fill these holes in an ad-hoc preprocessing stage by adding some points to the cloud.

Note in particular, from Fig. 15, that the high order interpolator provides a more faithful reconstruction, which can be also visually appreciated from some details of the dataset where the cross section of the point cloud it's comparable with its resolution.

4.5.2 A Teapot with Tiny Details

In the next test we focus our attention on the ability of our algorithm to capture tiny details of an object. A point cloud named "Teapot" [1] has been considered for this aim since it presents a more complex topology and parts whose cross section is more or less comparable with the size of the cloud. In Fig. 16, the reconstructions obtained with the WENO interpolator,

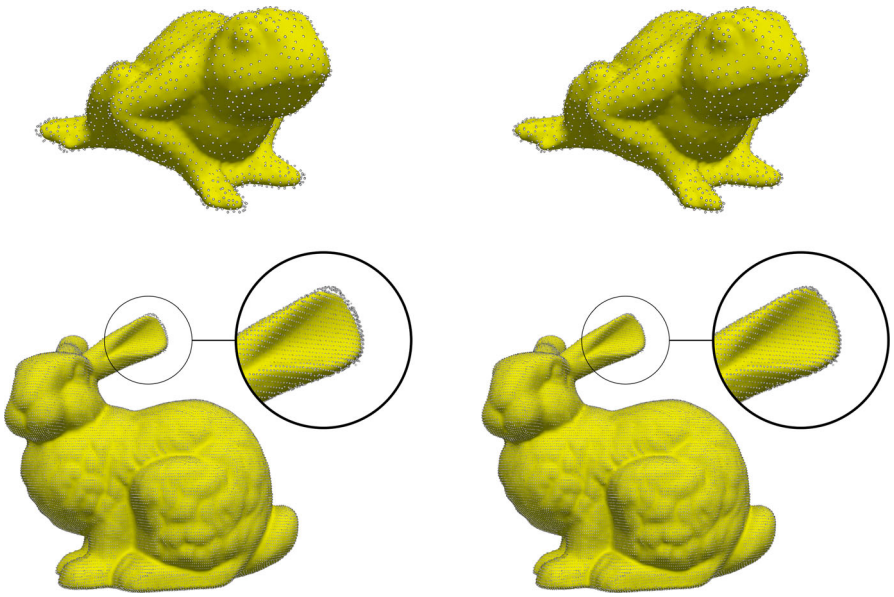

Fig. 15 The reconstructed surfaces of the “Frog" and the “Bunny", respectively on the first and on the second row. On the left, the result obtained with the Q1 interpolant, while on the right, the result obtained using the WENO interpolant. One can appreciate how the WENO recostruction better recovers the details of the paws, in the “Frog" case, and of the ear, in the “Bunny" case

Table 6 Errors and energy computed for the “Frog” at the end of each run

	Q1		WENO	
r	Energy	Error on $\mathcal{S}$	Energy	Error on $\mathcal{S}$
1	$2.70e-01$	$3.97e-03$	$2.64e-01$	$3.54e-03$
2	$2.77e-01$	$3.17e-03$	$2.75e-01$	$2.55e-03$
3	$2.82e-01$	$3.33e-03$	$2.82e-01$	$2.34e-03$

Table 7 Errors and energy computed for the “Bunny” at the end of each run

	Q1		WENO	
r	Energy	Error on $\mathcal{S}$	Energy	Error on $\mathcal{S}$
1	$9.80e-02$	$3.63e-03$	$8.78e-02$	$3.16e-03$
2	$8.00e-02$	$1.38e-03$	$7.62e-02$	$1.18e-03$
3	$7.23e-02$	$1.09e-03$	$7.08e-02$	$8.09e-04$

choosing respectively $C_{\Delta x} = 0.5$ and $C_{\Delta x} = 0.25$, are depicted. We point out that in is this test we force $\mu = 0$ in the first two runs in order to retain as much details as possible. In fact, higher values of μ expose to the risk of loosing important details of the shape. The higher resolution of the grid of course requires a great computational effort: the finer reconstruction took approximately six times as long as the first (see Table 9).

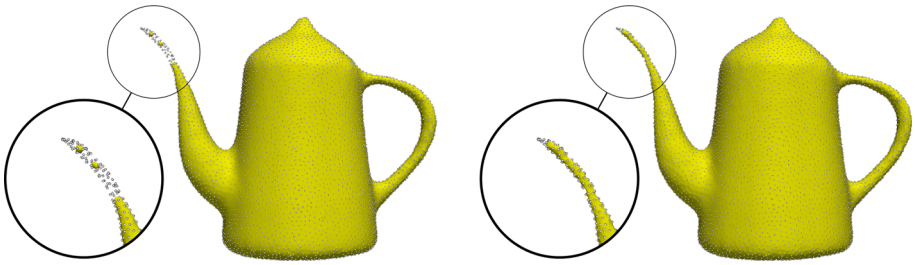

Fig. 16 Final reconstructions of the "Teapot" with WENO interpolator. The left panel collects the result obtained with $C_{\Delta x} = 0.5$, while on the right $C_{\Delta x} = 0.25$. The first setting fails in recovering the spout, while the second one succeeds

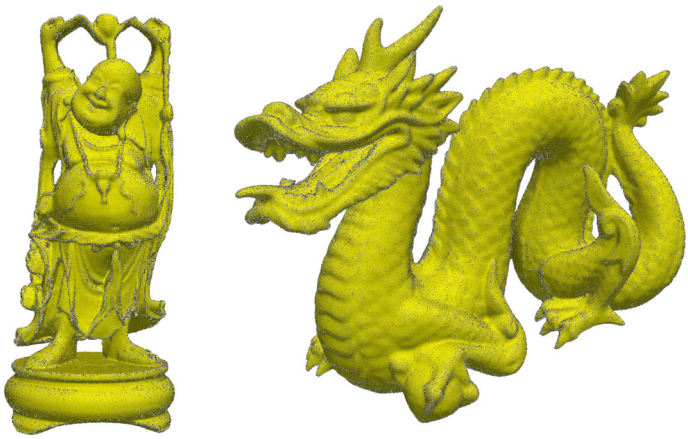

Fig. 17 Final reconstructions of the "Happy Buddha" and the "Dragon" using WENO interpolator

4.5.3 Complex Shapes and Topology: the "Happy Buddha" and the "Dragon"

Finally we have done some tests on very complex point clouds present in the Stanford 3D Scanning Repository [36]. We considered two data sets, named "Happy Buddha" and "Dragon" respectively, because they present some nice features like free holes, small bridges due to the carving and many details to be recovered. The results of our reconstructions are depicted in Fig. 17 and have been obtained setting $K_S = 10$ in the computation of the initial data, and $C_{\Delta x} = 4$ in order to limit the huge amount of grid points.

On the "Happy Buddha" test, we point out that the reconstructed surface recovered equally well both the very flat surfaces of the base and the tiny details on the sides, on the belly and on the drapery. Also all the holes of the highly nontrivial topology were correctly recovered, including the small ones on the sides: we point out that the initial data had detected the two big holes at the top but missed the two tiny ones at the sides which were recovered during the surface evolution, confirming the ability of the method to deal with topological changes of the surface.

On the "Dragon" test, we point out that the scales on the skin are still well approximated despite the curvature regularization. Also, we stress how the level set has been pushed inside the mouth during its evolution and how the sharp teeth shapes have been well approximated.

Table 8 Dimensions of the Cartesian grid involved for each 3d test at each progressive run

Point cloud	Grid		
	$r = 1$	$r = 2$	$r = 3$
Sphere	$54 \times 54 \times 54$	$78 \times 78 \times 78$	$135 \times 135 \times 135$
Cube&Spheres	$55 \times 62 \times 55$	$81 \times 94 \times 81$	$141 \times 166 \times 141$
Frog	$89 \times 75 \times 87$	$116 \times 89 \times 113$	$211 \times 157 \times 204$
Bunny	$109 \times 108 \times 92$	$177 \times 175 \times 142$	$332 \times 329 \times 262$
Knot	$105 \times 106 \times 67$	$169 \times 171 \times 93$	$317 \times 321 \times 165$
Knot (ellipse)	$105 \times 106 \times 67$	$169 \times 171 \times 93$	$317 \times 321 \times 165$
Mechanical part	$77 \times 77 \times 148$	$92 \times 92 \times 234$	$163 \times 163 \times 446$
Teapot	$119 \times 81 \times 102$	$176 \times 101 \times 143$	$330 \times 181 \times 264$
Teapot (finer)	$184 \times 109 \times 151$	$330 \times 181 \times 264$	$638 \times 340 \times 506$
Happy Buddha	$152 \times 331 \times 152$	$272 \times 631 \times 272$	$522 \times 1241 \times 522$
Dragon	$263 \times 193 \times 132$	$494 \times 354 \times 233$	$966 \times 687 \times 444$

Table 9 Computational times (min) of the algorithm

Point cloud	Ranks	Total CPU time	
		Q1	WENO
Sphere	16	0.03	0.05
Cube&Spheres	16	0.03	0.05
Frog	16	0.08	0.11
Bunny	16	0.28	0.34
Knot	16	0.27	0.42
Knot (ellipse)	16	0.32	0.68
Mechanical part	16	0.11	0.36
Teapot	32	0.43	0.58
Teapot (finer)	32	2.15	3.58
Happy Buddha	32	1.35	2.00
Dragon	32	1.48	2.15

4.6 Scalability of the Algorithm

Our algorithm, thanks to the localization of the computational effort on a band around the evolving zero level set, which is a codimension 1 variety of $\mathbb{R}^n$, has a cost that scales as $\mathcal{O}(1/\Delta x^{n-1})$ under grid refinement. This is confirmed by comparing the two "Teapot" experiments (Table 8 and 9). The memory footprint of our implementation scales instead as $\mathcal{O}(1/\Delta x^n)$, due to the data attached to the full grid $\mathcal{G}$. This has suggested to consider a distributed memory parallel implementation based on the MPI paradigm.

In this last subsection we have collected some scalability results to evaluate the efficiency of the algorithm. The test depicted in Fig. 18 has been performed using the "Dragon" point cloud (setting $C_{\Delta x} = 2$) progressively increasing the number of nodes, while keeping constant the number of processors per node.

Computing the parallel efficiency of a run with M cores as $\epsilon = \frac{40 \cdot T_{40}}{M \cdot T_M}$, where T_M is the computational time, we obtain 84%, 63%, 58%, respectively for 80, 120 and 160 cores.

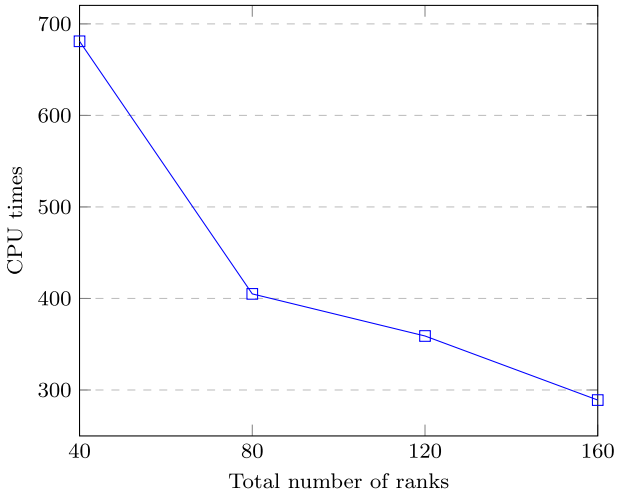

Fig. 18 Scalability results for the "Dragon" reconstruction. The blue marks report the CPU times in seconds required using a fixed number of ranks per node (40) and gradually increasing the number of nodes, from 1 to 4

This can be ascribed to the interplay between the Cartesian grid partitioning for $\mathcal{G}$ and the localization techniques of Sect. 3.8. In fact, when the number of cores is increased, also the chance of having cores with no (or small) intersection with the active computational band is also increased, with a negative impact on load balancing.

5 Conclusions

We presented a complete workflow for the reconstruction of surfaces starting from unorganized point cloud data, without assuming any information about the connection between points. We used the LSM to detect and evolve these surfaces in an implicit way. In particular, we used the signed distance function as level set function to preserve numerical accuracy and to get some nice geometrical features for free. The numerical method is based on a SL scheme with a local Q1 interpolator or a WENO one and thus it works locally and on fixed rectangular grids. The code runs in parallel and it turned out to be quite efficient despite the very fine resolution of the grids employed and the high amount of grid points involved in the computations. We presented several numerical tests in two and three dimensions measuring the error in terms of distance of the reconstructed surface from the cloud and distance from the exact signed distance function. The tests suggest that the workflow illustrated here is quite promising, especially using WENO techniques, which shows better results with respect to the simple Q1 interpolator. Future research will be directed towards improving the efficiency of the algorithm involving quadree and octree adaptive meshes to solve the load balancing issues highlighted in Sect. 4.6, focusing all the computational effort in the regions close to the data. Interesting directions of research would be also the replacement of the current SL scheme with its monotone version (see [11]) to achieve a monotone decrease of the error and of the energy functional.

Acknowledgements The authors would like to thank the PETSc developer team for promptly including in the 3.19 release the relevant features of DMSWARM that made possible the simulations in this paper. The

authors acknowledge the CINECA award under the ISCRA initiative, for the availability of high-performance computing resources and support. Both authors are members of the Gruppo Nazionale Calcolo Scientifico-Istituto Nazionale di Alta Matematica (GNCS-INdAM).

Funding Open access funding provided by Università degli Studi dell'Insubria within the CRUI-CARE Agreement. This research has been partly funded by the PRIN-PNRR project "MATHematical tools for predictive maintenance and PROtection of CULTtural heritage (MATHPROCULT)" (code P20228HZWR).

Data Availability The datasets analysed during the current study are available in the repositories cited in the text.

Declarations

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

References

1. 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
2. Amenta, N., Bern, M., Kamvysselis, M.: A new voronoi-based surface reconstruction algorithm. In: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1998, p. 415 - 422 (1998)
3. 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**(5), 1636–1667 (2004)
4. Balay, S., Abhyankar, S., Adams, M.F., Brown, J., Brune, P., Buschelman, K., Dalcin, L., Eijkhout, V., Gropp, W.D., Karpeyev, D., Kaushik, D., Knepley, M.G., May, D.A., McInnes, L.C., Mills, R.T., Munson, T., Rupp, K., Sanan, P., Smith, B.F., Zampini, S., Zhang, H., Zhang, H.: PETSc/TAO users manual. Tech. Rep. ANL-21/39 - Revision 3.19, Argonne National Laboratory (2023)
5. Balay, S., Gropp, W.D., McInnes, L.C., Smith, B.F.: Efficient management of parallelism in object oriented numerical software libraries. In: E. Arge, A.M. Bruaset, H.P. Langtangen (eds.) Modern Software Tools in Scientific Computing, pp. 163–202 (1997)
6. Berger, M., Tagliasacchi, A., Seversky, L., Alliez, P., Guennebaud, G., Levine, J., Sharf, A., Silva, C.: A survey of surface reconstruction from point clouds. Comput. Graphics Forum (2016). https://doi.org/10.1111/cgf.12802
7. Berger, M., Tagliasacchi, A., Seversky, L., Alliez, P., Levine, J., Sharf, A., Silva, C.: State of the art in surface reconstruction from point clouds. Eurographics 2014-State of the Art Reports (2014)
8. Bonaventura, L., Calzola, E., Carlini, E., Ferretti, R.: Second order fully semi-Lagrangian discretizations of advection-diffusion-reaction systems. J. Sci. Comput. (2021). https://doi.org/10.1007/s10915-021-01518-8
9. Bonaventura, L., Ferretti, R.: Semi-Lagrangian methods for parabolic problems in divergence form. SIAM J. Sci. Comput. **36**(5), A2458–A2477 (2014). https://doi.org/10.1137/140969713
10. Carlini, E., Falcone, M., Ferretti, R.: Convergence of a large time-step scheme for mean curvature motion. Interf. Free Bound. **12**(4), 409–411 (2010). https://doi.org/10.4171/IFB/240
11. Carlini, E., Ferretti, R.: A semi-Lagrangian approximation of min-max type for the stationary mean curvature equation. Numerical Math. Adv. Appl. (2008). https://doi.org/10.1007/978-3-540-69777-0_81

12. Carlini, E., Ferretti, R.: A semi-Lagrangian scheme with radial basis approximation for surface reconstruction. Comput. Vis. Sci. **18**(2–3), 103–112 (2017). https://doi.org/10.1007/s00791-016-0274-2
13. Carlini, E., Ferretti, R., Russo, G.: A weighted essentially nonoscillatory, large time-step scheme for Hamilton-Jacobi equations. SIAM J. Sci. Comput. **27**(3), 1071–1091 (2006). https://doi.org/10.1137/040608787
14. Carr, J., Beatson, R., Cherrie, J., Mitchell, T., Fright, W., Mccallum, B., Evans, T.: Reconstruction and representation of 3d objects with radial basis functions. ACM SIGGRAPH (2001). https://doi.org/10.1145/383259.383266
15. Carr, J., Beatson, R., McCallum, B., Fright, W., McLennan, T., Mitchell, T.: Smooth surface reconstruction from noisy range data. In: Proceedings of the 1st International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia, GRAPHITE, p. 119-126 (2003)
16. Cazals, F., Giesen, J.: Delaunay triangulation based surface reconstruction. Springer, Berlin Heidelberg (2006)
17. Cheng, Z.Q., Wang, Y.Z., Li, B., Xu, K., Dang, G., Jin, S.Y.: A survey of methods for moving least squares surfaces. In: IEEE/EG Symposium on Volume and Point-Based Graphics, pp. 9–23 (2008)
18. Clarelli, F., De Filippo, B., Natalini, R.: Mathematical model of copper corrosion. Appl. Math. Model. **38**(19–20), 4804–4816 (2014). https://doi.org/10.1016/j.apm.2014.03.040
19. Clarelli, F., Fasano, A., Natalini, R.: Mathematics and monument conservation: free boundary models of marble sulfation. SIAM J. Appl. Math. **69**(1), 149–168 (2008). https://doi.org/10.1137/070695125
20. Coco, A., Preda, S., Semplice, M.: From point clouds to 3d simulations of marble sulfation. Mathematical Modeling in Cultural Heritage, pp. 153–174. Springer Nature Singapore (2023)
21. 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). https://doi.org/10.1016/j.jcp.2018.01.016
22. 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). https://doi.org/10.1016/j.amc.2020.125503
23. Courant, R., Isaacson, E., Rees, M.: On the solution of nonlinear hyperbolic differential equations by finite differences. Comm. Pure Appl. Math. **5**(3), 243–255 (1952). https://doi.org/10.1002/cpa.3160050303
24. Daniel, P., Medl'A, M., Mikula, K., Remešíková, M.: Reconstruction of surfaces from point clouds using a lagrangian surface evolution model. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), p. 589–600 (2015)
25. Edelsbrunner, H.: Shape reconstruction with Delaunay complex. Lecture Notes Comput. Sci. **1380**, 119–132 (1998). https://doi.org/10.1007/BFb0054315
26. Falcone, M., Ferretti, R.: Consistency of a large time-step scheme for mean curvature motion. F. Brezzi, A. Buffa, S. Corsaro, A. Murli (eds.) Numerical Mathematics and Advanced Applications, pp. 495–502. Springer Milan (2003)
27. Falcone., M., Ferretti, R.: Semi-Lagrangian approximation schemes for linear and Hamilton-Jacobi equations. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA (2014)
28. 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**(1), 205–227 (2002). https://doi.org/10.1006/jcph.2001.6977
29. Gregorski, B., Hamann, B., Joy, K.: Reconstruction of b-spline surfaces from scattered data points. Proceed. Comput. Graphics Int. **2000**, 163–170 (2000). https://doi.org/10.1109/CGI.2000.852331
30. Haličková, J., Mikula, K.: Level set method for surface reconstruction and its application in surveying. J. Surveying Engrg. **142**(3), 04016007 (2016). https://doi.org/10.1061/(ASCE)SU.1943-5428.0000159
31. Hartmann, D., Meinke, M., Schröder, W.: Differential equation based constrained reinitialization for level set methods. J. Comput. Phys. **227**(14), 6821–6845 (2008). https://doi.org/10.1016/j.jcp.2008.03.040
32. He, Y., Huska, M., Kang, S.H., Liu, H.: Fast algorithms for surface reconstruction from point cloud. Springer Proceed. Math. Statistics **360**, 61–80 (2021). https://doi.org/10.1007/978-981-16-2701-9_4
33. Hoppe, H., Derose, T., Duchamp, T., Mcdonald, J., Stuet-zle, W.: Surface reconstruction from unorganized point clouds. Comput. Graphics (1992). https://doi.org/10.1145/142920.134011
34. Jiang, G.S., Shu, C.W.: Efficient implementation of weighted ENO schemes. J. Comput. Phys. **126**, 202–228 (1996). https://doi.org/10.1006/jcph.1996.0130
35. Kósa, B., Haličková-Brehovská, J., Mikula, K.: New efficient numerical method for 3d point cloud surface reconstruction by using level set methods. In: Proceedings of Equadiff 2017 Conference, p. 387–396 (2017)
36. Laboratory, S.U.C.G.: The stanford 3d scanning repository. http://graphics.stanford.edu/data/3Dscanrep/. Accessed March 2022

37. Liu, X.: Research on 3D object reconstruction method based on deep learning. Highlights Sci. Eng. Technol. **39**, 1221–1227 (2023). https://doi.org/10.54097/hset.v39i.6732
38. Marcon, M., Piccarreta, L., Sarti, A., Tubaro, S.: Fast point-cloud wrapping through level-set evolution. In: 1st European Conference on Visual Media Production (CVMP) 2004, pp. 119–125 (2004)
39. Osher, S., Fedkiw, R.: Level set methods and dynamic implicit surfaces, Applied Mathematical Sciences, p. XIII. Springer New York, NY (2003)
40. Osher, S., Sethian, J.A.: Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations. J. Computat. Phys. **79**(1), 12–49 (1988). https://doi.org/10.1016/0021-9991(88)90002-2
41. Park, I.K., Yun, I.D., Lee, S.U.: Constructing nurbs surface model from scattered and unorganized range data. In: Second International Conference on 3-D Digital Imaging and Modeling (Cat. No.PR00062), p. 312–320 (1999)
42. Peng, D., Merriman, B., Osher, S., Zhao, H.K., Kang, M.: A pde-based fast local level set method. J. Computat. Phys. **155**(2), 410–438 (1999). https://doi.org/10.1006/jcph.1999.6345
43. Remondino, F.: Heritage recording and 3d modeling with photogrammetry and 3d scanning. Remote Sensing **3**, 1104–1138 (2011). https://doi.org/10.3390/rs3061104
44. Remondino, F., El-Hakim, S.: Image-based 3d modelling: a review. Photogram. Rec. **21**, 269–291 (2006). https://doi.org/10.1111/j.1477-9730.2006.00383.x
45. 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)
46. Sharma, R., Schwandt, T., Kunert, C., Urban, S., Broll, W.: Point cloud upsampling and normal estimation using deep learning for robust surface reconstruction. In: Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, p. 70–79 (2021)
47. Strain, J.: Semi-Lagrangian methods for level set equations. J. Comput. Phys. **151**(2), 498–533 (1999). https://doi.org/10.1006/jcph.1999.6194
48. Sulzer, R., Marlet, R., Vallet, B., Landrieu, L.: A survey and benchmark of automatic surface reconstruction from point clouds (2024) arXiv:2301.13656
49. Sussman, M., Smereka, P., Osher, S.: A level set approach for computing solutions to incompressible two-phase flow. J. Computat. Phys. **114**(1), 146–159 (1994). https://doi.org/10.1006/jcph.1994.1155
50. 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). https://doi.org/10.1007/s11831-019-09320-4
51. Zeng, Y., Zhu, Y.: Implicit surface reconstruction based on a new interpolation / approximation radial basis function. Comput. Aided Geom. Design **92**, 102062 (2021). https://doi.org/10.1016/j.cagd.2021.102062
52. Zhao, H.K.: A fast sweeping method for eikonal equations. Math. Comput. **74**, 603–627 (2005). https://doi.org/10.1090/S0025-5718-04-01678-3
53. Zhao, H.K., Osher, S., Fedkiw, R.: Fast surface reconstruction using the level set method. In: Proceedings of IEEE Workshop on Variational and Level Set Methods in Computer Vision, p. 194–201 (2001)
54. Zhao, H.K., Osher, S., Merriman, B., Kang, M.: Implicit and nonparametric shape reconstruction from unorganized data using a variational level set method. Comput. Vis. Image Underst. **80**(3), 295–314 (2000). https://doi.org/10.1006/cviu.2000.0875