Implicit shape reconstruction of unorganized points using PDE-based deformable 3D manifolds

E.Franchini* S. Morigi† F. Sgallari‡

Abstract

In this work we consider the problem of shape reconstruction from an unorganized data set which has many important applications in medical imaging, scientific computing, reverse engineering and geometric modelling. The reconstructed surface is obtained by continuously deforming an initial surface following the Partial Differential Equation (PDE)-based diffusion model derived by a minimal volume-like variational formulation. The evolution is driven both by the distance from the data set and by the curvature analytically computed by it. The distance function is computed by implicit local interpolants defined in terms of radial basis functions. Space discretization of the PDE model is obtained by finite co-volume schemes and semi-implicit approach is used in time/scale. The use of a level set method for the numerical computation of the surface reconstruction allows us to handle complex geometry and even changing topology, without the need of user-interaction. Numerical examples demonstrate the ability of the proposed method to produce high quality reconstructions. Moreover, we show the effectiveness of the new approach to solve hole filling problems and Boolean operations between different data sets.

Keywords: shape reconstruction, RBF interpolation, PDE diffusion model, segmentation

1 Introduction

The shape reconstruction problem has several applications in areas that include computer visualization, data analysis, biomedical imaging, virtual simulation, computer graphics, etc. Surface reconstruction consists of finding out the original surface shape from partial information that can include points, pieces of curves, or surfaces. In this paper we consider implicit surface reconstruction from unorganized, eventually incomplete data sets. The reconstruction of a 3D model is usually obtained from scattered data points sampled from the surface of the physical 3D object. The acquisition of the data is in general realized through 3D scanning systems which are able to capture a dense sampling usually organized into range images, i.e. sets of distances to the data. The problem is ill-posed, i.e. there is no unique solution. Furthermore the problem can become quite challenging if the topology of the real surface is complicated or even unknown. A good reconstruction algorithm should be able to deal with non-uniform and incomplete data set in order to construct an arbitrary topology surface with a controlled hole filling strategy, smoothness and a right behavior for deformation, animation and other dynamical operation.

In general, the techniques for reconstructing a surface from unorganized data sets, can be divided into two main classes according to the way used to represent the surface: implicit (non-parametric approach) or explicit (parametric approach). Explicit surfaces in $\mathbb{R}^3$ are defined as the set of points $\{x(u,v), y(u,v), z(u,v)\}$ where the parameters $u, v \in \mathbb{R}$.

*Department of Mathematics-CIRAM, University of Bologna, Via Saragozza 8, 40123 Bologna, Italy. E-mail: franchini@dm.unibo.it

†Department of Mathematics-CIRAM, University of Bologna, Piazza Porta S. Donato 5, 40126 Bologna, Italy. E-mail: morigi@dm.unibo.it

‡Department of Mathematics-CIRAM, University of Bologna, Via Saragozza 8, 40123 Bologna, Italy. E-mail: sgallari@dm.unibo.it

Very popular parametric approaches are based on spline patches which give rise to smooth reconstructed surface. However these approaches require a nice parametrization and arbitrary topologies have to be handled by patching of different surface pieces, which can be very difficult for surface reconstruction from arbitrary data in three or higher dimensions [14],[26],[3].

Several algorithms of shape reconstruction follow a computational geometry approach and are mainly based on Delaunay triangulations and Voronoi diagrams. These methods construct a collection of simplexes that form a shape from a set of unorganized points. These approaches are versatile, but it's not trivial to handle non-uniform and noisy data [1].

Recently, implicit methods have captured a lot of attention. In the implicit representation, the geometry and topology of the surface is defined as a particular iso-contour of a scalar implicit function over $\mathbb{R}^3$. The implicit approaches reconstruct an implicit function such that a certain level set of this function fits the data best. The iso-contour represents the reconstructed surface. The advantages of these techniques are the topological flexibility, the possibility to easily capture geometry property of the surface, and to realize in a simple way the classical Boolean operations, while it is a challenge to deal with open or incomplete surfaces, [21].

Approaches based on moving least-squares technique have been used widely for the reconstruction of point-set surfaces, see [2], [16], [34].

A particularly powerful approach for implicit surface reconstruction based on Radial Basis Functions (RBF) has been relatively recently introduced to deal with large scattered data sets without relying on a priori knowledge of the object topology [8],[4],[5],[30]. The implicit signed distance function is constructed by approximation or interpolation of surface and exterior constraint points defined by the data set. The coefficients of the reconstructed function are determined by solving linear systems of equations. The computational cost is very high for large data sets, when the construction is global. These approaches are not particularly robust in the presence of holes and low sampling density, ad hoc techniques and a priori knowledge on the surface topology should be inferred to obtain a faithful reconstruction [7].

The shape reconstruction problem is a well-known fundamental problem in computer vision and image processing fields where it is better known as *segmentation* process and it represents a base component in automated vision systems and medical applications [12]. Medical imaging modalities such as magnetic resonance imaging, computed tomography, ultrasound scans produce large volumes of scalar or vector measurements that are in general corrupted by noise and often, large parts of the structure boundary are missing. From these discrete data, given on two-dimensional or three-dimensional uniform grids, some information, such as, for example, the morphology of specific body structures, has to be extracted.

Several variational formulations and PDEs methods have been proposed to extract objects of interest from 3D images such as the well known geodesic active contour model defined by Caselles et al. [9] and widely used in different image processing applications. In a more recent segmentation approach introduced in [29], the authors present a PDE geometric model for boundary completion, providing interesting results in recovering structures from 2D and 3D medical images. In [32] a weighted minimal surface model based on variational formulations is proposed. The unsigned distance function to the data set is computed by solving on a fixed grid the Eikonal equation, and the reconstructed surface is then obtained by solving with a time-explicit scheme a non-linear parabolic equation or, alternatively, a convection model.

In this work we are interested in extending some previous work on PDE-based models aimed at segmenting 3D objects from 3D uniform grids [10], to the challenging case of large unorganized set of points. We propose a two steps strategy where the first step computes the distance function to the data set, while the second step reconstructs the implicit continuous surface to obtain a watertight 3D model. We combine the accuracy of an RBF local reconstruction for the first step, with the efficiency in the second step of a semi-implicit scheme for the surface deformation based on a geometric model for boundary completion [10]. Moreover, we devise a strategy to accelerate the linear system solver for the obtained discretization problem.

In the present context, we consider the acquisition of the data realized typically by 3D scanning systems which are able to capture a huge amount of data, sometimes noisy data, and eventually with some missing parts. Moreover, we assume no a priori knowledge about the topology of the

surface to be reconstructed. As input our method requires a collection of constraint points that specify where the surface to be reconstructed is located. Most of the constraint points come directly from the input data and, in addition, some points have to be explicitly identified as being outside the surface. The use of a priori knowledge about the scanning acquisition system lets us define automatically the constraint points that lie outside the object.

The main goal is to reconstruct the shape of arbitrary topology objects with a controlled hole filling strategy using implicit surface representation.

At this aim, a preliminary computation is required to locally approximate the signed distance function to the data set by using local implicit interpolation. Using the signed distance function to compute the shape indicator, then the surface is recovered by continuously deforming the level sets of an initial 3D manifold towards the object boundaries. The evolution is driven both by the distance to the data set and by the curvature analytically computed by it.

The deformation model is based on a diffusion-advection PDE model which governs the 3D manifold evolutions. To easily deal with arbitrary topology data set, we use the level set methodology for the deformation of the 3D manifold.

The level set approach has been introduced by Osher and Sethian [22],[25], to solve problems of surface evolution and it is a powerful numerical technique that allows for cusps, corners, and automatic topological changes and provides easy discretizations on the Cartesian grid.

This paper is organized as follows. Section 2 introduces the 3D manifold evolution and related PDE-model which has been used to evolve the level sets surfaces towards the object boundaries. Section 3 discusses some numerical aspects involved in the reconstruction algorithm, such as the approximation of the distance function by local implicit interpolants based on radial basis functions, and the discretization of the proposed PDE model for the solution of the 3D manifold evolution. Numerical examples in Section 4 illustrate the performance of the reconstruction method when applied to real and synthetic data sets and Section 5 contains concluding remarks.

2 A weighted 3D manifold evolution

A parametric 3D manifold in 4D space is the natural extension to a parametric surface in 3D space defined as the set of point $\{x(u,v), y(u,v), z(u,v)\}$, where (u,v) belongs to a parametric domain in $\mathbb{R}^2$.

The graph of a trivariate function ϕ mapping an open set $\Omega \subset \mathbb{R}^3$ into $\mathbb{R}$, can be considered as a special 3D manifold using the parameterization

$$\mathbf{X}(\mathbf{u}) = \{u, v, w, \phi(u,v,w)\}, \quad (u,v,w) \in \Omega \subset \mathbb{R}^3. \tag{1}$$

The normal $n(\mathbf{u})$ of the 3D manifold at $\mathbf{X}(\mathbf{u})$ is given by

$$n(\mathbf{u}) = \frac{(-\phi_u, -\phi_v, -\phi_w, 1)}{\sqrt{1+\phi_u^2+\phi_v^2+\phi_w^2}}. \tag{2}$$

Considering a fixed parametric domain, the evolution of the 3D manifold in Ω corresponds to local variations in the ϕ function. Along a given $\mathbf{v}$ vector field defined on Ω, the mean curvature flow for the points $\mathbf{X}(\mathbf{u}) = \{u, v, w, \phi\}$, is the following:

$$\frac{\partial \mathbf{X}}{\partial t} = \frac{H}{n \cdot \mathbf{v}} \cdot \mathbf{v}, \tag{3}$$

with H the mean curvature and n the unit normal vector given by (2) [11]. Let us consider the vector field $\mathbf{v} = (0,0,0,1)$, then

$$\frac{1}{n \cdot \mathbf{v}} \cdot \mathbf{v} = \sqrt{1 + \|\nabla \phi\|^2}(0,0,0,1), \tag{4}$$

and we can rewrite (3) as

$$\frac{\partial \mathbf{X}}{\partial t} = (0,0,0,\frac{\partial \phi}{\partial t}). \tag{5}$$

Thus for tracking the evolving 3D manifold is sufficient to follow the evolution of ϕ via

$$\frac{\partial \phi}{\partial t} = H\sqrt{1 + \|\nabla \phi\|^2}. \tag{6}$$

Replacing in (6) the mean curvature H defined as function of ϕ as

$$H = div(\frac{\nabla \phi}{\sqrt{1 + \|\nabla \phi\|^2}}), \tag{7}$$

the mean curvature flow for ϕ is then given by

$$\frac{\partial \phi}{\partial t} = \sqrt{1 + \|\nabla \phi\|^2} div(\frac{\nabla \phi}{\sqrt{1 + \|\nabla \phi\|^2}}), \tag{8}$$

with initial condition, considered, for example, as the function $\phi^0 = 1/D$, where D is the distance function from a fixed point inside the region of interest for reconstruction.

It is interesting to notice that equation (8) can be also derived as the steepest descent of the volume of the 3D manifold

$$V := \int_\Omega dV, \qquad dV = \sqrt{1 + \|\nabla \phi\|^2} dxdydz. \tag{9}$$

Let us consider a new metric g applied to the space. For example, if $\mathcal{P}$ denotes the data set which in general can include points, curves or pieces of surfaces in $I\!R^3$, we define the new metric g as follows

$$g(\mathbf{x}) = d(\mathbf{x}) e^{\|k(\mathbf{x})\|}, \tag{10}$$

where $\mathbf{x} = (x_1, x_2, x_3) \in I\!R^3$, $d(\mathbf{x})$ =dist$(\mathbf{x}, \mathcal{P})$ is the distance function to $\mathcal{P}$ and $k(\mathbf{x})$ is the normalized mean curvature of $d(\mathbf{x})$ evaluated at $\mathbf{x}$. The metric g acts as a shape indicator on the data set which approaches to zero when $d(\mathbf{x})$ approaches to zero, that is near the object boundaries, it is zero for $d(\mathbf{x}) = 0$, on the object boundaries, and it is increased near boundaries characterized by high mean curvatures, thus avoiding early stops.

Then, according to the metric g defined by (10) in Ω the weighted volume on Ω is given by

$$V_g := \int_\Omega g(\mathbf{x}) dV \qquad dV = \sqrt{1 + \|\nabla \phi\|^2} dxdydz. \tag{11}$$

The local minimizer for the volume functional (11) which is attracted to the data set, with a more important attraction where the curvature is larger, is determined by the steepest descent of (11), namely

$$\frac{\partial \phi}{\partial t} = \sqrt{1 + \|\nabla \phi\|^2} \nabla. \left(g(\mathbf{x}) \frac{\nabla \phi}{\sqrt{1 + \|\nabla \phi\|^2}} \right), \tag{12}$$

or, equivalently, in advection-diffusion form

$$\frac{\partial \phi}{\partial t} = g(\mathbf{x}) \nabla. \left(\frac{\nabla \phi}{\sqrt{1 + \|\nabla \phi\|^2}} \right) + \nabla g \cdot \nabla \phi. \tag{13}$$

The PDE model (13) represents the mean curvature motion of the 3D manifold in 4D space with metric g. The metric g in (13) is the shape indicator function appropriately chosen so that the object boundaries act as attractors under a particular flow. This term allows us to extract sharp features, such as edges, corners, spikes, and to accelerate the deformation of the initial function. In section 4 example 4.2 we demonstrate the effectiveness of the g metric. In the evolution of ϕ according (13) the 3D manifold assumes constant values for most regions far from the boundaries. The first term in (13) corresponds to a minimal volume regularization weighted by the function g, while the second term corresponds to the attraction to the data, given by the distance and

curvature field. The advection term in equation (13) introduces a driving force which moves the level surfaces towards the object boundaries.

More details about the use of the Riemann mean curvature flow of graphs and about the role of this model in image segmentation and completing missing boundary can be found in Sarti et al. [29].

In [10] the authors proposed a variant of the segmentation PDE equation for dealing with the boundary completion problem introducing a parameter ε. The hole filling strategy we propose for the object boundary reconstruction problem is based on this idea, thus the PDE model (13) takes the form:

$$\frac{\partial \phi}{\partial t} = \sqrt{\varepsilon^2 + \|\nabla\phi\|^2}\nabla. \left(g(\mathbf{x}) \frac{\nabla\phi}{\sqrt{\varepsilon^2 + \|\nabla\phi\|^2}} \right), \tag{14}$$

where the variability in the parameter ε provides both a regularization effect and a hole filling strategy. The role of ε parameter as a regularization has been proposed by Evans and Spruck in [15] as a tool to prove existence of a viscosity solution of equations of mean curvature flow type. In [10] the ε parameter is interpreted as a modelling parameter which helps in speed up the reconstruction process in case of noisy data and also helps to complete missing boundaries. If ε is close to 1 then the behavior of the flow is mostly diffusive. When instead we decrease ε, i.e. we stay closer to the level set flow (15), we close larger gaps, see examples in Section 4.

The PDE model (14) reduces to (12) for $\varepsilon = 1$, while for $\varepsilon = 0$ and $g(\mathbf{x}) = d(\mathbf{x})$ we obtain the PDE model proposed by Zhao et al. in [31] for surface reconstruction using a level set approach, namely

$$\begin{aligned} \frac{\partial \phi}{\partial t} &= \|\nabla\phi\|\nabla \cdot \left[g \frac{\nabla\phi}{\|\nabla\phi\|} \right] \\ &= \|\nabla\phi\| \left[\nabla g \frac{\nabla\phi}{\|\nabla\phi\|} + g\nabla \cdot \frac{\nabla\phi}{\|\nabla\phi\|} \right] \end{aligned} \tag{15}$$

where $\phi(\mathbf{x}, t)$, represents the surface Γ as the set where $\phi(\mathbf{x}, t) = 0$. In particular, the level set function $\phi(\mathbf{x}, t)$ is negative inside Γ and positive outside Γ. Thus the surface is captured for each time step, by merely locating the set $\Gamma(t)$ for which ϕ vanishes. Instead of tracking the zero level set as in (15), in the proposed equation (14) the evolution moves all the iso-surfaces towards the object boundaries thus topological changes such as breaking and merging can be easily handled. The PDE model (15) has been obtained by minimizing the surface energy functional

$$E(\Gamma) = \int_{\Gamma} d(\mathbf{x}) ds, \tag{16}$$

where ds is the surface element. As derived in [32] the gradient flow of the energy functional (16) is

$$\frac{d\Gamma}{dt} = -\left[\nabla g(\mathbf{x}) \cdot n + g(\mathbf{x}) H\right] n, \tag{17}$$

and the minimizer solution of the gradient flow satisfies the Euler-Lagrange equation

$$\left[\nabla g(\mathbf{x}) \cdot n + g(\mathbf{x}) H\right] = 0 \tag{18}$$

where $n = \frac{\nabla\phi}{|\nabla\phi|}$ is the unit outward normal and $H = \nabla \cdot \frac{\nabla\phi}{|\nabla\phi|}$ is the mean curvature of Γ.

The time evolution PDE for the level set function is defined such that the zero level set has the same motion law as the moving surface, i.e. $\Gamma(t) = \{\mathbf{x} : \phi(\mathbf{x}, t) = 0\}$:

$$\frac{d\phi(\Gamma(t), t)}{dt} = \phi_t + \frac{d\Gamma(t)}{dt} \cdot \nabla\phi = 0$$

Replacing $\frac{d\Gamma(t)}{dt}$ with the gradient flow (17) we get the PDE model (15).

3 Numerical aspects of the reconstruction algorithm

Given an unorganized set $\mathcal{P}$ of N points $P_i \in I\!R^3, i = 1, \cdots, N$, we want to reconstruct the surface Γ defined as the level set of an implicit function $\phi(t, \mathbf{x}), t \in [0, T], \mathbf{x} \in \Omega \subset I\!R^3$ obtained by evolving the model (14). At this aim, we need two numerical ingredients: first, we have to compute the distance function to the arbitrary data set $\mathcal{P}$, then the second step involves the solution of the time dependent PDE (14) on a uniform grid defined on Ω. The numerical technique involved to solve the latter is based on an efficient semi-implicit scheme and finite volume space discretization [10].

The shape reconstruction method is described by Algorithm 3.1 and discussed in more details in the rest of this section.

Algorithm 3.1 Recostruction Algorithm

INPUT: data set $\mathcal{P}$, TOL

OUTPUT: reconstructed surface Γ

STEP **pre-comp**:

determine the spherical cover domain Ω_1

compute $d(\boldsymbol{x})$ *and* $k(\boldsymbol{x})$ *on* Ω_1 *by local RBF reconstruction;*

$\phi(\mathbf{x}, 0) = \phi^0$;

STEP **evolve**:

repeat

compute $\phi(\mathbf{x}, t_n)$ *in (14) by solving (36) for* Φ, $\forall \mathbf{x} \in \Omega_1$*;*

until $\|\phi(\mathbf{x}, t_n) - \phi(\mathbf{x}, t_{n-1})\|_2 < TOL$*;*

STEP **post-comp** *visualize the s-level set of* ϕ*, where* $s = (max(\Phi) + min(\Phi))/2$.

In a pre-processing step, **pre-comp**, we determine the spherical cover of the data set and then we compute the distance function to the data set as a local interpolant function only in points belonging to the spherical cover. Precisely, for a given h-dense data set $\mathcal{P}$, r cannot be a point of Γ if $dist(r, \mathcal{P}) > h$. Thus for each $P_i \in \mathcal{P}$ we define $B_i = \{\mathbf{x} \in I\!R^3 : |\mathbf{x} - P_i| \leq h\}$, the sphere of radius h and center P_i. Hence the working domain $\Omega_1 \subset \Omega, \Omega_1 = \bigcup_{i=1}^{N} B_i$ completely encloses the data set.

The restriction of the domain from Ω to Ω_1, let us speed up the computation of the solution of the PDE model (14) in step **evolve**. In fact, instead of solving it on a uniform volume grid on Ω, which clearly leads to inefficient computation and undue storage, we computes the PDE model solution only on points sufficiently close to the surface to be reconstructed, that is on Ω_1.

Although the idea of computational adaptivity has been used also in the fast sweeping algorithm by Zhao et al. [32], our approach is different. They considered the solution of the Eikonal equation by an upwind differencing scheme for front evolution where the solution is computed on a sweeping narrow band with readjusting neighbors. We compute the surface evolution at each time step, only on all grid points contained on the pre-computed spherical cover by a finite volume discretization of the PDE equation (14). Our proposal has been made possible thanks to the use of Krylov-space iterative linear solvers, as detailed in Section 3.3.

In the final STEP **post-comp**, the reconstructed surface is obtained from the implicit surface ϕ as the zero level set of the function $\phi(\mathbf{x}) - s$, that is the s-level set of ϕ: $\{(u, v, w) \in \Omega : \phi(\mathbf{x}) = s\}$, where s is the average between the maximum and the minimum of the ϕ function. This is motivated by the fact that the flow driven by (14) forms a sharp step in the proximity of the object boundaries, while it approaches at constant values inside/outside the object.

3.1 Computing the distance function

In [31] and [32] the construction of the distance function $d(\mathbf{x})$ from unorganized points on a rectangular grid, follows a PDE-based approach by solving the Eikonal equation:

$$\|\nabla d(\mathbf{x})\| = 1, \qquad d(\mathbf{x}) = 0, \mathbf{x} \in \Gamma, \tag{19}$$

which is the stationary case of the Hamilton-Jacobi equation. We refer the reader to [20] for more detailed information. In particular, given N data points, first each data point has to be located within a grid cell and the exact distance values at the vertices of the grid cell have to be determined. These distance values initialize the grid on which the Eikonal equation (19) has to be solved. One way to solve (19) is to use upwind finite difference schemes and iterate the solution in time, (see [31]). The Fast Marching Method proposed by Sethian [25] solves (19) leading to an algorithm with time complexity $O(MlogM)$, for M grid points. This approach provides in general good results when the given data is represented by continuous curves, however, it fails to compute a good approximation of the distance function to the real shape Γ when isolated points are given, moreover, the results strongly depend on the chosen grid resolution which consequently limits the algorithm efficiency. Another problem involved in the Eikonal equation's solution is the discontinuity at points equidistant from two separate data points.

In this paper we propose an alternative method to evaluate the distance function without solving the Eikonal equation. We approximate the signed distance function by using local implicit interpolants of an arbitrary scattered data set. The implicit representation is based on Radial Basis Functions (RBFs). The signed distance function is analytically constructed in a small neighborhoods around the data points and then evaluated on a set of grid points $\mathbf{x} \in \Omega_1$ in order to construct the shape indicator function $g(\mathbf{x})$ in (14). For a discussion on the choice of the grid size used in solving the PDE model (14), we refer the reader to Section 3.2.2.

At each point P_i in the data set is associated a sphere B_i of the spherical cover; we reconstruct the local distance function on B_i by computing a local RBF interpolant of data contained in B_i.

Let us focus our attention on the construction of a single interpolant on a generic spherical domain B. The constraints may be points on the surface to be reconstructed, that is $\mathbf{x}_j \in \mathcal{P}$, or off-surface points external to the object in order to avoid the trivial solution of a vanishing interpolant everywhere. At each point $\{\mathbf{x}_j \in B; j = 1, \ldots, N_B\}$, a function value f_j is associated according to its *signed-distance* to the set $\mathcal{P}$:

$$\begin{aligned} f_j &= 0 \quad \text{if } \operatorname{dist}(\mathbf{x}_j, \mathcal{P}) = 0 \\ f_j &\neq 0 \quad \text{if} \quad \mathbf{x}_j \quad \text{is an off-surface point,} \end{aligned} \tag{20}$$

where $f_j > 0$ for points outside the object surface and $f_j < 0$ for points inside.

When the data set is provided e.g. by a 3D scanning system, points on the object are acquired directly by the digitalization system, while constraints outside of the object can be captured if the line-of-sight to the scanner is used during the acquisition phase. In practice, the exterior constraints are located at the same distance away from the surface constraints towards the scanner viewpoint and assign them a function value of 1.0. Since they do not represent actual data a sparse sampling of exterior constraints is sufficient to correctly solve the interpolation problem.

The *local implicit reconstruction* problem consists of determining a function $d(\mathbf{x})$ which satisfies the interpolation conditions:

$$d(\mathbf{x}_j) = f_j, \quad j = 1, \ldots, N_B, \tag{21}$$

see [8] for a detailed discussion on RBF local approximations and related computational and stability aspects.

The local reconstruction of the distance function $d(\mathbf{x})$ on B, gives us an implicit, analytical representation which can be evaluated anywhere, in particular, we will evaluate $d(\mathbf{x})$ at each point $\mathbf{x} \in B$ belonging to the 3D grid used for the space discretization of the PDE evolution model (14).

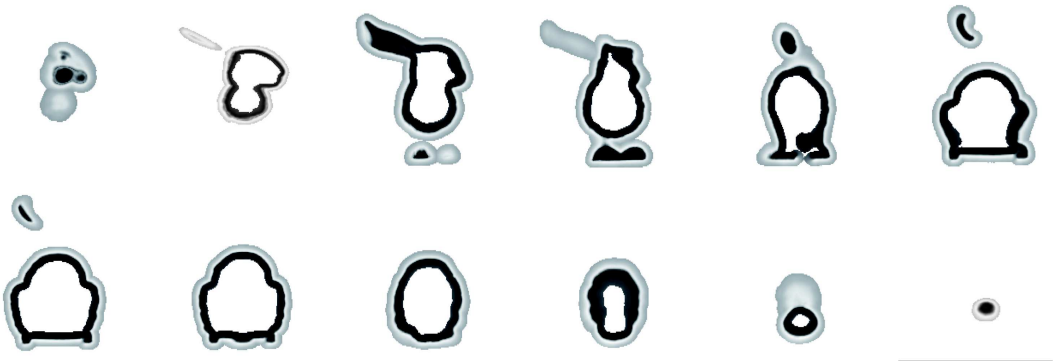


Figure 1: Slices from the signed distance function reconstruction of the `bunny` data set (see Fig.2).

For the local implicit distance function we consider a representation based on Radial Basis Functions since the RBFs represent a well established tool for multivariate scattered data interpolation

The RBF interpolant of the data

$$\{(\mathbf{x}_j, f_j)\}_{j=1}^{N_B}, \qquad \mathbf{x}_j \in B, f_j \in \mathbb{R} \tag{22}$$

with positive definite radial function $\varphi : [0, +\infty) \rightarrow I\!R$, is given by

$$d(\mathbf{x}) = \sum_{k=1}^{N_B} a_k \varphi(\|\mathbf{x} - \mathbf{x}_k\|_2). \tag{23}$$

The coefficients a_k, $k = 1, \ldots, N_B$ are obtained imposing the interpolation conditions (21), i.e., by solving the linear system

$$d(\mathbf{x}_j) = \sum_{k=1}^{N_B} a_k \varphi(\|\mathbf{x}_j - \mathbf{x}_k\|_2) \quad j = 1, \ldots, N_B. \tag{24}$$

It is well known that the linear system (24) is uniquely solvable, as long as the radial basis functions φ are positive definite, such as for example is the case for some popular choices proposed in the literature such as inverse multiquadrics, Gaussians, and compactly supported radial basis functions [33].

Given the analytical reconstruction of $d(\mathbf{x})$ the mean curvature can be reliably calculated as follows:

$$k = \nabla \cdot \left(\frac{\nabla d}{|\nabla d|} \right).$$

Using the notations $d_x = \frac{\partial d}{\partial x}, d_y = \frac{\partial d}{\partial y}, d_z = \frac{\partial d}{\partial z}, d_{xy} = \frac{\partial^2 d}{\partial x \partial y}, d_{xz} = \frac{\partial^2 d}{\partial x \partial z}, d_{yz} = \frac{\partial^2 d}{\partial y \partial z}$, we can compute analytically the curvature term k as

$$k = (d_x^2 d_{yy} - 2 d_x d_y d_{xy} + d_y^2 d_{xx} + d_x^2 d_{zz} - 2 d_x d_z d_{xz} + d_z^2 d_{xx} + d_y^2 d_{zz} - 2 d_y d_z d_{yz} + d_z^2 d_{yy}) / \|\nabla d\|^3.$$

Associated with the spherical covering, a family of non-negative weight functions $\{w_j\}_{j=1,N}$, with limited support $supp(w_j) \subseteq B_j$, is constructed, with the additional property that $\sum_j w_j = 1$ in the entire domain Ω_1. For each grid point $\mathbf{x} \in \Omega_1$ which belongs to support B_j, the evaluation of distance function $d(\mathbf{x})$ is given by the sum of all the contributions of the distance function values obtained for each $P_j, j = 1, \ldots, N$, such that $\mathbf{x} \in B_j$, suitably blended by the weights $\{w_j\}_{j=1,N}$.

In Fig. 1 we show some slices from the signed distance function reconstruction of the `bunny` data set (for the final reconstruction see Fig.2). The values for $d(\mathbf{x})$ are mapped in the range $[0, 255]$

to be represented as grey-level images. The influence of the spherical cover is well represented by non-white pixels in the images.

If the scattered data set presents some holes smaller than a diameter h, then the distance function $d(\mathbf{x})$ on Ω_1 computed on the spherical cover for a h-dense data set, can fill these holes. However, in general, the STEP `pre-comp` in Algorithm 3.1 leaves gaps larger than h in the $d(\mathbf{x})$ computation since it is difficult to estimate a suitable value for h which predicts a correct reconstruction. The STEP `evolve` in Algorithm 3.1 will then provide the correct control on the hole filling during the surface reconstruction.

3.2 Solving the non-linear PDE diffusion equation

The computational method for solving (14) is based on an efficient semi-implicit co-volume scheme suggested in [10]. It provides an unconditionally stable, semi-implicit time discretization, and a 3D co-volume spatial discretization on a grid.

The equation we want to solve is (14) where $\phi(\mathbf{x}, t)$ is the unknown function, defined in $\Omega \times [0, T]$, and $\Omega \subset I\!R^3$ contains all the data points. The endpoint of the interval $[0, T]$ represents a time when the final reconstruction result is achieved. In practice T is chosen a posteriori.

The equation (14) is accompanied with Dirichlet boundary conditions

$$\phi(\mathbf{x}, t) = \phi^D \quad \text{in } \partial\Omega \times [0, T], \tag{25}$$

and with initial condition

$$\phi(\mathbf{x}, 0) = \phi^0(\mathbf{x}) \quad \text{in } \Omega. \tag{26}$$

In surface reconstruction we use Dirichlet boundary conditions and without loss of generality we may assume $\phi^D = 0$. We assume that initial state of the function is bounded, i.e. $\phi^0 \in L_\infty(\Omega)$. In particular we have considered

$$\phi^0(\mathbf{x}) = \frac{1}{\sqrt{\|\mathbf{C} - \mathbf{x}\|^2 + \gamma}}, \tag{27}$$

where $\gamma > 0.0$ can be considered as a regularization parameter, and $\mathbf{C}$ is a point inside the domain, typically the center of Ω. To further accelerate the convergence, we have also considered the distance function to initialize the function ϕ, that is

$$\phi^0(\mathbf{x}) = d(\mathbf{x}). \tag{28}$$

In our approach, the reconstruction is an evolutionary process given by the solution of equation (14). The stopping criterion is the change, in L_2-norm, of solution in time less than a prescribed threshold.

The mean curvature flow is weighted by the shape indicator function $g : I\!R^3 \to I\!R^+$, which provides useful information of the object we want to reconstruct. In fact, this function allows us to stop evolution of ϕ near the data set, i.e. where the distance is (almost) zero. We notice a similar behavior in the edge detection function $g(\|\nabla I\|)$, used in the Perona-Malik model for segmentation problems (∇I is the image gradient), where the evolution process is arrested when the segmentation function achieves the high gradient region, i.e. an edge zone [23].

In order to provide high quality reconstructions we introduced in (10) a term connected with the curvature of the surface. The idea is to increase the attraction of ϕ towards object's sharp features. In (14), where $0 \leq g(\mathbf{x}) \leq e$, $e = 2.71828182$, the motion of level sets is influenced by the surface features expressed in g.

3.2.1 Semi-implicit time discretization

For the time discretization of (14), we first choose a uniform discrete time step τ, then we replace time derivative in (14) by backward difference. The nonlinear terms of the equation are treated from the previous time step while the linear ones are considered on the current time level, this means semi-implicitness of the time discretization. By such approach we get the semi-discrete in time scheme:

Let τ be a fixed number, g be given by (10), ϕ^0 be a given initial function. Then, for every discrete time step $t_n = n\tau$, $n = 1, \dots N$, we look for a function ϕ^n, solution of the equation

$$\frac{1}{\|\nabla\phi^{n-1}\|_\varepsilon}\frac{\phi^n - \phi^{n-1}}{\tau} = \nabla.\left(g\ \frac{\nabla\phi^n}{\|\nabla\phi^{n-1}\|_\varepsilon}\right). \tag{29}$$

where $\|\nabla\phi^{n-1}\|_\varepsilon = \sqrt{\varepsilon^2 + \|\nabla\phi^{n-1}\|^2}$.

We observe that since $g(\mathbf{x})$ is computed as a pre-processed step on the data set in input, in the evolution (14) it remains the same at each time step.

3.2.2 Co-volume spatial discretization in three dimensions

The computational domain is obtained by decomposing Ω into cubic cells. According to [10] the construction of co-volume mesh has to use 3D tetrahedral finite element grid to which it is complementary. We denote by $\mathcal{T}$ this 3D tetrahedral grid.

In this method, only the centers of cells in Ω represent Degree of Freedom (DF) nodes, i.e. we solve the equation at a new time step updating the function only in these DF nodes. We denote these values ϕ_p^n, $p = 1, \dots, M$. Since there will be one-to-one correspondence between co-volumes, DF nodes and grid points, to avoid any confusion, we use the same notation for them. The co-volume mesh consists of M cells associated with DF nodes p of $\mathcal{T}$.

The choice of the number of co-volume M, that is of the volume grid size, is independent on the density of the initial unorganized data set, and it only affects the accuracy of the reconstruction as well as the computational complexity of the model. In fact, the difficulties given by a data set that is not uniformly sampled with some patches that are very dense and others that are very sparse, are efficiently managed by the computation of the analytical distance function, as discussed in Section 3.1. The choice of the grid size only effects the visualization of the results on different resolution and scale. Finer resolution will lead to more detailed reconstructions.

Following the notation in [10], for each DF node p let C_p be the set of all DF nodes q connected to the node p by an edge. This edge will be denoted by σ_{pq} and its length by h_{pq}. Then every co-volume p is bounded by the planes e_{pq} that bisect and are perpendicular to the edges $\sigma_{pq}, q \in C_p$.

We denote by $\mathcal{E}_{pq}$ the set of tetrahedras having σ_{pq} as an edge. In our configuration every $\mathcal{E}_{pq}$ consists of 4 tetrahedras.

As it is usual in finite volume methods [19, 13], we integrate (29) over every co-volume p obtaining

$$\int_p \frac{1}{\|\nabla\phi^{n-1}\|_\varepsilon}\frac{\phi^n - \phi^{n-1}}{\tau}dx = \int_p \nabla.\left(g\frac{\nabla\phi^n}{\|\nabla\phi^{n-1}\|_\varepsilon}\right)dx. \tag{30}$$

For the right hand side of (30) using divergence theorem we get

$$\begin{aligned}\int_p \nabla.\left(g\frac{\nabla\phi^n}{\|\nabla\phi^{n-1}\|_\varepsilon}\right)dx &= \int_{\partial p}\frac{g}{\|\nabla\phi^{n-1}\|_\varepsilon}\frac{\partial\phi^n}{\partial\nu}ds \\ &= \sum_{q\in C_p}\int_{e_{pq}}\frac{g}{\|\nabla\phi^{n-1}\|_\varepsilon}\frac{\partial\phi^n}{\partial\nu}ds.\end{aligned}$$

where ν denotes the unit normal to ∂p. So we have an integral formulation of (29)

$$\int_p \frac{1}{\|\nabla\phi^{n-1}\|_\varepsilon}\frac{\phi^n - \phi^{n-1}}{\tau}dx = \sum_{q\in C_p}\int_{e_{pq}}\frac{g}{\|\nabla\phi^{n-1}\|_\varepsilon}\frac{\partial\phi^n}{\partial\nu}ds \tag{31}$$

In every discrete time step t_n of the method (29) we have to evaluate gradient of ϕ at the previous time step $\|\nabla\phi^{n-1}\|_\varepsilon$. We consider a piecewise linear approximation of ϕ on this grid, such that its gradient has a constant value in tetrahedras. We will denote this constant value by $\|\nabla\phi_T\|$. Then we consider the value of g in the midpoint of σ_{pq}, denoting it by g_{pq}. For the approximation of the right hand side of (31) we get

$$\sum_{q\in C_p}\left(\sum_{T\in\mathcal{E}_{pq}} c_{pq}^T\frac{g_{pq}}{\|\nabla\phi_T^{n-1}\|_\varepsilon}\right)\frac{\phi_q^n - \phi_p^n}{h_{pq}}, \tag{32}$$

with c_{pq}^T is the area of the portion of e_{pq} that is in T, and the left hand side of (31) is approximated by

$$M_p^\varepsilon m(p)\frac{\phi_p^n - \phi_p^{n-1}}{\tau} \tag{33}$$

where $m(p)$ is measure in $\mathbb{R}^3$ of co-volume p and M_p^ε is an approximation of the capacity function $\frac{1}{\|\nabla\phi^{n-1}\|_\varepsilon}$ inside the finite volume p. If we define the coefficients

$$\begin{aligned}\beta_p^{n-1} &= M_p^\varepsilon m(p) \\ \alpha_{pq}^{n-1} &= \frac{1}{h_{pq}} \sum_{T\in\mathcal{E}_{pq}} c_{pq}^T \frac{g_{pq}}{\|\nabla\phi_T^{n-1}\|_\varepsilon}\end{aligned} \tag{34}$$

we get from (32)-(33) the following

Fully-discrete semi-implicit co-volume scheme: *Let $\phi_p^0,\ p = 1,\dots,M$ be given discrete initial values of the function. Then, for $n = 1,\dots,N$ we look for $\phi_p^n, p = 1,\dots,M$, satisfying*

$$\beta_p^{n-1}\ \phi_p^n + \tau \sum_{q\in C_p} \alpha_{pq}^{n-1}(\phi_p^n - \phi_q^n) = \beta_p^{n-1}\ \phi_p^{n-1}. \tag{35}$$

Applying boundary conditions, we get a system of linear equations which can be rewritten in matrix-vector form as

$$A\Phi = b, \tag{36}$$

with the coefficient matrix $A \in \mathbb{R}^{M\times M}$ which is a symmetric and diagonally dominant M-matrix, and $\Phi = (\phi_1,\dots,\phi_M)$ is the vector solution of the linear system.

The semi-implicit time discretization scheme used to numerically solve the formulation (14) turns out to be unconditionally stable as follows from the following result proved in [18].

Theorem 3.2 The linear system (36) obtained by the scheme (35), for any $\tau > 0, \varepsilon > 0$ and for every time step $n = 1,\cdots,N$ has a coefficient matrix that is symmetric and diagonally dominant thus a unique solution $\Phi^n = (\phi_1^n, \phi_2^n, \cdots, \phi_M^n)$ exists. Moreover, for any $\tau > 0, \varepsilon > 0$, the following L_∞ stability estimate holds

$$\min_p \phi_p^0 \le \min_p \phi_p^n \le \max_p \phi_p^n \le \max_p \phi_p^0, \quad 1 \le n \le N. \tag{37}$$

Any efficient iterative algorithm can be applied to solve the linear system (36), for example the PCG (preconditioned conjugate gradient) is suitable for sparse, symmetric, diagonally dominant M-matrices [27]. In the experiments presented in section 4 we used PCG and the iterative algorithm is stopped at the lth iteration if the norm of the residual is less than a tolerance $1\cdot 10^{-3}$. Incomplete Cholesky factorization is used as effective tool to construct efficient preconditioners for symmetric positive definite M-matrices. However, the system dimension can be significantly reduced if only grid points close to the region of interest could be involved in the computation. In the next section we introduce an efficient linear solver strategy suitable for reduced version of the linear system (36) which considers only a small portion Ω_1 of the grid domain Ω with a negligible penalization on the results.

3.3 Fast linear system resolution

The computation of an approximate solution of the PDE non-linear model (14) would require the update of the unknown function $\phi(x,t)$ for each co-volume of the domain Ω, at each time step n, which is a time consuming process. Since the unknown function $\phi(x,t)$ evolves only on nodes sufficiently close to the object boundary, we speed up the computation by determining the updated values for $\phi(x,t)$ only for the nodes $\mathbf{x} \in \Omega_1$ inside the spherical cover.

In practice, at each time step, we reduce significantly the number of unknowns in the linear system (36), thus saving both storage requirement and computational costs. Since at each row of A

corresponds a node in Ω, considering a limited number of nodes $M_1 << M$, we get a linear system with a sparse coefficient matrix which contains $M - M_1$ zero rows. Moreover, if we consider to apply a Krylov-type iterative method, like PCG or gmres method, where the main computational step involves a matrix-vector product with a vector of unknowns which contains $M - M_1$ zeros, then we end up with a matrix A which has $M - M_1$ zero rows and columns. It is easy to verify that applying a suitable permutation to rows and columns of A, and corresponding elements of b, we get a linear system with a coefficient matrix $\widetilde{A} \in \mathbb{R}^{M_1 \times M_1}$ with full rank which has the same epta-diagonal structure as A, that is, it's symmetric and positive definite. In a similar way, the same permutation applied to the components of the right-hand side vector b, leads to a vector $\widetilde{b} \in \mathbb{R}^{M_1}$. Therefore, instead of solving the linear system (36) which involves a $M \times M$ coefficient

	PCG	PCGs		PCG	PCGs		PCG	PCGs
dim	1000	144		1000	343		1000	512
nz	6400	1008		6400	2401		6400	3584
$\#its_{in}$	63	11		74	9		25	9
$\|res\|$	$6.5 \cdot 10^{-6}$	$5.5 \cdot 10^{-6}$		$1.6 \cdot 10^{-5}$	$1.4 \cdot 10^{-5}$		$3.9 \cdot 10^{-6}$	$5.8 \cdot 10^{-6}$
$\|err\|$	-	$3.6 \cdot 10^{-4}$		-	$1.9 \cdot 10^{-7}$		-	$4.7 \cdot 10^{-8}$

Table 1: Resolution of the linear systems (36) and (38) by pcg iterative solver. The Table reports the relative errors ($\|err\|$), the residual norms ($\|res\|$), the number of iterations ($\#its_{in}$), and the number of non-vanishing elements (nz) for matrices of dimension dim and full rank.

$\#its_{out}$	$\|err\|$
2	$5.6 \cdot 10^{-7}$
4	$1.7 \cdot 10^{-6}$
6	$2.5 \cdot 10^{-6}$
8	$2.7 \cdot 10^{-6}$
10	$2.8 \cdot 10^{-6}$

Table 2: Relative errors in the approximate solution ϕ as functions of the number of outer iterations ($\#its_{out}$).

matrix, we apply a Krylov-like iterative method for computing the solution $\widetilde{\Phi}$ of the linear system

$$\widetilde{A}\Phi = \widetilde{b}. \tag{38}$$

In general, iterative methods for solving linear systems (36) and (38) require to store only the non-vanishing elements of the coefficient matrix and to compute at each iterative step a matrix-vector multiplication which is inexpensive because of the lower dimension matrix.

In order to estimate the error between the approximated solutions $\widetilde{\Phi}$ and Φ we compare the results obtained by solving the linear systems (36) and (38) using the pcg algorithm with the stopping criterium driven by the tolerance $1 \cdot 10^{-6}$, for three simple tests.

Let us consider a data set of points on a cube, inside a grid of resolution 10^3, for a fixed time step n, with cubes defined by an increasing size, and corresponding increasing number of data points (one for each voxel). The dimension of the cube determines the number of unknowns in (38). In Table 1 we illustrate the results for the three tests. The results displayed in Table 1 columns 2-3 are related to cube dimension 4, in Table 1 columns 4-5 the cube dimension is 6, while in Table 1 columns $6 - 7$ the cube-dimension is 7. All the coefficient matrices have full rank.

In Table 1, for each of the three examples, the column denoted by *PCG* represents the solution of the linear system (36), while the column labelled *PCGs* reports the results obtained by solving (38). The rows in Table 1 display the dimensions of the matrices $A \in \mathbb{R}^{M \times M}$ and $\widetilde{A} \in \mathbb{R}^{M_1 \times M_1}$ denoted by dim, the non-vanishing elements by nz, the norm of the residuals, $\|res\|$, the relative

errors on the approximate solutions

$$\|err\| = \|\Phi - \widetilde{\Phi}\|_2 / \|\Phi\|_2,$$

and the number of iterations ($\#its_{in}$) required to solve the linear systems by the pcg method. The results are qualitatively the same from a visual inspection, and the relative errors reported in Table 1 confirm the goodness of the approximate solutions.

While in Table 1 we report the results related to a single time step, in Table 2, we consider the relative errors in the approximate solution $\widetilde{\Phi}$ for an increasing number of time iterations ($\#its_{out}$) in STEP `evolve` of Algorithm 3.1. At each time step the dimension of the linear systems (38) remains the same, and the error on the solution propagated by the function evolution is limited in time.

4 Experimental results

In this section we present some numerical examples of shape reconstructions obtained by applying the proposed algorithm on synthetic and real data sets. The computations are carried out on a CPU INTEL XEON $2GHz$ with $8Gb$ memory. All the reconstructions are on a $150 \times 150 \times 150$ grid and the resulting surfaces are displayed using the marching cube method in the Visualization Toolkit package [24], for rendering iso-surfaces from a 3D grid. In the following examples, the computation of the distance function, as discussed in Section 3.1, is based on inverse multiquadric RBFs, but similar results can be obtained by using Gaussian or compactly supported RBFs.

Example 4.1. In the first example we demonstrate the ability of Algorithm 3.1 of dealing with moderately large data sets representing complex shapes with small detailed features. The original data sets are the `teapot` (4255 points), the Stanford `bunny` (34835 points) and the `gargoyle` (10141 points).

For a better understanding of the evolution process computed by Algorithm 3.1 in STEP `evolve`, in Fig.2 a sequence of evolving steps is illustrated. All the images show the same iso-surface extracted by the evolving 3D manifold. The initial iso-surface given by (27) evolves following a mean curvature flow, where the diffusion is driven by the distance field and the curvature of the original data set. The level sets accumulate along the boundaries of the `bunny` and the evolution stops as soon as the solutions of two successively time steps differ for less than the tolerance $1 \cdot 10^{-4}$. All the experiments reported in this section have been obtained using this tolerance as input parameter TOL in Algorithm 3.1.

In Fig.3(a) the original cloud of points for the `gargoyle` data set is shown together with the corresponding reconstruction (Fig.3(b)) obtained after 10 time steps of Algorithm 3.1, using a time step $1 \cdot 10^{-3}$ and initial conditions given by (28). The quality of the details in the reconstructed surface strongly depends on the resolution of the 3D grid used in the STEP `evolve` of Algorithm 3.1. Finer resolution grids allow for a more accurate reconstruction, but are obviously much more computation demanding.

The `teapot` original data set shown in Fig.4(a) has been reconstructed by 16 time steps of Algorithm 3.1 using a time step $1 \cdot 10^{-3}$, with initial conditions given by (28), and the reconstructed surface is illustrated in Fig.4(b).

Example 4.2. The boolean operations of two implicit surfaces $\mathcal{M}_1$ and $\mathcal{M}_2$ can be carried out quite easy, using the min, max tools on the related signed distance functions $d_1(\mathbf{x})$ and $d_2(\mathbf{x})$, see, for example [32]. In fact the union, intersection and differences between two surfaces can be obtained by applying the evolving PDE (14) with shape indicator function (10) where $d(\mathbf{x})$ is defined respectively by

$$\begin{array}{ll} d(\mathbf{x}) = \min\{d_1(\mathbf{x}), d_2(\mathbf{x})\}, & \textbf{union} \\ d(\mathbf{x}) = \max\{d_1(\mathbf{x}), d_2(\mathbf{x})\} & \textbf{intersection} \\ d(\mathbf{x}) = \max\{d_1(\mathbf{x}), -d_2(\mathbf{x})\} & \textbf{difference}(\mathcal{M}_1 - \mathcal{M}_2) \\ d(\mathbf{x}) = \max\{-d_1(\mathbf{x}), d_2(\mathbf{x})\} & \textbf{difference}(\mathcal{M}_2 - \mathcal{M}_1). \end{array} \tag{39}$$

Examples of boolean operations are shown in Fig.5 where the data sets of a `sphere` and a `cube` have been composed to obtain respectively the union, intersection and difference (`cube` minus

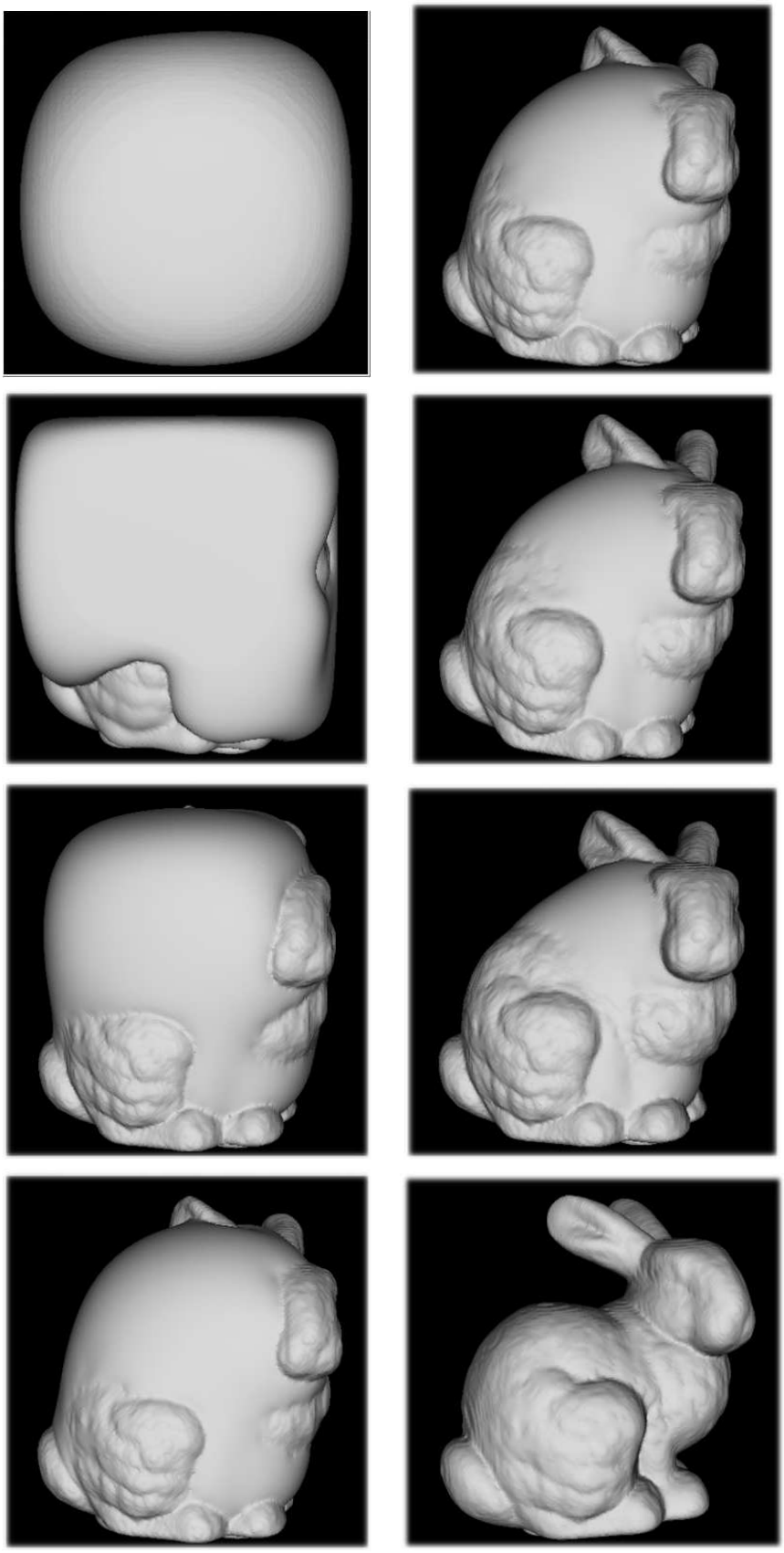
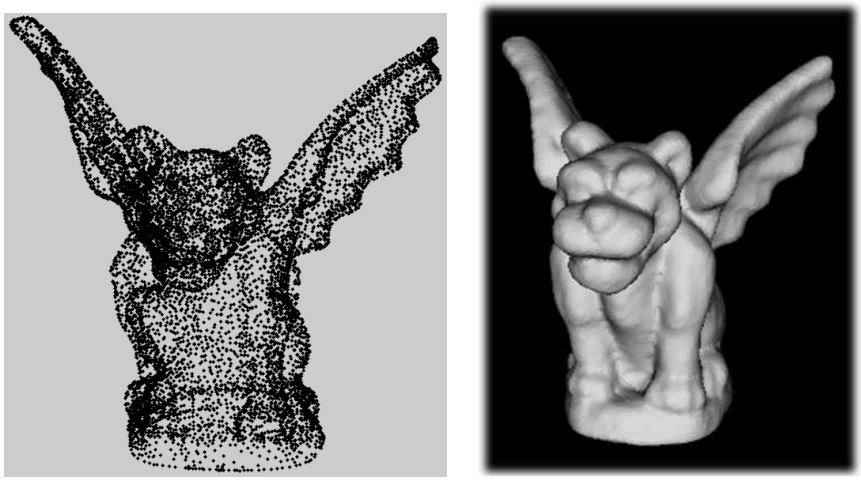
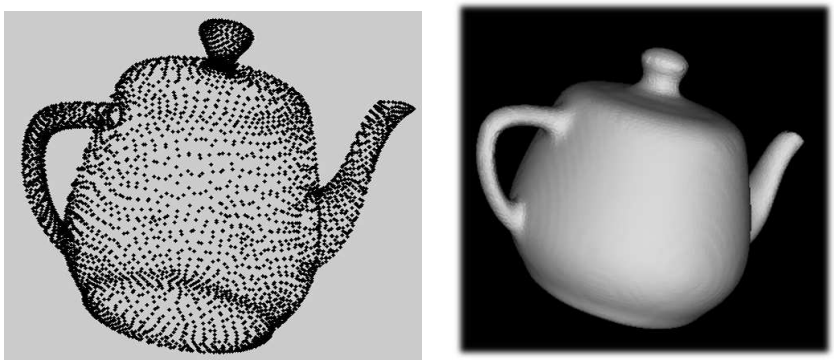


Figure 2: Example 4.1. Results of the surface evolution for the `bunny` reconstruction for different time steps.



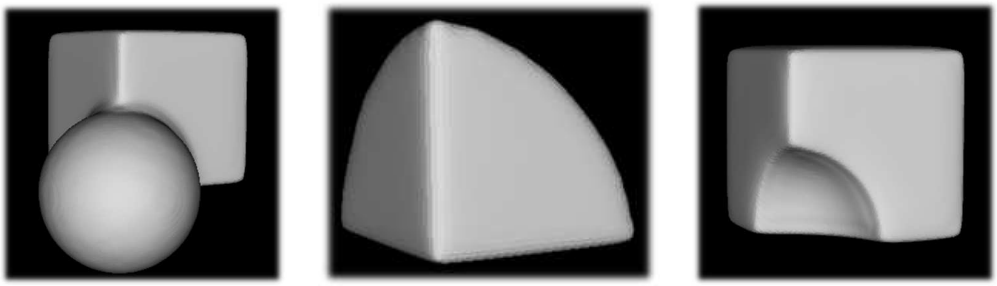
(a) (b)

Figure 3: Example 4.1. Reconstruction of the `gargoyle` data set: (a) cloud of points (b) reconstructed surface obtained by Algorithm 3.1.



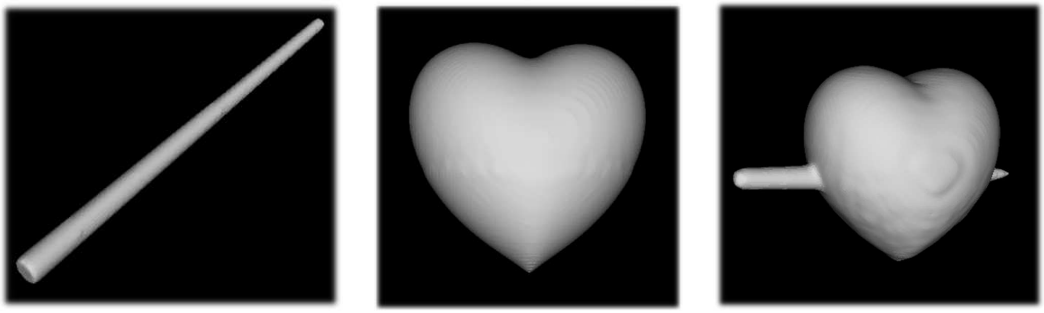
(a) (b)

Figure 4: Example 4.1. Reconstruction of the `teapot` data set: (a) cloud of points (b) reconstructed surface obtained by Algorithm 3.1.



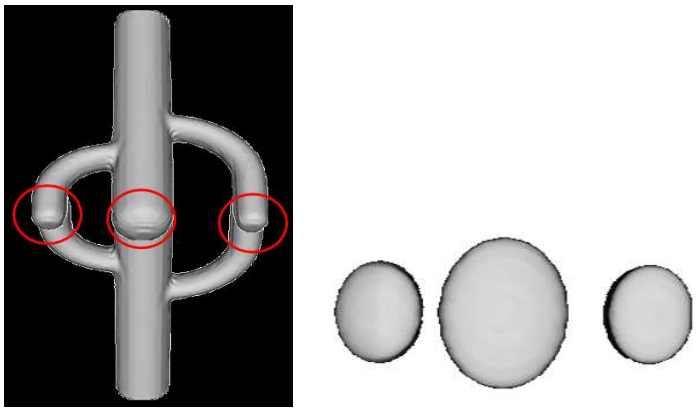
(a) (b) (c)

Figure 5: Example 4.2. Boolean operations between two data sets `cube` and `sphere`: (a) union (b) intersection (c) difference `cube` minus `sphere`.



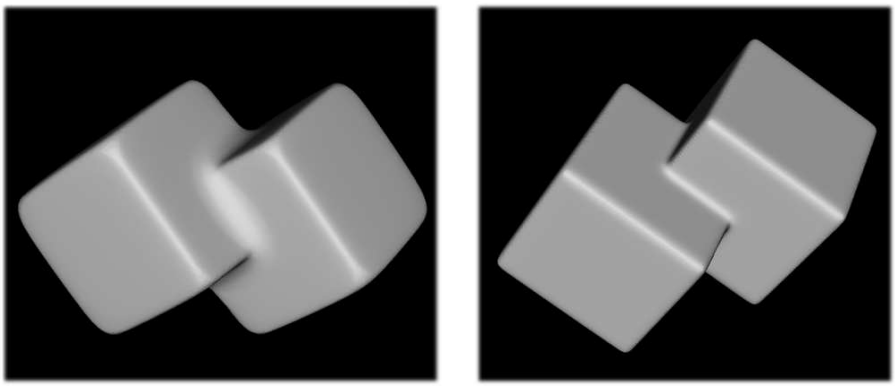
(a) (b) (c)

Figure 6: Example 4.2. The union between a `truncated cone` (a) and a `heart shaped` model (b) gives rise to the resulting surface in (c).



(a) (b)

Figure 7: Example 4.2. The intersection between two `cactus` data sets (a) produces three disconnected surfaces (b) .



(a) (b)

Figure 8: Example 4.2. Curvature effect on the shape reconstruction using the shape indicator function (10): (a) without curvature contribution (b) with curvature contribution.

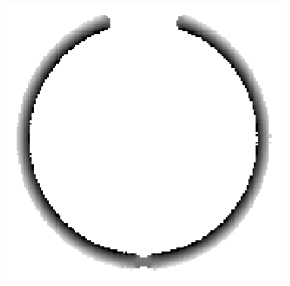
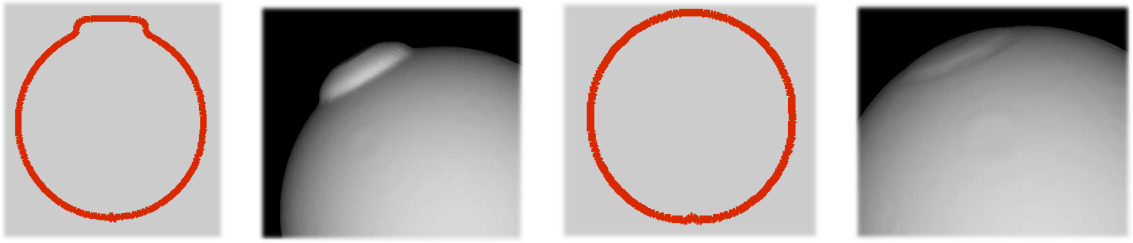


Figure 9: Example 4.3. Signed distance function computed on the data set `sphere` with two holes.



(a) (b) (c) (d)

Figure 10: Example 4.3. Completing missing boundaries of a `sphere` data set: (a) reconstruction using $\varepsilon = 1$, (b) corresponding silhouette, (c) reconstruction using $\varepsilon = 10^{-4}$, (d) corresponding silhouette.

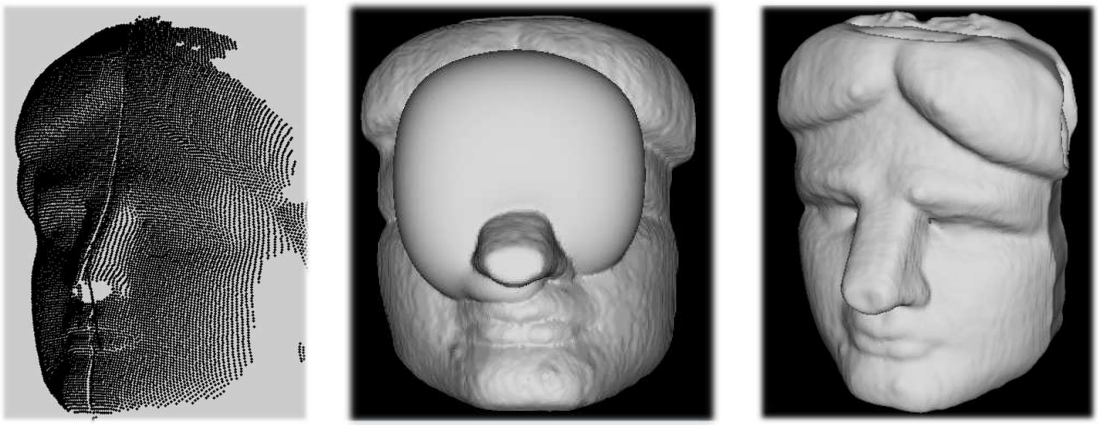
`sphere`) between the two surfaces. In Fig.5 the reconstructed resulting surfaces have been obtained after 50 time steps of Algorithm 3.1 using a time step $1 \cdot 10^{-3}$, with initial conditions given by (27). The distance function $d(\mathbf{x})$ has been computed by applying (39) to the signed distance functions $d_1(\mathbf{x})$ and $d_2(\mathbf{x})$ obtained by the STEP `pre-comp` in Algorithm 3.1.

In Fig.6, another example of boolean operations between a `truncated cone` and a `heart-shaped` model is shown. The latter is represented mathematically by a sixth order implicit polynomial function, $(2x^2 + y^2 + z^2 - 1)^3 - (0.1x^2 + y^2)z^3 = 0$. The surface union of the two shapes, shown in Fig.6(c), is reconstructed after 12 time steps by Algorithm 3.1 with initial conditions given by (28), using a time step $\tau = 5 \cdot 10^{-4}$.

The intersection of two `cactus` shapes shown in Fig.7 gives raise to three disconnected surfaces, obtained by applying 25 steps of Algorithm 3.1 with initial conditions given by (27), using a signed distance function $d(\mathbf{x})$ pre-computed using (39).

In Fig.8 we demonstrate the effect of the curvature in the final reconstruction of the union between two `cube` data sets. In Fig.8(a) the resulting shape is obtained by setting $g(\mathbf{x}) = d(\mathbf{x})$, while in Fig.8(b) the reconstruction is obtained using $g(\mathbf{x})$ defined as in (10), where the mean curvature k acts as an attractor which speeds up the evolution for high curvature values. Smoother corners and edges can be observed in Fig.8(a) where the evolution when the level sets approach to the data (and $d(\mathbf{x})$ approaches to zero values) tends to slow down and fails to capture sharp features.

Example 4.3. The ability of Algorithm 3.1 in completing missing boundaries is illustrated with two examples. In the first example, we removed two sets of adjacent points from a `sphere` data set, creating two circular holes, one bigger than the other. The signed distance function obtained by running STEP `pre-comp` in Algorithm 3.1 is illustrated in Fig.9. It reproduces the big hole, while slightly reduces the little one in the opposite side. The reconstruction by running 45 time steps of the STEP `evolve` in Algorithm 3.1 with $\varepsilon = 1$ and initial conditions given by (27), is shown in Fig.10(a) together with a slice section (Fig.10(b)) to better emphasize the silhouette



(a) (b) (c)

Figure 11: Example 4.3. Reconstruction of the `statue-head` data set: (a) cloud of points (b) reconstructed surface obtained by Algorithm 3.1 setting $\varepsilon = 1$ (c) reconstructed surface obtained by Algorithm 3.1 setting $\varepsilon = 10^{-4}$.

of the reconstructed shape. The evolving level sets are exiled from the expected real boundaries. The reconstruction obtained after 45 time steps using $\epsilon = 1 \cdot 10^{-4}$, illustrated in Fig.10(c) together with a slice section in Fig.10(d), completely recovers the shape boundaries.

The last example demonstrates the hole filling strategy on a data set `statue-head` with 9928 points, obtained by a 3D laser scanner system, which presents a long thin hole in correspondence of the forehead and nose (see Fig.11(a)). The 3D manifold evolution fills the gap after 25 time steps of the Algorithm 3.1 with initial conditions (28), when $\varepsilon = 1 \cdot 10^{-4}$, (see Fig.11(c)), while the object boundaries are not correctly recovered for $\varepsilon = 1$, as can be observed in Fig.11(b).

5 Comparison and conclusions

The examples in Section 4 have shown some of the good properties of the surface reconstruction method proposed which include good accuracy, faithful reproduction of sharp features, and robustness in the presence of holes and low sampling density. Since the surfaces are described using implicit functions, tools such as shape blending, offsets, and Boolean operations are simple to perform.

These preliminary results let us compare our approach with other existing implicit surface reconstruction algorithms, in particular, the approaches based on RBF partition of unity, and the methods driven by PDE deformable models, which share similarities with our work. In the following comparison we will take into account three important aspects in the reconstruction process from scattered data, which are the accuracy, the quality of the sampling data sets, and the computational performance of the method.

From the class of RBF partition of unity methods, our method inherits the accuracy of a local RBF reconstruction exploited in STEP `pre-comp` in Algorithm 3.1. However, the RBF partition of unity methods work well only when the data is densely sampled and the geometric structures to be reconstructed are simple, well defined and can be deduced from the samples themselves. In the presence of holes and low sampling density the second step of our algorithm produces a complete watertight model which provides a boundary completion without user-intervention. This second step makes our method computationally less appealing than a fast RBF partition of unity method. At this aim, we introduced a semi-implicit time discretization, and a strategy for accelerating the linear system solver involved in the numerical solution of formulation (14). Moreover, further developments on the PDE model (14), see [17], have demonstrated as an anisotropic variant of this model, which includes a diffusion tensor, can be easily adopted to reconstruct thin structures close

to each other. We refer the reader to [28], [6] for other approaches which provide topology-stable reconstructions.

From the class of reconstruction methods based on PDE deformable models, such as for example [31], [32], our method inherits the capability to provide reasonable reconstructions in case of arbitrary and complex topologies, holes and non-uniform data sets. However, methods belonging to this class suffer of a weak reconstruction accuracy and low computational efficiency, see Section 3.1 for more details. These problems are successfully solved by the proposed two-steps method. For example, to improve the recover of sharp features we have introduced in (14) a diffusion term driven by the curvature.

In the remaining of this section we comment on computational times, and on how to process non-uniform data set with considerable noise and huge data.

The running time depends mainly on the resolution of the grid on which we solve the equation (14), but also on the radius of the spherical cover for the computation of the distance function. The number of data points is important only in the pre-processing step, when the distance function and the curvature term have to be reconstructed close to the data set.

The main contribution to the total computational cost for each reconstruction is due to the linear system solver (36) invoked at each time step for the numerical solution of formulation (14). Using a 150^3 grid, the cost for each time step, in STEP `evolve` in Algorithm 3.1, is 25 seconds independently on the data set. However, when the fast linear system strategy is considered, thus replacing (36) with the reduced linear system (38), we have a significant performance improvement. For the data sets considered in the examples we get the following times for each time step: `teapot` 0.32 secs., `bunny` 2.97 secs., `gargoyle` 1.67 secs., `statue-head` 9.10 secs. The number of required time iterations for each reconstruction is reported for each data set in the example description.

The sizes of the computed examples are moderately large and have been chosen according to the available computational resources. Huge amount of data can be processed by the proposed algorithm on a more powerful computational platform.

In case of noisy data the RBF interpolation process we used in STEP `pre-comp` in Algorithm 3.1 for all the shown examples, can be easily replaced by a RBF least-squares approximation, as shown in [8], which provides a control on the smoothness of the reconstruction with the same computational effort. Moreover, increasing the radius of the spheres $B_i, \forall i$ in the spherical cover, leads to smoother reconstructions.

Finally, the ability of Algorithm 3.1 in completing missing boundaries, illustrated in Example 4.3, is a demonstration of the robustness of the algorithm in processing non-uniform data sets.

Acknowledgements This work has been supported by PRIN-MIUR-Cofin 2006, project, by "Progetti Strategici EF2006" University of Bologna, and by University of Bologna "Funds for selected research topics".

References

[1] N.Amenta,M.Bern,and M.Kamvysselis, A new Voronoi-based Surface Reconstruction Algorithm, in Proc. SIGGRAPH, ACM Press/ACM SIGGRAPH, pp. 415–420, 1998.

[2] N. Amenta, K.J. Yong, Defining point-set surfaces, in Proc. SIGGRAPH, ACM Press/ACM SIGGRAPH 2004, pp. 264–270, 2004.

[3] C.L.Bajaj, F.Bernardini,,and G.Xu, Automatic Reconstruction of Surfaces and Scalar Fields from 3D Scans, in Proc. SIGGRAPH, ACM Press/ACM SIGGRAPH, pp. 109–118, 1995.

[4] J.C. Carr, R.K. Beatson, J.B. Cherrie, T.J. Mitchell, W.R. Fright, B.C. McCallum, T.R. Evans, Reconstruction and Representation of 3D Objects with Radial Basis Functions, In *Proc. of SIGGRAPH 2001, ACM Press* pp.67–76 (2001).

[5] J.C. Carr, R.K. Beatson, B.C. McCallum, W.R. Fright, T.J. McLennan, T.J. Mitchell, Smooth surface reconstruction from noisy range data, In *Proc. of Graphite 2003, ACM Press* pp.119-126 (2003).

[6] Y. Lipman, D. Cohen-Or, D. Levin and H. Tal-Ezer, Parameterization-free projection for geometry reconstruction, ACM Transactions on Graphics, Vol. 26(3): 22, 2007.

[7] G. Casciola, D. Lazzaro, L.B. Montefusco and S. Morigi, Fast surface reconstruction and hole filling using Radial Basis Functions, *Numerical Algorithms* **39** pp.289–305 (2005)

[8] G. Casciola, D. Lazzaro, L.B. Montefusco and S. Morigi, Shape preserving surface reconstruction using locally anisotropic RBF Interpolants, *Computer and Mathematics with Applications* **51** pp.1185–1198 (2006)

[9] V. Caselles, R. Kimmel, and G. Sapiro, Geodesic active contours, International Journal of Computer Vision, Vol. 22, pp. 61-79, 1997.

[10] Corsaro S., Mikula K., Sarti A., Sgallari F., Semi-implicit covolume method in 3D image segmentation, SIAM J. Sci. Comput., Vol. 28, n. 6, pp. 2248-2265, 2006.

[11] Deckelnick, K. and Dziuk, G., Numerical approximation of mean curvature flow of graphs and level sets, in Mathematical aspects of evolving interfaces. (L. Ambrosio, K. Deckelnick, G. Dziuk, M. Mimura, V. A. Solonnikov, H. M. Soner, eds.), pp 53-87, Springer, Berlin-Heidelberg-New York, 2003.

[12] Deschamps, T., Malladi, R., Rawe, I., Fast evolution of image manifolds and application to filtering and segmentation in 3D medical images, IEEE Transactions on Visulization and Computer Graphics, Vol. 10, No. 5, pp. 525-535,2004.

[13] Eymard, R., Gallouet, T. and Herbin, R., The finite volume method, in: Handbook for Numerical Analysis, Vol. 7 (Ph. Ciarlet, J. L. Lions, eds.), Elsevier, 2000.

[14] M. Eck, H.Hoppe, Automatic Reconstruction of B-spline surfaces of Arbitrary Topological Types, Computer Graphics Porceedings, (Proc. SIGGRAPH, 1996), pp. 325–334, 1996.

[15] Evans, L.C. and Spruck, J., Motion of level sets by mean curvature I, J. Diff. Geom., Vol. 33, pp. 635-681, 1991.

[16] S. Fleishman, D. Cohen-Or, Daniel, C. Silva, Robust Moving Least-squares Fitting with Sharp Features, ACM Transactions on Graphics, Vol. 24(3), pp.544-552, 2005.

[17] E.Franchini, S.Morigi, F.Sgallari, Segmentation of 3D tubular structures by a PDE-based anisotropic diffusion model, Lecture Notes in Computer Science, Proceeding of Mathematical Methods for Curves and Surfaces, in press.

[18] Handlovičová, A., Mikula, K. and Sgallari, F., Semi–implicit complementary volume scheme for solving level set like equations in image processing and curve evolution, Numer. Math., Vol. 93, pp. 675-695, 2003.

[19] Le Veque, R., Finite volume methods for hyperbolic problems, Cambridge Texts in Applied Mathematics, Cambridge University Press, 2002.

[20] Malladi, R., Sethian, J.A. and Vemuri, B., Shape modeling with front propagation: a level set approach, IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 17, pp. 158-174, 1995.

[21] Osher, S. and Fedkiw, R., Level set methods and dynamic implicit surfaces,Springer-Verlag, 2003.

[22] Osher S., Sethian J., Fronts propagating with curvature dependent speed, algorithms based on a Hamilton-Jacobi formulation, J. Comp. Phys., Vol. 79, pp. 12-49, 1988.

[23] P. Perona and J. Malik, Scale-space and edge detection using anisotropic diffusion, IEEE Trans. Pattern Anal. Mach. Intell., **12** (1990), pp. 629–639.

[24] W.Schroeder, K.Martin, B.Lorensen, The Visualization Toolkit An Object-Oriented Approach To 3D Graphics, 4th Edition Kitware, Inc. publishers, 2006.

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

[26] H.Hoppe, T. DeRose, T. Duchamp, H.Jin, J.McDonald, and W.Stuetzle, Piecewise smooth surface reconstruction, Computer Graphics (Proc. SIGGRAPH, 1993), pp. 35–44, 1993.

[27] Saad, Y., Iterative methods for sparse linear systems, PWS Publ. Comp., 1996.

[28] A. Sharf, T. Lewiner, G. Shklarski, S. Toledo, D. Cohen-Or, Interactive Topology-aware Surface Reconstruction, ACM Transactions on Graphics, Vol. 26(3): 43, 2007.

[29] Sarti, A., Malladi, R. and Sethian, J.A., Subjective Surfaces: A Geometric Model for Boundary Completion, International Journal of Computer Vision, Vol. 46, No. 3, pp. 201-221, 2002.

[30] G. Turk, J.F. O'Brien, Modelling with Implicit Surfaces that Interpolate, *ACM Transaction on Graphics* **21,4** pp.855–873 (2002).

[31] 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 Metods in Computer Vision (VLSM 2001), 2001, pp. 194–202.

[32] Zhao H.K., Osher S., Merriman B., Kang M., Implicit and non-parametric shape reconstruction from unorganized data using a variational level set method, Computer Vision and Image Understanding, Vol. 80, pp. 295-319, (2000).

[33] H. Wendland, Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree, Adv. Comput. Math. 4 pp.389-396 (1995).

[34] H. Xie, Wang J., Hua J., Quin H., Kaufman A., Piecewise C^1 continuous surface reconstruction of noisy point clouds via local implicit quadric regression, IEEE Visualization 2003, pp. 91-98, 2003.