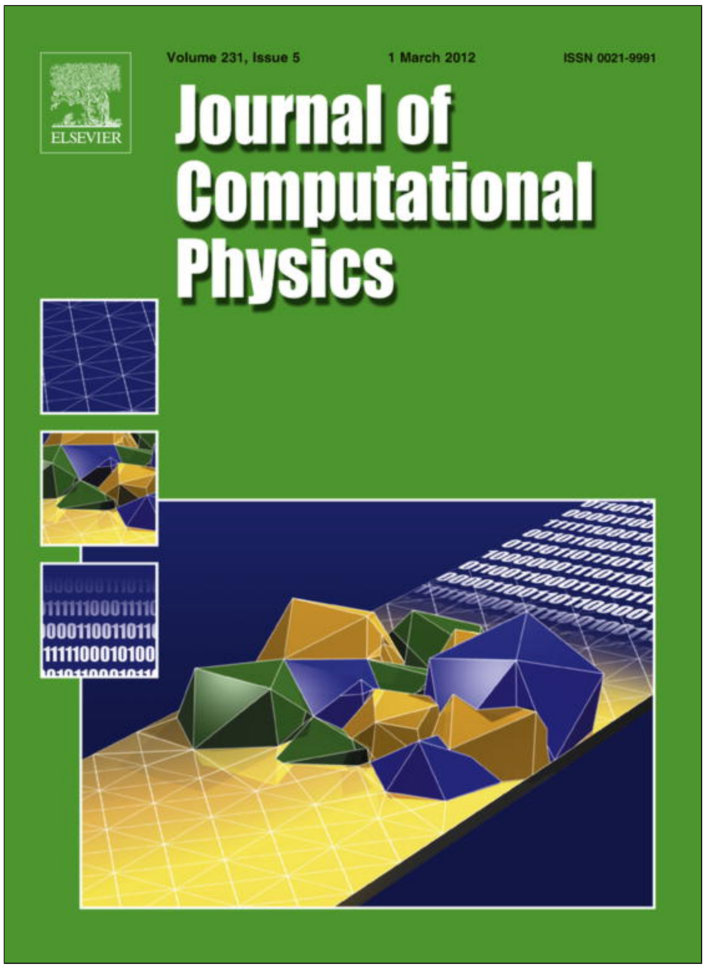
Volume 231, Issue 5
1 March 2012
ISSN 0021-9991
ELSEVIER
Journal of Computational Physics

Journal of Computational Physics 231 (2012) 2376–2387

Contents lists available at SciVerse ScienceDirect

Journal of Computational Physics

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

CAD and mesh repair with Radial Basis Functions

E. Marchandise [*,1], C. Piret [1], J.-F. Remacle

Université catholique de Louvain, Institute of Mechanics, Materials and Civil Engineering (iMMC), Avenue G. Lemaître, 4, 1348 Louvain-la-Neuve, Belgium

ARTICLE INFO

Article history:
Received 20 July 2011
Received in revised form 24 November 2011
Accepted 28 November 2011
Available online 8 December 2011

Keywords:
Geometry processing
Hole filling algorithm
Radial Basis Functions
RBF
Surface remeshing
Surface parametrization
STL file format
Surface mapping
Harmonic map
Orthogonal Gradients method

ABSTRACT

In this paper we present a process that includes both model/mesh repair and mesh generation. The repair algorithm is based on an initial mesh that may be either an initial mesh of a dirty CAD model or STL triangulation with many errors such as gaps, overlaps and T-junctions. This initial mesh is then remeshed by computing a discrete parametrization with Radial Basis Functions (RBF's).

We showed in [1] that a discrete parametrization can be computed by solving Partial Differential Equations (PDE's) on an initial correct mesh using finite elements. Paradoxically, the meshless character of the RBF's makes it an attractive numerical method for solving the PDE's for the parametrization in the case where the initial mesh contains errors or holes. In this work, we implement the Orthogonal Gradients method to be described in [2], as a RBF solution method for solving PDE's on arbitrary surfaces.

Different examples show that the presented method is able to deal with errors such as gaps, overlaps, T-junctions and that the resulting meshes are of high quality. Moreover, the presented algorithm can be used as a hole-filling algorithm to repair meshes with undesirable holes. The overall procedure is implemented in the open-source mesh generator Gmsh [3].

1. Introduction

Using CAD data for finite element analysis has become the actual standard in the engineering practice. Yet, geometries that come out of design offices are not free of problems: slivers, cross-overs, surfaces with multiples unnecessary patches, super-small model entities and many other issues that are encountered in the CAD data make the meshing process a nightmare. Those *dirty geometries* are still the cause of time consuming repair processes. The same kind of issues are present when dealing with STL triangulations as the input geometry: the mesh may be noisy, self-intersecting, not watertight, with T-junctions[2] and have undesirable holes. Fig. 1 gives an example of such dirty CAD models or STL triangulations that need to be repaired.

There are two approaches for cleaning dirty geometries: one acts on the CAD model and one acts on the mesh.

The first approach corrects the geometry directly by using point and edge merging algorithms [4–6]. Those approaches provide then specific tools for model correction, controlled primarily by the user [7,8]. Presently, there are also many commercial software modules that claim to be able to perform automatic geometry healing. However, these third party software modules can only rectify common geometry problems and a successful or unsuccessful outcome is possible. Thus there is yet no absolute solution for geometry/mesh healing of CAD models.

* Corresponding author. Tel.: +32 10472061; fax: +32 10472179.
E-mail addresses: emilie.marchandise@uclouvain.be (E. Marchandise), cecile.piret@uclouvain.be (C. Piret), jean-francois.remacle@uclouvain.be (J.-F. Remacle).
[1] These authors equally contributed to the work.
[2] A T-Junction is an intersection of two or more faces in a mesh where the vertex of one face lies on the edge or interior of another face.

0021-9991/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2011.11.033

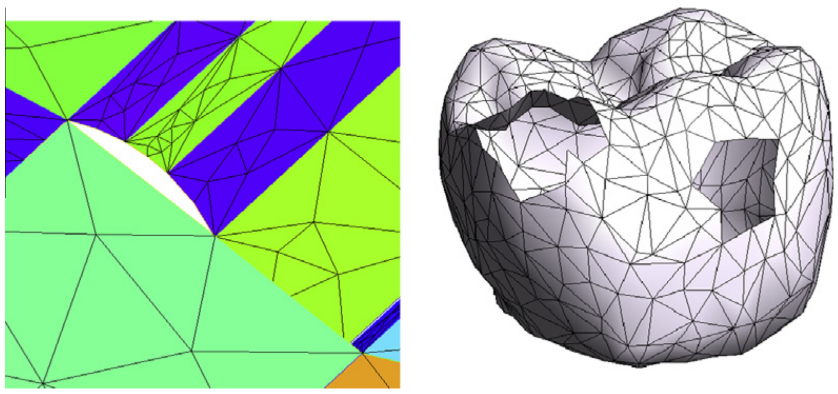

Fig. 1. Two examples for which a CAD and mesh repair algorithm is needed. The figure on the left shows an initial triangulation of a dirty CAD model with topological errors: gaps (holes), overlaps and T-junctions. The right figure displays an STL triangulation of a tooth that contains undesirable holes.

Another approach is that of correcting an initial triangulation of the model through the addition of triangles and different stitching procedures [9–11]. In the same vein, Nooruddin and Turk [12] proposed a method to repair polygonal meshes using volumetric techniques. Unfortunately, not all of those algorithms deal with geometric intersections and/or inconsistent topologies. Also based on an initial triangulation, some authors [13,14] suggested some specific hole-filling algorithms that employ RBF's as an interpolation technique to construct an implicit surface patch and to intersect this implicit surface with the existing triangulation. However, the intersection method to reconstruct the mesh for the hole is a complex process. Moreover, the resulting mesh may contain triangles of low quality when the holes to fill are quite small.

In this paper, an original alternative approach is presented. The repair process includes both model/mesh repair and mesh generation. The remeshing procedure relies on a discrete parametrization. Surface parametrization techniques [15–17] originate mainly from the computer graphics community: they have been used extensively for applying textures onto surfaces [18,19] and have more recently become a very useful and efficient tool for many mesh processing applications [17,20,21]. In the context of remeshing procedures, the initial surface is parametrized onto a planar surface, the surface is meshed using any standard planar mesh generation procedure and the new triangulation is then mapped back to the original surface [22,23]. In recent papers [1,24,25], our group showed that harmonic maps can be computed efficiently by solving partial differential equations (PDE's) on the initial triangulation with finite elements.

However, in the context of CAD and mesh repair, the initial triangulation may contain topological errors such as holes, T-junctions and overlaps. Classical mesh-based numerical techniques for solving PDE's such as finite elements or finite volumes cannot be used with such dirty meshes. The meshless character of the RBF's makes them an attractive alternative for solving those PDE's. Although the RBF method has been used as an interpolation technique since the 1970s, it is only in the 1990s that it was introduced as a technique to solve PDE's [26,27]. Its high accuracy and meshless character have made it the method of choice for problems set on complicated geometries [28]. Often overlooked for having poor stability and high complexity issues, the method has finally gained acknowledgement. A number of studies showed the method's great potential by solving full-scale geophysical applications, and by showing that RBF's could compete with the most trusted numerical techniques [29–33]. Although a good part of the RBF literature deals with surface reconstruction [34,35], or more recently PDE's over spheres [31,30,36], no technique has been developed to solve PDE's on arbitrary surface, until [2], which provides the very first methods, based on RBF's, for solving PDE's on completely arbitrary surfaces. In this work, we implement the RBF's Orthogonal Gradients method of [2] which relies on a level set representation of the initial surface.

The paper is organized as follows. In Section 2, we present the PDE's for solving the parametrization. In Section 3, we show how to solve the PDE's with the RBF Orthogonal Gradients method (OGr) centered on the mesh vertices of the initial triangulation. Section 4 explains how to choose the RBF parameters of the presented Ogr method. Next, we explain in Section 5 how to find the inverse mapping in order to be able to project the new points on the 3D surface. Finally, results are presented to validate the method and to reveal the efficiency and accuracy of our proposed algorithm.

2. Discrete parametrization

The discrete parametrization aims at computing the discrete mapping $\mathbf{u}(\mathbf{x})$ that maps every mesh vertex $\mathbf{x}$ of an initial triangulation of a surface Γ to a point $\mathbf{u}$ of Γ' embedded in $\mathcal{R}^2$:

$$\mathbf{x} = \{x, y, z\} \in \Gamma \subset \mathcal{R}^3 \mapsto \mathbf{u}(\mathbf{x}) = \{u, v\} \in \Gamma' \subset \mathcal{R}^2. \qquad (1)$$

We restrict ourselves to the parametrization of surfaces that can be mapped onto a subset of $\mathcal{R}^2$, which means that surfaces we deal with have their genus equal to zero and have at least one boundary. In some recent papers [24,25], efficient techniques have been developed to split objects with complex topologies into simpler ones in the context of surface remeshing.

In this work, we have chosen a harmonic mapping for the parametrization [1,22,23]. Harmonic maps $\mathbf{u}(\mathbf{x})$ can be computed by solving two PDE's that are the Laplace equations on the 3D surface Γ:

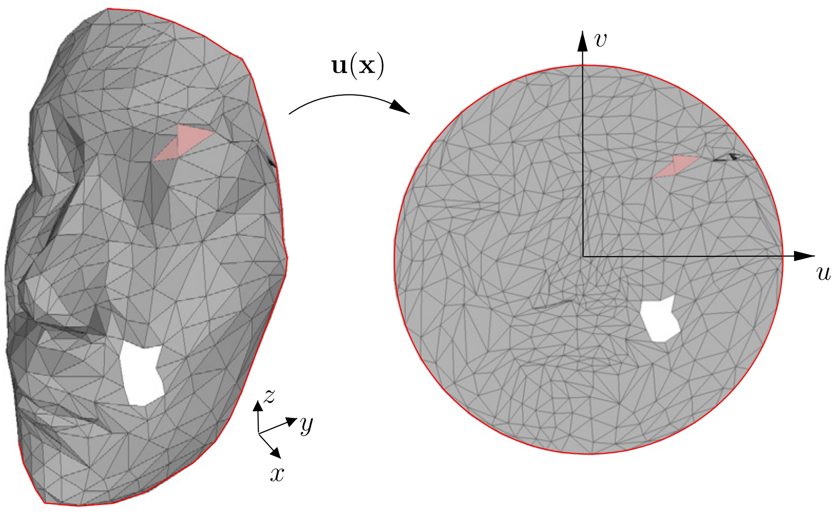

Fig. 2. Discrete parametrization. The harmonic map $\mathbf{u}(\mathbf{x})$ creates a correspondence between a 3D surface mesh (here a dirty STL triangulation with self-intersecting triangles) and a 2D mesh. The mapping is computed by solving the Laplace equations (2) on the 3D surface with Dirichlet boundary conditions on the boundary nodes (nodes on the red line). Those Dirichlet conditions are such that the surface is mapped onto a unit disk. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

$$\Delta u(\mathbf{x}) = 0, \quad \Delta v(\mathbf{x}) = 0, \quad \forall \mathbf{x} \in \Gamma, \tag{2}$$

with appropriate Dirichlet boundary conditions for one of the closed boundaries of the surface Γ,

$$u(l) = \cos(2\pi l/L), \quad v(l) = \sin(2\pi l/L) \tag{3}$$

and with Neumann boundary conditions for the other boundaries. In (3), l denotes the curvilinear abscissa of a point along the boundary of total length L.

Fig. 2 shows an example of discrete parametrization of mesh model of a face. As can be seen in Fig. 2, the triangulation contains some self-intersecting triangles and has some undesirable holes. The harmonic map $\mathbf{u}(\mathbf{x})$, computed by solving the Laplace Eq. (2), creates a correspondence between a 3D surface mesh (here a dirty STL triangulation with self-intersecting triangles) and a 2D mesh. Planar meshing algorithms can then be used in order to remesh and hence to repair the model of the face.

3. Parametrization with Radial Basis Functions

Different techniques have been recently suggested to solve PDE's on arbitrary surfaces with RBF's [2]. In this work, we use [2]'s Orthogonal Gradients method that relies on a level set representation of the initial surface. Let M be the number of points on the initial triangulation (red points in Fig. 3).

We use the multiquadric radial functions ϕ to represent the solutions (u, v) of the Laplace equations on Γ.[3]

$$u(\mathbf{x}) = \sum_{i=1}^{N} \lambda_i \phi(\|\mathbf{x} - \mathbf{x}_i\|), \quad \text{with} \quad \phi(r) = \sqrt{1 + \epsilon^2 r^2}, \tag{4}$$

where $\|\cdot\|$ denotes the Euclidean norm, $\mathbf{x}_i$ are the number of points N in the data set,[4] λ_i are the RBF's expansion coefficients, and ϵ is the shape parameter. All C^∞ smooth radial functions yield a similar accuracy, but we will limit ourselves to using multiquadric radial functions in this work as they have the advantage of allowing additional smoothing if necessary [34].

Let us find a matrix D that discretizes, via an RBF representation, a continuous differential operator L. We can rewrite the RBF interpolation (4) in matrix form for the N points:

$$U = A\Lambda, \quad \text{with} \quad A_{ij} = \phi(\|\mathbf{x}_i - \mathbf{x}_j\|), \quad U_i = u(\mathbf{x}_i), \tag{5}$$

where A is the RBF interpolation matrix of size $N \times N$ and Λ is the vector of expansion coefficients λ_i. Analytically applying the differential operator to the radial function interpolation $u(\mathbf{x})$ gives

$$u^L(\mathbf{x}_k) = \sum_{j=1}^{N} \lambda_j \underbrace{L\phi(\|\mathbf{x} - \mathbf{x}_j\|)_{\mathbf{x}=\mathbf{x}_k}}_{B^L_{kj}}, \tag{6}$$

[3] In the remainder of this section, we develop the Laplacian of u. The Laplacian of v can be developed in a similar way.

[4] We show in the remainder of this section that we take either $N = M$, $N = M + 1$ or $N = 3M$.

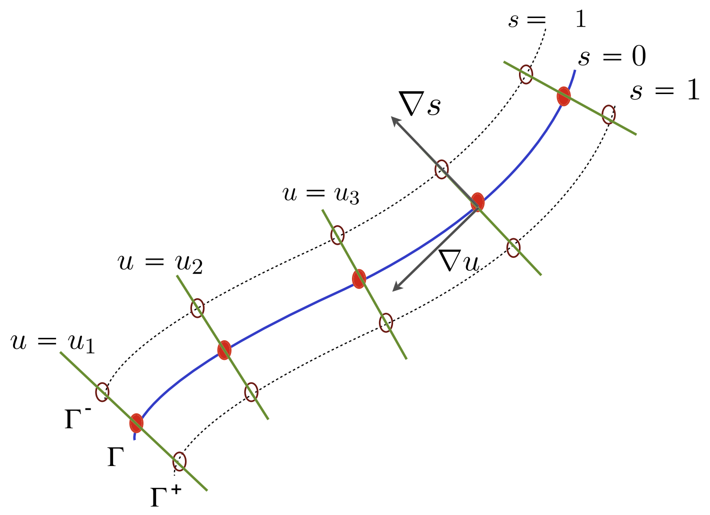

Fig. 3. Illustration of the RBF Orthogonal Gradients method. N points are uniformly distributed on the main surface Γ (in red). They are the mesh vertices of the initial triangulation. At each point, the normal **n** to the surface is computed and 2 new points are obtained at distance δ from the surface, one on either side of Γ. One can see these $2M$ additional points on the two gray layers (which we call Γ^+ and Γ^-) that surround Γ. We have now a set of $N = 3M$ points for two RBF expansions on these $3M$ nodes, one for the level set distance function $s(\mathbf{x})$ and one for the solution to the PDE $u(\mathbf{x})$. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

where u^L is the value of the differential operator applied to u at each $\mathbf{x}_k$. Thus we can write the differentiation of U in matrix form:

$$U^L = B^L \Lambda, \tag{7}$$

where matrix B^L is of size $K \times N$, and K is the set of points on which we wish to compute the differentiation. For example, the discrete Laplacian of u at points $\mathbf{x}_k$ can then be written as $(B^\Delta \Lambda)$. By combining (5) and (7), we can eliminate the expansion coefficient vector Λ and write:

$$U^L = \underbrace{B^L A^{-1}}_{D^L} U, \tag{8}$$

where D^L (of size $K \times N$) is the matrix of differentiation, i.e. the discretization of L. Note that if L is the identity operator I, then D^I is an interpolation matrix that interpolates the values of $u(\mathbf{x}_i)$ at points $\mathbf{x}_k$.

We wish now to compute the Laplacian operator on the 3D surface Γ by using the Orthogonal Gradients method of Piret [2] (see an illustration in Fig. 3). This method is loosely inspired from the closest point method [37] in that the goal is to make that function constant in the direction normal to the surface Γ. If that is the case, the Laplacian contribution from the normal direction vanishes and leaves only the Laplacian contribution from the tangential direction, i.e. the surface Laplacian. Hence, by canceling the derivative of u in the direction normal to the surface:

$$\nabla_{\mathbf{n}} u = 0, \quad \nabla_{\mathbf{n}} \cdot \nabla_{\mathbf{n}} u = 0, \tag{9}$$

we only keep the surface Laplacian on Γ:

$$\Delta u = (\nabla_{\mathbf{n}} + \nabla_{\mathbf{t}}) \cdot (\nabla_{\mathbf{n}} + \nabla_{\mathbf{t}}) u = \Delta_{\mathbf{t}} u. \tag{10}$$

In (10), **n** and **t** denote respectively the surface normal and tangential directions.

In order to have a reliable RBF representation of the surface Γ and to reliably compute the surface normals, it is now quite usual [28,35] to introduce additional points, on either side of the surface, and to define the level set distance function as $s(\mathbf{x}) = 0$ on Γ, $s(\mathbf{x}) = \pm 1$ on $\Gamma^\pm$, where $\Gamma^\pm$ are surfaces that surround and that are parallel to Γ (see Fig. 3). We thus extend the original M points of the surface inward and outward as follows:

$$\mathbf{x}_i^\pm = \mathbf{x}_i \pm \delta \mathbf{n}(\mathbf{x}), \tag{11}$$

where $\mathbf{n}(\mathbf{x})$ is the unit normal at point $\mathbf{x}_i$ and δ is the offset parameter. The normals are computed from the differentiation of an RBF expansion for a smooth function $\sigma(\mathbf{x})$ that is given a value of $\sigma(\mathbf{x}) = 0$ (as observed in [38], RBFs are bad at representing non-zero constants) on the M points on the surface and a value of $\sigma(\mathbf{x}) = 1$ on an additional point located far away from the surface:[5]

$$\mathbf{n} = \nabla \sigma. \tag{12}$$

[5] This additional point guarantees that the normal is always well defined even on planar surface parallel to the coordinate axis.

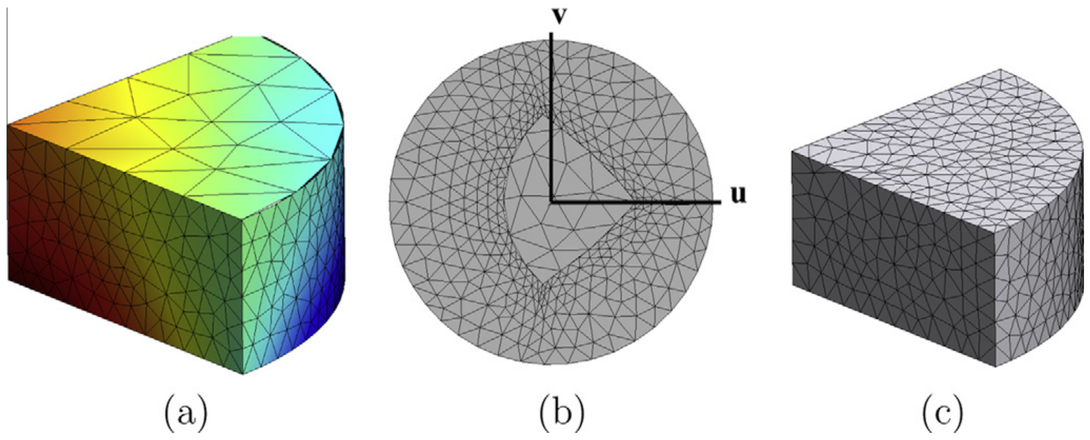

Fig. 4. Parametrization with RBF's of a mesh of a dirty CAD model. The mesh has T-joints with non-conforming elements. We show (a) the solution $u(\mathbf{x})$ to the Laplace equation, (b) the parametric space $\{u, v\}$ which is the solution to both Laplace equations and (c) the new mesh of the initial CAD that has conformity along common edges.

In (12), the three components of the gradient of σ are computed using Eq. (8).

We have now a set of $N = 3M$ points on which we can define two RBF expansions on these $3M$ nodes, one for the level set distance function $s(\mathbf{x})$ and one for the solution to the PDE $u(\mathbf{x})$.

The high order version of the RBF Orthogonal Gradients method (OGr) is used for the computation of the discrete surface Laplacian on the $3M$ points. It consists of adding the two constraints (9) together with the three-dimensional RBF Laplacian of u (see Eq. (6)). Those two constraints enable us to restrict the 3D Laplacian to the surface. We can then write the following augmented linear system of can then be written as $\widetilde{B}^L\Lambda = \bar{0}$:

$$\underbrace{\begin{pmatrix} \Delta\Phi(\mathbf{x}) \\ \mathbf{n}.\nabla\Phi(\mathbf{x})_{\mathbf{x}=\mathbf{x}_i} \\ (\mathbf{n}.\nabla)(\mathbf{n}.\nabla)\Phi(\mathbf{x})_{\mathbf{x}=\mathbf{x}_i} \end{pmatrix}}_{\widetilde{B}^L} \underbrace{\begin{pmatrix} \Lambda^0 \\ \Lambda^+ \\ \Lambda^- \end{pmatrix}}_{\Lambda} = \begin{pmatrix} \bar{0} \\ \bar{0} \\ \bar{0} \end{pmatrix}, \tag{13}$$

where $\widetilde{B}^L$ is of size $3M \times 3M$, $\Phi(\mathbf{x})$ is a $M \times 3M$ matrix whose entries are RBF radial functions, and the upperscripts ^{0}, $^+$ and $^-$ stand for the nodes that are respectively on Γ, Γ^+ and Γ^- (see Fig. 3).

In (13), all the partial derivatives of radial functions (i.e. the B^L matrixes introduced in Eq. (6)) can be computed analytically.[6] We give in the next equations some details of Eq. (13):

$$\begin{aligned} &\Delta\Phi(\mathbf{x}) = B^{\Delta} \\ &\mathbf{n}.\nabla\Phi = s_x B^{\partial x} + s_y B^{\partial y} + s_z B^{\partial z} \\ &(\mathbf{n}.\nabla)(\mathbf{n}.\nabla)\Phi = (s_x)^2 B^{\partial xx} + (s_y)^2 B^{\partial yy} + (s_z)^2 B^{\partial zz} + 2s_x s_y B^{\partial}_{xy} + 2s_x s_z B^{\partial}_{xz} + 2s_y s_z B^{\partial}_{yz} + (s_x s_{xx} + s_y s_{xy} + s_z s_{xz}) B^{\partial x} \\ &\qquad + (s_x s_{xy} + s_y s_{yy} + s_z s_{yz}) B^{\partial y} + (s_x s_{xz} + s_y s_{yz} + s_z s_{zz}) B^{\partial z}. \end{aligned}$$

Using (5), we can eliminate the vector of expansion coefficients Λ and come up with a linear system of size $3M \times 3M$:

$$(\widetilde{B}^L A^{-1})U = F, \tag{14}$$

where F is the right hand side that is zero for all the $3M$ entries, except for the j^{th} entries $(1 < j < M)$ corresponding to the boundary nodes. Indeed, the Dirichlet boundary conditions (3) on the boundary nodes of $\mathbf{x}_i$ are applied directly on the linear system (14) by setting the entries of the corresponding line to zero, the diagonal term to 1 and the right hand side to the cosine value (3). The solution of this linear system is the solution of the Laplace's equation for u (2). Following the same procedure, we can also solve Laplace's equation for v. The solution of those two Laplace's equations (u, v) is then the parametrization $\mathbf{u}(\mathbf{x})$ of the 3D surface onto a 2D surface.

In order to illustrate the parametrization with RBFs, we show in Fig. 4, the parametrization of a mesh of a dirty CAD model. As can be seen on the Fig. 4a, the mesh has T-joints with non-conforming elements. We show the solution $u(\mathbf{x})$ to the Laplacian computed by solving (14) with the M nodes of the initial triangulation. The parametrization $\mathbf{u}(\mathbf{x}) = \{u(\mathbf{x}), v(\mathbf{x})\}$ of the initial mesh (shown in Fig. 4b) is then given to planar meshing algorithms in order to create a new mesh (see Fig. 4(c)).

[6] See [28] for a few formulas.

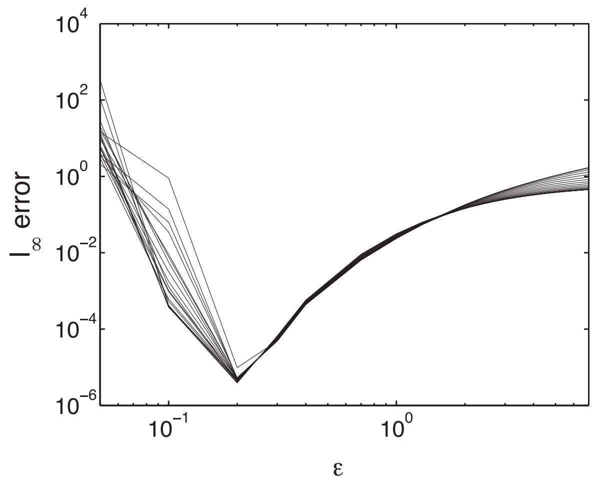

Fig. 5. Relative l_∞ error, versus the shape parameter ϵ^*, in applying the surface Laplacian on the unit sphere to the spherical harmonic $Y_{15}^3(x, y, z)$. We show the error versus ϵ^*, for a set of δ^* values, such that $0 < \delta^* < 1$. The experiment was conducted on a set of 900 nodes, near-uniformly distributed on the surface of the sphere.

4. RBF parameters

The only two parameters that need to be properly defined are the shape parameter ϵ defined in (4) and the offset parameter δ defined in (11). Both of these parameters have a direct impact on both the accuracy and the conditioning of the differentiation matrix. A lot of work has been done on the search for an 'optimal' shape parameter ϵ. Since the topology and the nodes' distribution and density also impact the conditioning, finding a formula for this 'optimal' ϵ is near impossible. As ϵ gets smaller, the accuracy improves but the conditioning worsens [28,39]. Unless one uses one of the algorithms that bypass the small shape parameter conditioning issue [36,40,41], the emphasis in setting the parameters has to be put on keeping a good conditioning. Since we know the smallest distance between two nodes ($d_{\min}$), and that the nodes are uniformly distributed, we can 'normalize' the shape parameter as $\epsilon = \frac{\epsilon^*}{d_{\min}}$. Parameter ϵ^* will vary with the total number of nodes (global method) or with the number of nodes in a cluster (local method). A lot less work has been made in finding an optimal δ. In order to avoid level sets crossing, we set $\delta = d_{min} \times \delta^*$, where $0 < \delta^* < 1$. For our examples, which feature relatively small sets of nodes, we set $\delta^* = 0.33$ and $\epsilon^* = 0.5$. If M were to be much larger, ϵ^* would need to grow with M. Fig. 5 shows, for different values of δ^*, the l_∞ norm of the error, versus ϵ^*, in applying the Laplacian to the spherical harmonic $Y_{15}^3(x, y, z)$. We notice the error behavior described above, in which the error decreases with ϵ^* until the error abruptly grows due to the bad conditioning associated with very small shape parameters. We also notice that the value of δ does not have a big influence on the error in this case.

5. Inverse mapping x(u)

Once the harmonic map $\mathbf{u}(\mathbf{x})$ has been computed, it can be used as input for planar surface meshers (delaunay, frontal, etc.) to produce high quality triangulations [1]. This is illustrated in Fig. 6. The top figures show a mesh of a CAD model with topological and geometrical errors such as gaps, overlaps and T-junctions together with its parametrization $\mathbf{u}(\mathbf{x})$. The bottom figures of Fig. 6 show the remeshing procedure. We can see an intermediate step of the frontal mesh algorithm[7] in the parametric space as well as the resulting mesh in the 3D space.

The only information we need to provide to those meshers is the inverse map $\mathbf{x}(\mathbf{u})$ and the Jacobians of the mapping $\mathbf{x}_u$ and $\mathbf{x}_v$ for very new point $\mathbf{u}_i$ to be inserted by the planar mesh algorithms (see the blue and green point in Fig. 6). For example, the jacobians are used by the meshing algorithms for computing edge lengths, angles and areas from the metric tensor

$$\mathbf{M} = \mathbf{x}_\mathbf{u}^T \mathbf{x}_\mathbf{u} = \begin{bmatrix} \mathbf{x}_u \cdot \mathbf{x}_u & \mathbf{x}_u \cdot \mathbf{x}_v \\ \mathbf{x}_v \cdot \mathbf{x}_u & \mathbf{x}_v \cdot \mathbf{x}_v \end{bmatrix}. \tag{15}$$

In this section, we explain how to compute the inverse mapping. We already have the solution of the harmonic map at the M points of the mesh: $\mathbf{u}_i(\mathbf{x}_i)$. The planar mesh algorithm needs to insert a new point $\mathbf{u}_p$ in the parametric space and needs therefore to know $\mathbf{x}_p(\mathbf{u}_p)$ as well as the mesh metric in order to have edge length and angle measures. Fig. 6 (right) shows such an example of two point insertions (blue and green points) during the frontal mesh algorithm: the green point belongs to a parametrized triangle and the blue point lies inside a hole.

[7] The gray triangles are the resolved layers, the red the active layers and the white the waiting layers.

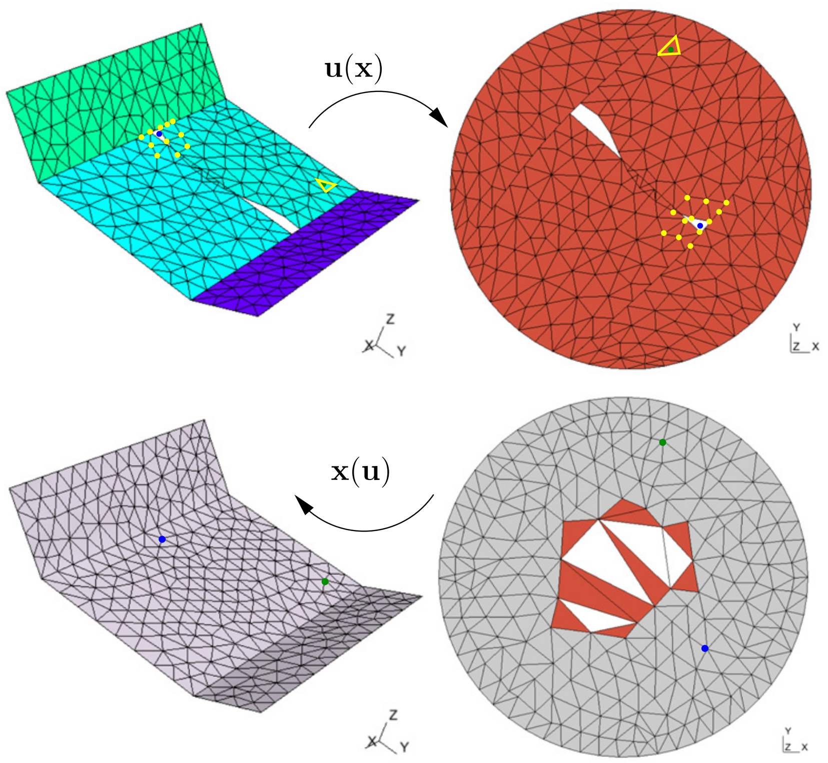

Fig. 6. Initial triangulation of a CAD model with topological and geometrical errors such as gaps, overlaps and T-junctions (top left) and discrete parametrization $\mathbf{u}(\mathbf{x})$ computed with RBF's (top right). One can see clearly the holes and overlaps of the initial mesh in the parametrization. The bottom figures show respectively the new mesh computed with a frontal mesh algorithm in the 3D space (left) and parametric space (right) (at an intermediate stage of the frontal algorithm). For the remeshing algorithm (bottom figures), we need to give to the planar mesher the inverse mapping $\mathbf{x}(\mathbf{u})$. How to find this inverse mapping is illustrated for two specific points (blue and green) and will be explained in Section 5. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Let suppose that our first mesh is obtained by meshing different CAD patches that have a given parametric description $\mathbf{u}^{CAD}$. In the case the new mesh point to be inserted by the meshing algorithm, lies within a mesh triangle T_j (green point in Fig. 6), then the inverse mapping can be found as follows [1].

1. Find the unique triangle T_j of the parametric space Γ' that contains point $\mathbf{u}$ (yellow triangle in Fig. 6); Note that in the case the parametrization contains overlaps or intersections that come from the dirty model, then any of the triangles containing the point $\mathbf{u}$ can be chosen;
2. Compute local coordinates $\xi = (\xi, \eta)$ of point $\mathbf{u}$ inside triangle T_j;
3. Compute the parametric coordinate of the CAD patch as follows: $\mathbf{u}^{CAD}(\xi, \eta) = (1 - \xi - \eta)\mathbf{u}_1^{CAD} + \xi \mathbf{u}_2^{CAD} + \eta \mathbf{u}_3^{CAD}$.
4. Use the CAD model to obtain the inverse mapping $\mathbf{x}(\mathbf{u}^{CAD})$ and the derivatives of this mapping.

Suppose now that the point to be inserted does not lie within any parametrized triangle (i.e. a point inside a hole such as the blue point in Fig. 6) or suppose that the triangulation does not have an underlying CAD model (STL triangulations obtained from image segmentation). In this case, the inverse mapping $\mathbf{x}_p$ for a point $\mathbf{u}_p = (u_p, v_p)$ is computed with a local RBF interpolation of the form (8):

$$X^L = \underbrace{B_{\mathbf{u}}^L A_{\mathbf{u}}^{-1}}_{D_{\mathbf{u}}^L} X, \qquad (16)$$

where X is the vector of the closest P points to $\mathbf{x}_i$. Those closest points are found by computing the P closest[8] parametrized points $\mathbf{u}_i$ to $\mathbf{u}_p$ in the set of the M parametrized points (see the yellow points in Fig. 6) and by taking the corresponding points $\mathbf{x}_i$ in the 3D space. In (16), X^L is the inverse mapping we are looking for (of size $K = 1$) and $D_{\mathbf{u}}^L$ is the matrix of differentiation in the parametric space. This matrix is of size $K \times P$ and the subscript $\mathbf{u}$ indicates that the radial basis functions are written in the parametric space: $\phi(\mathbf{u} - \mathbf{u}_i)$. From (16), we obtain easily the inverse quantities needed by the planar meshes. Indeed, when $L = I$, we get the inverse map $\mathbf{x}_p(\mathbf{u}_p)$, when $L = B^{\partial_u}$, we obtain the Jacobian $\mathbf{x}_{u,p}$ and when $L = B^{\partial_v}$, we have the Jacobian $\mathbf{x}_{v,p}$.

[8] A fast kdtree method is used for computing those closest points.

From Fig. 6, we can see that the proposed method can be used as a hole-filling algorithm. Indeed, thanks to the RBF interpolation, we are able to find new mesh vertices $\mathbf{x}_p$ corresponding to parametrized points $\mathbf{u}_p$ that are located inside undesirable holes.

6. Complexity

Conventional RBF interpolations are quite expensive since they require $\mathcal{O}(N^2)$ storage for the interpolation and $\mathcal{O}(N^3)$ arithmetic operations. Moreover, the Orthogonal Gradients method (OGr) used to solve the Laplacian restricted to a surface needs $N = 3M$ data points, where M is the number of mesh vertices of the initial triangulation.

However, different algorithmic advances involving RBF-generated finite differences (as in [29]), hierarchical and fast-multipole like methods combined with interpolatory filters could be implemented in order to reduce the computational cost to $\mathcal{O}(N)$ for storage and $\mathcal{O}(N \log N)$ for the arithmetic operations [42]. We note that fast algorithms involving compact radial functions, such as Wendland functions, will not be useful in this context. While the interpolation matrix produced will be sparse, no such guarantee will be given about its inverse, thus also about the differentiation matrix, defined as a product of this inverse.

As such asymptotically faster methods are out of the scope of this work, we have decided to report our results with the original $\mathcal{O}(N^3)$ method. Meanwhile, we have reduced the overall computational time by taking a reduced number of mesh vertices $N = 3m$, with $m < M$ as input for the RBF interpolations. The solution at the original M mesh vertices is recovered using classical RBF interpolations (see Eq. (4)). The reduction of the amount of data points is performed using quadratic edge collapse decimation or classical clustering techniques [43].

7. Results

The different examples aim to show that the presented algorithm can be seen as a CAD and mesh repair algorithm but also a quality remeshing algorithm, and a hole filling-algorithm.

In the first example we consider a dirty CAD model from a rocket reinforcement of a vehicle. Most of the time, a straightforward meshing of the patches of a clean CAD does not give a suitable mesh for finite element analysis. An efficient manner to build a high quality mesh for those CAD models is then to build from the initial CAD mesh a cross-patch parametrization that enables the remeshing of merged patches. Indeed, as most surface mesh algorithms mesh model faces individually, mesh points are generated on the bounding edges of those patches and if thin patches exist in the model they will result in the creation of small distorted triangles with very small angles. Those low quality elements present in the surface mesh will often hinder the convergence of the FE simulations on those surface meshes. An efficient manner to build a high quality mesh for those CAD models is then to build from the initial CAD mesh a *cross-patch parametrization* that enables the remeshing of merged patches. The new mesh is then build in the cross-patch parametric space and the new points are projected back onto the CAD patches using the parametric representation of the patches $\mathbf{u}^{CAD}$ (e.g. NURBS). Now, if the CAD is dirty such as the example of Fig. 7, standard techniques for computing cross-patch parametrizations will fail and the method we present in this paper can be considered as an original approach for creating a finite element mesh based of the dirty CAD that does that bypasses the geometrical repair. The dirty CAD model of Fig. 7 is given in IGES format, and includes 141 NURBS surfaces. The CAD model contains a lot of gaps, overlaps, redundant surfaces, T-joints and patches with reverted normals. This CAD model has so many topological and geometrical errors that none our available commercial packages that claim to perform geometrical CAD healing was able to repair the model.

Eighteen groups of patches (also called *compounds*) have been created and a cross-patch parametrization has been computed with the presented method for every of those compounds. This is implemented quite simply in Gmsh by creating a.geo file that reads:

```
Mesh.RemeshParametrization = 2;// (2) for rbf
Merge ‘‘CAD.iges";

Mesh.CharacteristicLengthFactor = 0.1;
Compound Surface (20000) = {34:65,118};
```

The new mesh (Fig. 7 bottom) is obtained with a planar Delaunay mesh algorithm with a given uniform mesh size field. This new mesh is made of 13216 triangles that have a high mean quality of $\bar{\kappa} = 0.94$. Parameter κ is such that an equilateral triangle has $\kappa = 1$ and a degenerated flat triangle has $\kappa = 0$. The total time for meshing the compounds is 40 s.

The next example highlights the capability of the presented method to be used as a hole filling algorithm. Fig. 8 shows a mutilated mesh of a skull and the mesh after the remeshing and hole filling process. Fur the purpose of the parametrization, the mutilated mesh has been cut into two different mesh partitions and a uniform mesh size field was given to the Delaunay planar mesh algorithm. Error analyses have been performed by calculating with the Metro tool [44] the absolute geometric deviation of the newly created nodes from the non-mutilated mesh of the skull (Hausdorff distance). The L_2 error given in

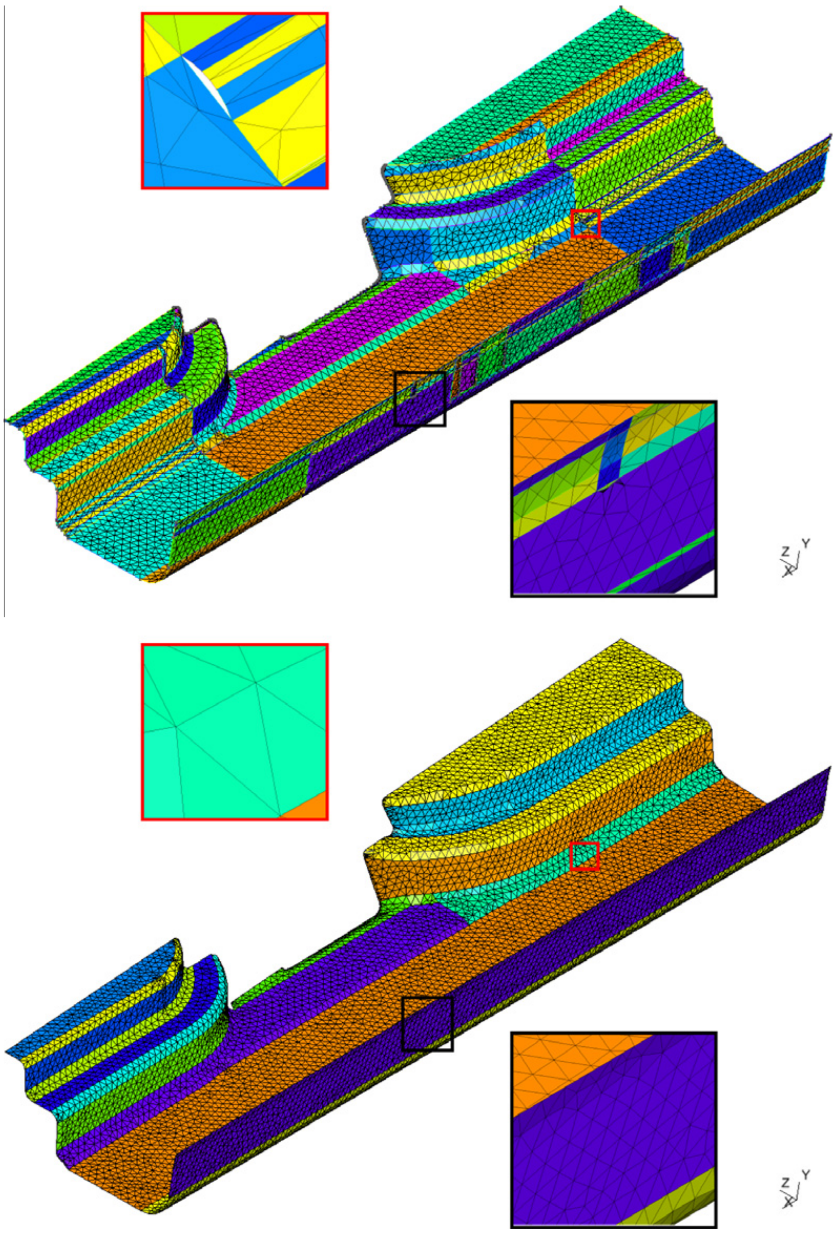

Fig. 7. Example of an initial triangulation of a CAD model from the automotive industry that contains a lot of gaps, overlaps and T-joints. The bottom figure shows the new mesh that is suitable for finite element simulations.

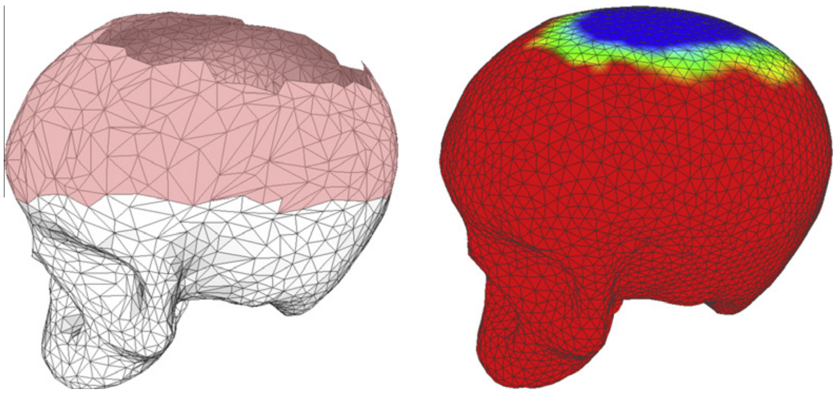

Fig. 8. Mutilated mesh of a skull and remeshed skull with hole filling ($P = 120$ for the inverse map). The color of the right figure indicates the error map. Note that it is a red-green–blue map, so red is min and blue is max error. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 1
Hausdorff distance between the non-mutilated mesh of the skull and the mutilated mesh as a function of the number of P neighboring vertices. Those P neighboring vertices come into play for the computation of the inverse map.

Neighborhood size P	L_2 error
30	6.3e-3
60	2.6e-3
120	2.3e-3

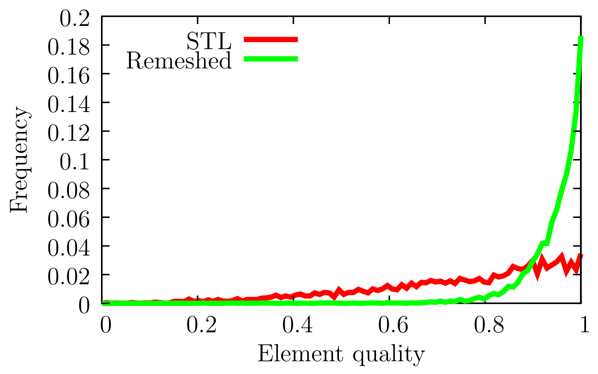

Fig. 9. Remeshing the skull while filling the holes. Quality histogram κ of the initial STL triangulation (3700 triangles) and the new mesh (8740 triangles).

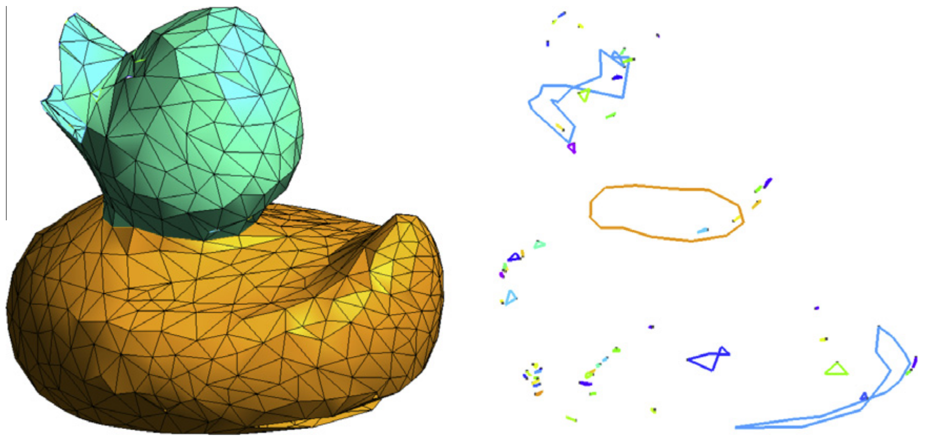

Fig. 10. Raw mesh model of a duck that has been split into two patches for the parametrization (left). The right figure shows the boundary gaps and non-conforming edges of this mesh as well as the orange curve which is the interface between the two surface patches (orange and green). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 1 is normalized by the diagonal of the bounding box. The error is computed for different meshes of the skull that have been computed by changing the number of neighboring vertices P in the computation of the inverse map (see Section 5). The small errors of Table 1 show that the underlying shape is predicted with good geometric fidelity. The presented algorithm could then be used with success in the restoration and reverse engineering of bio-models, and the designing of implants such as titanium plates to restore a damaged skull.

As can be seen for the example of the skull, the remeshing procedure with hole-filling renders also high quality meshes. This is highlighted in the quality histogram of both meshes (initial STL triangulation and new mesh) shown in Fig. 9. The fact that our repair process includes also mesh generation makes our algorithm quite original compared to classical hole-filling algorithms presented in the literature.

The next example shows a raw mesh model of a duck obtained by segmentation. This mesh is made of 1326 triangles and $M = 624$ mesh vertices and has been downloaded from the repository of the AIM@SHAPE project.[9] As can be seen from Fig. 10, the mesh is noisy, self-intersecting, has degenerate and overlapping elements, and is not watertight. The initial mesh contains no less than 57 boundary gaps and non-conforming edges. The parametrization has been computed with RBF's using a reduced

[9] http://www.shapes.aim-at-shape.net/view.php?id=717.

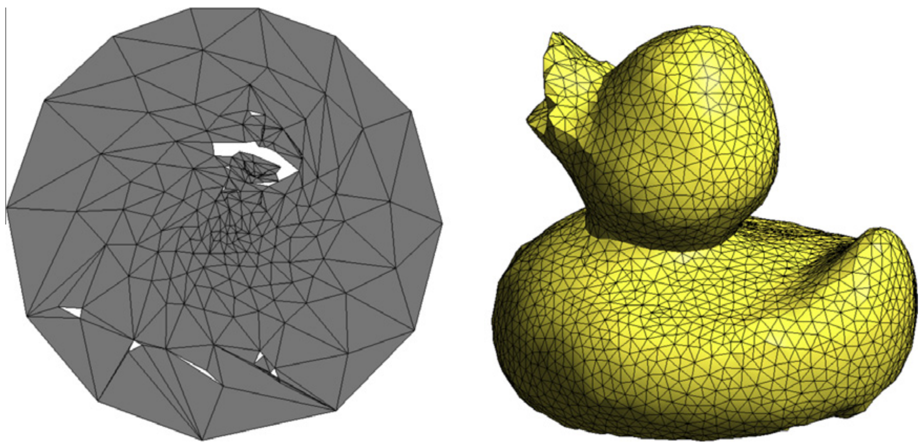

Fig. 11. Parametrization $\mathbf{u}(\mathbf{x})$ of the green surface patch and final mesh made of 5952 triangles. The biggest hole of the parametrization corresponds to the duck's bill and the smaller ones the gaps in the initial triangulation.

number of mesh vertices $m = 443$ obtained by quadratic edge collapse. The parametrization is shown in Fig. 11 (left) for the green patch and the final mesh is shown in Fig. 11 (right). The resulting mesh is made of 5952 triangles and has a very high mean quality of $\bar{\kappa} = 0.923$. The L_2 Hausdorff distance (normalized by the bounding box) between the initial triangulation and the new mesh is only 0.0025. The total time for remeshing is 6.5 s and only 29% of that total time is used for the parametrization with RBF's.

8. Conclusion

In this paper, we presented a brand new approach for repairing and remeshing, which paradoxically finds its strength in the meshless character of the RBF method on which our approach is based. We showed that our algorithm gives excellent results in repairing serious topological and geometrical errors such as holes, reversed normals, overlaps and T-junctions. We also showed in our examples that the presented algorithm is also able to keep some specified topological edges.

The approach makes use of the RBF Orthogonal Gradients method, recently introduced in [2], to solve PDEs on arbitrary surfaces using RBF's. We use this method to solve Laplace's equation (with Dirichlet boundary conditions on a closed boundary curve of the surface) to obtain a discrete parametrization. Next, the surface is remeshed in the parametric space with a computed inverse mapping. We showed that the presented algorithm can be seen as a CAD and mesh repair algorithm but also a quality remeshing algorithm, and a hole filling-algorithm.

Our technique can be further improved by computing conformal mappings instead of harmonic mappings. Indeed, as conformal mappings preserve angles, they will enable us to generate also high quality quadrilateral meshes. Moreover, they require only two Dirichlet boundary conditions for the solution of the mapping. Also different algorithmic advances involving hierarchical and fast-multipole like methods combined with interpolatory filters can be used to reduce the computational cost associated to the RBF interpolation. Meanwhile, we found that this cost is still quite small compared to a user controlled CAD repair followed by a meshing.

Acknowledgements

The rocket reinforcement model of Fig. 7 is provided courtesy of ArcelorMittal and the duck model of Fig. 10 is provided courtesy of IMATI by the AIM@SHAPE Shape Repository.

This work has been partially supported by the Belgian Walloon Region under WIST grants ONELAB 1017086 and DOMHEX 1017074.

References

[1] J.-F. Remacle, C. Geuzaine, G. Compère, E. Marchandise, High quality surface remeshing using harmonic maps, International Journal for Numerical Methods in Engineering 83 (2010) 403–425.
[2] C. Piret, The orthogonal gradients method: a radial basis function solution method for solving partial differential equations on arbitrary surfaces, Journal of Computational Physics, in preparation.
[3] C. Geuzaine, J.-F. Remacle, Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities, International Journal for Numerical Methods in Engineering 79 (11) (2009) 1309–1331.
[4] G. Butlin, C. Stops, CAD data repair, in: Proceedings of the Fifth International Meshing Roundtable, Pennsylvania, U.S.A., (1996) 7–12.
[5] M. Jones, M. Price, G. Butlin, Geometry management support for auto-meshing, in: Proceedings Fourth International Meshing Roundtable, (1995) 153–164.
[6] J.P. Steinbrenner, N.J. Wyman, J.R. Chawner, Procedural CAD model edge tolerance negotiation for surface meshing, Engineering with Computers 17 (2001) 315–325.

[7] N.A. Peterson, K. Kyle, Detecting translation errors in CAD surfaces and preparing geometries for mesh generation, Tech. Rep. UCRL-JC-144019., Center for Applied Scientific Computing, Lawrence Livermore National Labs, Livermore, CA 94551 (2001).

[8] X. Sheng, I. Meier, Generating topological structures for surface models, IEEE Computer Graphics and Applications 15 (6) (1995) 35–41.

[9] G. Barequet, M. Sharir, Filling gaps in the boundary of a polyhedron, Computer Aided Geometric Design 12 (1995) 207–229.

[10] G. Barequet, Using geometric hashing to repair CAD objects, IEEE Computational Science and Engineering (1997) 22–28.

[11] S. Morvan, G. Fadel, IVES: An interactive virtual environment for the correction of.STL files, in: Proceedings of Conference on Virtual Design, University of California, Irvine, (1996).

[12] F. Nourrudin, G. Turk, Simplification and repair of polygonal models using volumetric techniques, IEEE Transactions on Visualisation and Computer Graphics 9 (2) (2003) 191–205.

[13] X. Wu, M. Wang, B. Han, An automatic hole-filling algorithm for polygonal meshes, Computer-Aided Design and Applications 5 (6) (2008) 889–899.

[14] J. Branch, F. Prieto, P. Boulanger, A hole-filling algorithm for triangular meshes using local Radial Basis Function, in: Springer-Verlag (Ed.), Fifteenth International Meshing Roundtable, (2006) 411–432.

[15] Y. Zheng, N. Weatherill, O. Hassan, Topology abstraction of surface models for three-dimensional grid generation, Engineering with Computers 17 (2001) 28–38.

[16] D.L. Marcum, Efficient generation of high-quality unstructured surface and volume grids, Engineering with Computers 17 (2001) 211–233.

[17] B. Levy, S. Petitjean, N. Ray, J. Maillot, Least squares conformal maps for automatic texture atlas generation, in: Computer Graphics (Proceedings of SIGGRAPH 02), (2002) 362 – 371.

[18] C. Bennis, J.-M. Vézien, G. Iglésias, Piecewise surface flattening for non-distorted texture mapping, ACM SIGGRAPH Computer Graphics (1991) 237–246.

[19] J. Maillot, H. Yahia, A. Verroust, Interactive texture mapping, in: Proceedings of ACM SIGGRAPH'93, (1993) 27–34.

[20] A. Sheffer, E. Praun, K. Rose, Mesh parameterization methods and their applications, Foundation and Trends in Computational Graphical Visualisation 2 (2) (2006) 105–171.

[21] M.S. Floater, K. Hormann, Surface Parameterization: A Tutorial and Survey, in: In Advances in Multiresolution for Geometric Modelling, Springer, 2005, pp. 157–186.

[22] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, W. Stuetzle, Multiresolution analysis of arbitrary meshes, in: SIGGRAPH '95: Proceedings of the Twentysecond Annual Conference on Computer graphics and interactive techniques, (1995) 173–182.

[23] D.L. Marcum, A. Gaither, Unstructured surface grid generation using global mapping and physical space approximation, in: Proceedings, Eighth International Meshing Roundtable, (1999) 397–406.

[24] E. Marchandise, G. Compère, M. Willemet, G. Bricteux, C. Geuzaine, J.-F. Remacle, Quality meshing based on STL triangulations for biomedical simulations, International Journal for Numerical Methods in Biomedical Engineering 83 (2010) 876–889.

[25] E. Marchandise, C. Carton de Wiart, W. Vos, C. Geuzaine, J.-F. Remacle, High quality surface remeshing using harmonic maps. Part II: Surfaces with high genus and of large aspect ratio, International Journal for Numerical Methods in Engineering 86 (11) (2011) 1303–1321.

[26] E. Kansa, Multiquadrics, a scattered data approximation scheme with applications to computational fluid-dynamics (i): Surface approximations and partial derivative estimates, Computers and Mathematics with Applications 19 (1990) 127–145.

[27] E. Kansa, Multiquadrics: a scattered data approximation scheme with applications to computational fluid-dynamics (ii): Solutions to parabolic, hyperbolic and elliptic Partial Differential Equations, Computers and Mathematics with Applications 19 (1990) 147–161.

[28] G. Fasshauer, Meshfree approximation methods with MATLAB, Interdisciplinary mathematical sciences, World Scientific, (2007). URL<http://www.books.google.com/books?id=gtqBdMEqryEC>.

[29] N. Flyer, B. Fornberg, Radial Basis Functions: Developments and applications to planetary scale flows, Computers and Fluids 46 (2011) 23–32.

[30] N. Flyer, G.B. Wright, A Radial Basis Function method for the shallow water equations on a sphere, Proceedings of the Royal Society 465 (2009) 1949–1976.

[31] N. Flyer, G.B. Wright, Transport schemes on a sphere using Radial Basis Functions, Journal of Computational Physics 226 (2007) 1059–1084.

[32] G.B. Wright, N. Flyer, D. Yuen, A hybrid Radial Basis Function - pseudo-spectral method for thermal convection in a 3d spherical shell, Geochemistry Geophysics Geosystems 11 (7) (2010) 1–18.

[33] J. Schmidt, C. Piret, N. Zhang, B. Kadlec, et al, Modeling of tsunami waves and atmospheric swirling flows with Graphics Processing Unit (GPU) and Radial Basis Functions (RBF), Concurrency and Computation.: Practice and Experience 22 (2010) 1813–1835.

[34] J. Carr, R. Beatson, B. McCallum, W. Fright, et al., Smooth surface reconstruction from noisy range data, in: GRAPHITE'03, (2003) 119–126.

[35] R. Beatson, J. Cherrie, T. McLennan, et al, Surface reconstruction via smoothest restricted range approximation, Geometric Modeling and Computing 46 (2004) 41–52.

[36] B. Fornberg, C. Piret, On choosing a Radial Basis Function and a shape parameter when solving a convective PDE on a sphere, Journal of Computational Physics 227 (2008) 2758–2780.

[37] C.B. Macdonald, S.J. Ruuth, The implicit closest point method for the numerical solution of Partial Differential Equations on surfaces, SIAM Journal on Scientific Computing 31 (2009) 4330–4350.

[38] B. Fornberg, T. Driscoll, G. Wright, R. Charles, Observations on the behavior of Radial Basis Function approximations near boundaries, Computers & Mathematics with Applications 43 (3-5) (2002) 473–490.

[39] B. Fornberg, G. Wright, Stable computation of multiquadric interpolants for all values of the shape parameter, Computers and Mathematics with Applications 48 (2004) 853–867.

[40] B. Fornberg, C. Piret, A stable algorithm for flat Radial Basis Functions on a sphere, SIAM Journal of Scientific Computing 30 (1) (2007) 60–80.

[41] B. Fornberg, E. Larsson, N. Flyer, Stable computations with gaussian Radial Basis Functions, SIAM Journal of Scientific Computing 33 (2) (2011) 869–892.

[42] J. Cherrie, R. Beatson, G. Newsam, Fast evaluation of Radial Basis Functions: Methods for generalized multiquadrics in rn, SIAM Journal of Scientific Computing 23 (5) (2002) 1549–1571.

[43] M. Garland, P. Heckbert, Surface simplification using quadric error metrics., in: In SIGGRAPH '97: Proceedings of the Twentyfourth Annual Conference on Computer Graphics and Interactive Techniques, ACM Press/Addison-Wesley Publishing Co, New York, NY, USA, (1997) 209–216.

[44] P. Cignoni, C. Rocchini, R. Scopigno, Metro: Measuring error on simplified surfaces, Computer Graphics Forum 17 (2) (1998) 167–174.