Mesh and CAD repair based on parametrizations with Radial Basis Functions

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

Université catholique de Louvain, Institute of Mechanics, Materials and Civil Engineering (iMMC), Place du Levant 1, 1348 Louvain-la-Neuve, Belgium
cecile.piret@uclouvain.be, jean-francois.remacle@uclouvain.be, emilie.marchandise@uclouvain.be

Summary. The goal of this paper is to present a new repair process that includes both model/mesh repair and mesh generation. The repair algorithm is based on an initial mesh of the CAD that may contain topological and geometrical errors. This initial mesh is then remeshed by computing a discrete parametrization with radial basis functions (RBF's).

[34] showed that a discrete parametrization can be computed by solving PDE's on an initial correct triangulation 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.

In this work, we implement the Orthogonal Gradients method which was recently introduced in [32], as a technique to solve PDE's on arbitrary surfaces with RBF's. We will implement the low order version of the algorithm, which already gives great results in this context.

Different examples show that the presented method is able to deal with errors such as gaps, overlaps, T-joints and simple holes 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 [18].

Key words: geometry processing; hole filling algorithm; radial basis functions; RBF; surface remeshing; surface parametrization; STL file format; surface mapping; harmonic map; Orthogonal Gradients method

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 [1] and have undesirable holes. Figure 1 gives an example of such dirty CAD models or STL triangulations that need to be repaired.

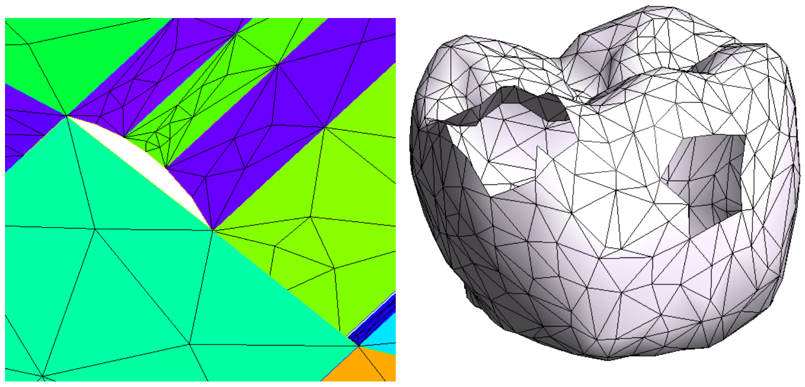


Fig. 1. Two examples for which a cad and mesh repair algorithm is needed. The figure on the left shows an initial triangulation of a dirty CAD model with topological errors: gaps (holes), overlaps and T-junctions. The right figure displays an STL triangulation of a tooth that contains undesirable holes.

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 [5, 19, 38]. Those approaches thus provide specific tools for model correction that are controlled by the user [31, 37]. 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.

Another approach is that of correcting an initial triangulation of the model through the addition of triangles and different stitching procedures [2, 1, 29]. In the same vein, Nooruddin and Turk [30] proposed a method to repair polygonal meshes using volumetric techniques. Unfortunately, those algorithms do not consider geometric intersections and inconsistent topology may be present. Also based on an initial triangulation, Wu et al. [40] suggested some specific hole-filling algorithms that employ RBF's as an interpolation technique to

[1] 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.

construct an implicit surface patch and to intersect this implicit surface with the existing triangulation. However, the intersection method to reconstruct the mesh is quite complex.

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 an discrete parametrization. Surface parametrization techniques [41, 27, 22] originate mainly from the computer graphics community: they have been used extensively for applying textures onto surfaces [4, 24] and have become a very useful and efficient tool for many mesh processing applications [22, 36, 9]. In the context of remeshing procedures, the initial surface is parametrized onto a surface in $\mathcal{R}^2$, the surface is meshed using any standard planar mesh generation procedure and the new triangulation is then mapped back to the original surface [7, 28]. We showed in [34, 26, 25] that harmonic maps can be computed efficiently by solving partial differential equations (PDE's) on the initial triangulation with finite elements.

In the context of CAD and mesh repair, the initial triangulation may contain topological errors such as holes, T-junctions and overlaps that make numerical techniques such as finite elements fail. The meshless character of the RBF's makes it then an attractive numerical method 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 [20, 21]. Its high accuracy and meshless character have made it the method of choice for problems set on complicated geometries [8]. 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 [10, 12, 11, 39, 35]. Although a good part of the RBF literature deals with surface reconstruction [6, 3], no technique has been developed to solve PDE's on them, until [32], 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 [32] 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 RBF's Orthogonal Gradients method centered on the mesh vertices of the initial triangulation. Next, we explain in section 4 how to find the inverse mapping in order to be able to project the new points on the 3D surface (CAD patches or discrete 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 $\mathcal{R}^2$:

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

In the remainder of this paper, we restrict ourselves to the parametrization of non-closed surfaces meshes of zero genus. If the surface mesh is closed, of genus greater that zero, or non-manifold, we partition the initial mesh into different mesh patches [26, 25] and compute a discrete parametrization for each of those mesh patches with the presented method.

In this work, we have chosen a harmonic mapping for the parametrization [28, 7, 34]. Harmonic maps $\mathbf{u}(\mathbf{x})$ can be computed by solving two Laplace's equations on the 3D surface Γ:

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

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

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

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 T.

We will show in the next section how to compute the discrete parametrization using RBF's. Figure 2 shows a triangulation of a CAD model with topological and geometrical errors such as gaps, overlaps and T-junctions. Those gaps presented in the initial mesh are magnified during the parametrization procedure. The T-junctions are also visible in both the 3D space and the parametric space. With the presented approach, there is no need to identify those T-junctions and gaps. The mesh points of the dirty initial triangulation are directly used to compute a discrete parametrization $\mathbf{u}(\mathbf{x})$ with RBF's. The remeshing procedure will then be performed in the parametric space given this mapping $\mathbf{u}(\mathbf{x})$ and the parametric description of the 4 CAD patches $\mathbf{u}_j^{CAD}$ visible on the left figure 2. For the remeshing of the holes, RBF interpolations will be used.

3 Harmonic map with Radial Basis functions

In the case where the initial triangulation contains topological and geometrical errors (such as in Fig. 2), standard numerical methods based on meshes such as finite elements, or finite volumes fail to compute the solutions to Laplace's

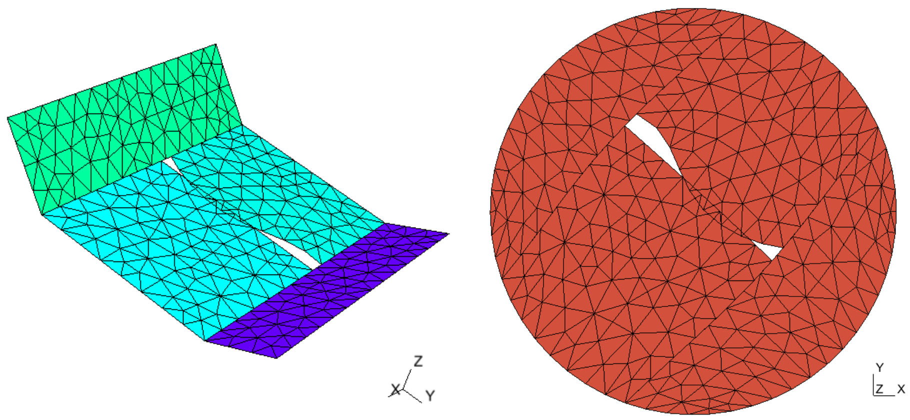


Fig. 2. Initial triangulation of a CAD model with topological and geometrical errors such as gaps, overlaps and T-junctions (left) and its discrete parametrization $\mathbf{u}(\mathbf{x})$ computed with RBF's (right). One can see clearly the T-junctions, holes and overlaps of the initial mesh in the parametric space.

Name of RBF	**Abbreviation**	**Definition**
Multiquadric	MQ	$\sqrt{1+(\varepsilon r)^2}$
Inverse multiquadric	IMQ	$\frac{1}{\sqrt{1+(\varepsilon r)^2}}$
Inverse quadratic	IQ	$\frac{1}{1+(\varepsilon r)^2}$
Gaussian	GA	$e^{-(\varepsilon r)^2}$

Table 1. Definitions of the most commonly used C_∞ radial functions.

equation (2). The meshless character of the RBF's makes it then an attractive numerical method for solving the two Laplace's equations (2) whose solution is the harmonic map.

Different techniques have been recently suggested to solve PDE's on arbitrary surfaces with RBF's [32]. In this work, we use [32]'s simplest version of the Orthogonal Gradients method, which relies on a level set representation of the initial surface. Let M be the number of points on the initial triangulation (points on the surface Γ in Figures 3 and 4).

In this work, we use the MQ (Table 1) radial function $\phi(r)$ to represent the solutions (u, v) of the Laplace's equations on Γ.[2]

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

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

where $|| \cdot ||$ denotes the Euclidean norm, $\mathbf{x}_i$ are the set of N data points, λ_i are the RBF's expansion coefficients, and ϵ is the shape parameter. All C_∞ smooth radial functions yield a similar accuracy, and any of them could be used.

Let us find a matrix D that discretizes, via an RBF representation, a continuous differential operator L (for the Laplacian operator, we have $L = \mathcal{L} = \partial_{xx} + \partial_{yy} + \partial_{zz}$). We can rewrite the RBF interpolation (4) in matrix form for the N points:

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

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

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

where u^L is the value of the differential operator applied to u at each $\mathbf{x}_k$. Thus we have $B^L \varLambda = U^L$ in matrix form, where matrix B is of size $K \times N$, and K is the number of points on which we wish to compute the differentiation. Eliminating the expansion coefficient vector $\varLambda$ leads to:

$$U^L = \underbrace{B^L A^{-1}}_{D^L} U. \tag{7}$$

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 of a function defined on the 3D surface $\varGamma$ by using the Orthogonal Gradients method of Piret [32]. This method is loosely inspired from the closest point method [23] in that the goal is to make that function constant in the direction normal to the surface $\varGamma$. 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.

In order to have a reliable RBF representation of the surface $\varGamma$ and to reliably compute the direction that is normal to the surface, it is now quite usual to introduce additional points, on either side of the surface, and to define the level set distance function as $s(\mathbf{x}) = 0$ on $\varGamma$, $s(\mathbf{x}) = \pm 1$ on $\varGamma^{\pm}$, where $\varGamma^{\pm}$ are surfaces that surround and that are parallel to $\varGamma$ (as in Figures 3 and 4) (see for example [3, 8]) The alternative to introducing additional points is to define a level surface $s(\mathbf{x}) = c$ ($c \neq 0$ to avoid the trivial RBF expansion) on Γ through the original nodes only. However, it is known that RBFs do a poor job of interpolating constant values [16]. The interpolated level-set would risk to be self-intersecting, which would invalidate any result. Adding nodes on either sides of Γ makes the distance function interpolation much more stable. We thus extend the original M points of the surface inward and outward as follows:

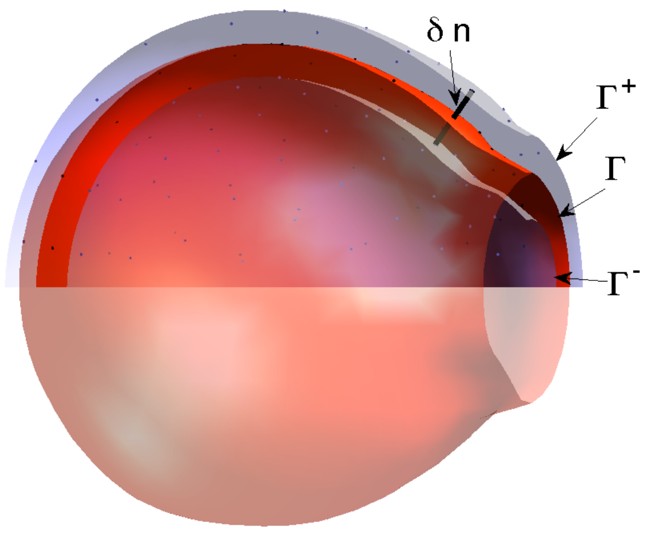


Fig. 3. Illustration of the RBF Orthogonal Gradients method. M points are uniformly distributed on the main surface Γ (in red). They are the mesh vertices of the initial triangulation. At each point, the normal $\mathbf{n}$ to the surface is computed and 2 new points are obtained at distance δ from the surface, one on either side of Γ. One can see these $2M$ additional points on the two grey layers (which we call Γ^+ and Γ^-) that surround Γ.

A schema of the RBF Orthogonal Gradients method is displayed in Figure 4 . M points (full red dots) are uniformly distributed on Γ. At each point, the normal $\mathbf{n}$ to the surface is computed and two new points are obtained

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

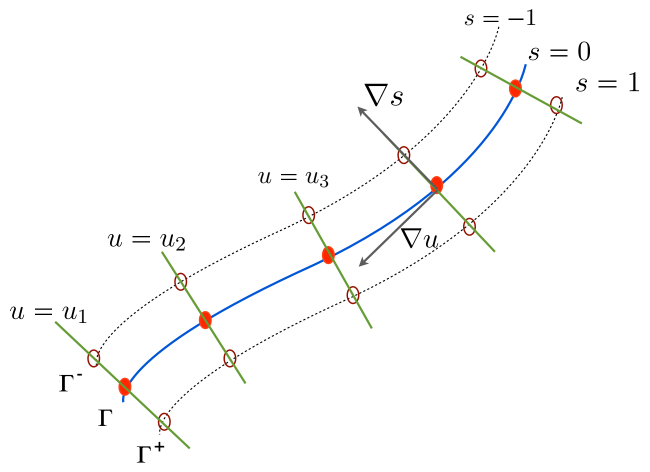


Fig. 4. Schema of the RBF Orthogonal Gradients method.

at distance δ from the surface, one on either side of Γ (empty red dots). We 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 $u(\mathbf{x})$. We set $s(\mathbf{x}) = 0$ on Γ, $s(\mathbf{x}) = \pm 1$ on $\Gamma^{\pm}$, thus ∇s points in the normal direction to Γ. Moreover, as we let $u(\mathbf{x_i}) = u_i$ on Γ, we set $u(\mathbf{x_i^{\pm}}) = u_i$, where $\mathbf{x_i^{\pm}}$ are the points corresponding to $\mathbf{x_i}$ on $\Gamma^{\pm}$, making $u(\mathbf{x})$ constant in the direction that is normal to Γ. This guarantees that, when we apply the Laplacian to $u(\mathbf{x})$, the normal component of the laplacian will vanish, and only the surface laplacian will remain.

We have thus a set of $N = 3\,M$ points that we use to compute the differentiation matrix for the Laplacian $D^{\mathcal{L}}$. This differentiation matrix is of size $M \times 3M$. In order to restrict the Laplacian operator to the surface Γ, we have to cancel the derivative of u in the direction normal to the surface:

$$\nabla_{\mathbf{n}} u = 0, \quad \nabla_{\mathbf{n}} \cdot \nabla_{\mathbf{n}} u = 0 \tag{10}$$

so that we only keep the restriction of the Laplacian on the surface Γ.

$$\nabla^2 u = (\nabla_{\mathbf{n}} + \nabla_{\mathbf{t}}) \cdot (\nabla_{\mathbf{n}} + \nabla_{\mathbf{t}}) u = \nabla_{\mathbf{t}} \cdot \nabla_{\mathbf{t}} u, \tag{11}$$

In (11), n and t denote respectively the surface normal and tangential directions.

There are two versions of the RBF Orthogonal Gradients method. The first version is less accurate but very easy to implement. This method assigns a constant value of $u(\mathbf{x})$ in the normal direction to the surface, i.e at the inward and outward points of $\mathbf{x}_i$, i.e $u(\mathbf{x}_i) = u(\mathbf{x}_i^e)$.

If the initial approximation of the normals (9) (which we use to compute the additional $2M$ points, thus before the level set distance function has been defined) is not accurate enough, the extended points $\mathbf{x}_i^e$ need to be corrected in order to guarantee the cancelation of the derivatives in the normal direction. The orthogonality condition is then $u(\mathbf{x}_i) = u(\mathbf{x}_i^{ec})$, where $u(\mathbf{x}_i^{ec})$ can be computed from (7) as $U^{ec} = D^I U$, so that we come up with the following linear system of size $3M \times 3M$ to solve:

$$\begin{pmatrix} D^{\mathcal{L}} \\ I_\Gamma - D^I \\ I_\Gamma - D^I \end{pmatrix} U = \bar{0}, \tag{12}$$

where I_Γ is an $M \times 3M$ matrix whose entries are zero everywhere except for the entries (i, i) (corresponding to points $\mathbf{x}_i$ on Γ) that have a value of 1.

The second version of the RBF Orthogonal Gradients method is more accurate, and we will implement it as part of our future work. Dirichlet boundary conditions (3) on the boundary nodes of $\mathbf{x}_i$ are then applied directly on the linear system (12) 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). This algorithm has a global and a local version. In the global version, the derivatives are computed using all the nodes, while in the local version, the derivatives at a particular point are computed using only the points in a cluster of neighbors. Although the local method loses the potential for spectral accuracy in the solution representations, it still is a high order method and it has a significantly reduced complexity compared to the global method, since the differentiation matrices are now sparse.

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 (8). 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 [8, 17]. Unless one uses one of the algorithms that bypass the small shape parameter conditioning issue [15, 14, 13], the shape parameter will need to be set large enough to have 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 $\epsilon = \frac{\epsilon^*}{d_{\min}}$. 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. However, the accuracy seems less sensitive to its value than to the value of the shape parameter. 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 $\epsilon^* = M/300$. If M were to be much larger, ϵ^* would need to grow exponentially with M.

4 Inverse mapping

Once the harmonic map has been computed, it can be used as input for planar surface meshers (delaunay, frontal, etc.) to produce high quality triangulations [34]. It is this remeshing step performed in the parametric space that will remove the overlaps, intersections or T-junctions. The only information we need to provide to the planar meshers is the inverse map $\mathbf{x}(\mathbf{u})$ and the Jacobians of the mapping $\mathbf{x}_u$ and $\mathbf{x}_v$. The jacobians are then used to build the metric tensor (or first fundamental form)

$$\mathbf{M} = \mathbf{x}_{,\mathbf{u}}^T \mathbf{x}_{,\mathbf{u}} = \begin{bmatrix} \mathbf{x}_{,u} \cdot \mathbf{x}_{,u} & \mathbf{x}_{,u} \cdot \mathbf{x}_{,v} \\ \mathbf{x}_{,v} \cdot \mathbf{x}_{,u} & \mathbf{x}_{,v} \cdot \mathbf{x}_{,v} \end{bmatrix} \tag{13}$$

in order to compute edge lengths, angles and areas.

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.

Figure 5 shows the remeshing procedure in the parametric space as well as the resulting mesh for the CAD model of Fig. 2. As can be seen from Fig. 5, the presented algorithm is also able to keep some specified shape features such as topological edges.

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, then the inverse mapping can be found as follows [34].

1. Find a triangle T_j of the parametric space Γ' that contains point $\mathbf{u}$; Note that in the case the parametrization contains overlaps or intersections that come from the dirty model (see right Fig. 2), 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 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 (7):

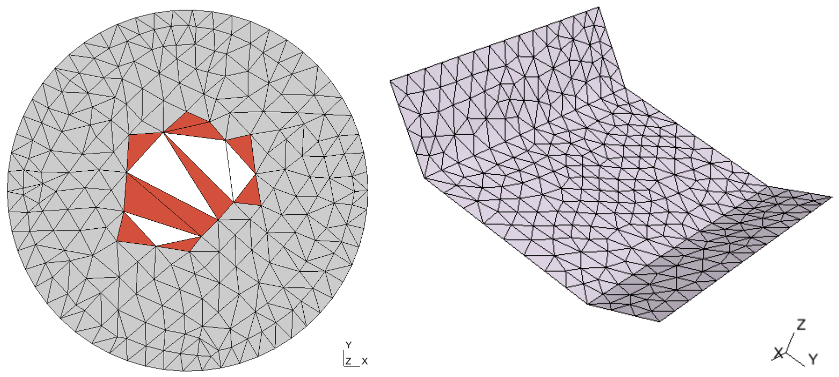


Fig. 5. Starting from a initial triangulation of a CAD model with topological and geometrical errors and from its parametrization (see the Fig. 2), we compute with a planar mesh generator the new mesh. The left figure shows an intermediate step of the frontal mesh algorithm (the grey triangles are the resolved layers, the red the active layers and the white the waiting layers) in the parametric space and the right figure shows the resulting mesh in the 3D space.

$$X^L = \underbrace{B^L_{\mathbf{u}} A^{-1}_{\mathbf{u}}}_{D^L_{\mathbf{u}}} X, \tag{14}$$

where X is taken as the vector of the closest P points $\mathbf{x}_i$. Those closest points are found by computing the P closest[4] parametrized points $\mathbf{u}_i$ to $\mathbf{u}_p$ in the set of the M parametrized points and by taking the corresponding points in the 3D space $\mathbf{x}_i$. In 14, X^L is the inverse mapping we are looking for (of size $K = 1$) and $D^L_{\mathbf{u}}$ is the matrix of differentiation in the parametric space of size $K \times P$ (i.e. with the radial basis functions written in the parametric space: $\phi(\mathbf{u} - \mathbf{u}_i)$). From (14), we obtain easily the inverse quantities needed by the planar meshes. Indeed, when $L = I$, we get the inverse map $\mathbf{x}_p(\mathbf{x}_u)$, when $L = \partial_u$, we obtain the Jacobian $\mathbf{x}_{u,p}$ and when $L = \partial_v$, we have the Jacobian $\mathbf{x}_{v,p}$.

From Fig. 5, 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.

5 Results

Two examples aim to show that the presented algorithm can be seen as a CAD and mesh repair algorithm.

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

The first example shows a raw mesh model of a face (Fig. 6). The triangulation is made of 1048 triangles, contains different self-intersection triangles (see red triangles in Fig. 6) and has two undesirable holes. It follows that the parametrization cannot be computed with a finite element Laplacian. The parametrization computed with RBF's is shown in Fig. 7. A Frontal-Delaunay algorithm [33] was used for the remeshing in the parametric space. We first compare the quality of the initial and final mesh by computing for every triangle the aspect ratio κ that is the ratio between the inscribed and circumscribed radius of the triangle [18, 34]. Parameter κ is such that an equilateral triangle has $\kappa = 1$ and a degenerated flat triangle has $\kappa = 0$. The initial triangulation of the face has $\bar{\kappa} = 0.69$ and $\kappa_{min} = 0.01$, while the new mesh made of 2705 triangles has $\bar{\kappa} = 0.94$ and $\kappa_{min} = 0.36$. The L_2 Hausdorff distance (normalized by the bounding box) between the initial triangulation and the new mesh is only 0.0007. This small error show that the underlying shape is predicted with good geometric fidelity.The total time for remeshing the face is $1s^5$, and 25% of that time is for computing the parametrization with RBFs.

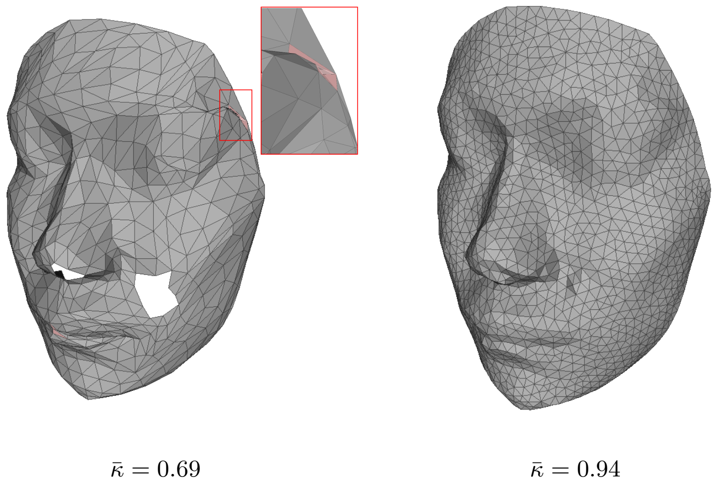


Fig. 6. Dirty raw mesh model of the face (left) and the new mesh (right) and the average quality $\bar{\kappa}$ of the meshes. The red triangles on the left are the self-intersecting triangles.

In the next 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

[5] on a MACBOOK PRO clocked at 2.4 GHz

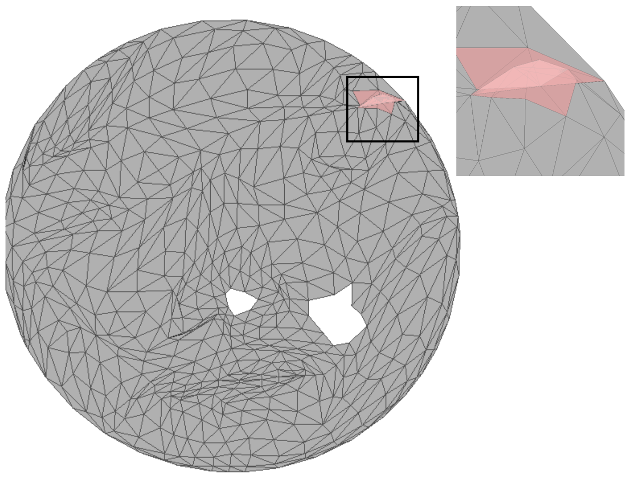


Fig. 7. The parametrized mesh computed with RBFs. The red triangles are the self-intersecting triangles.

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. 8, 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. 8 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.

Seventeen 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. 8 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 $\bar{\kappa} = 0.94$. The total time for meshing the compounds is $30s$. This example also demonstrates that our method works with non-uniformly distributed nodes. The minimum distance between two nodes is $1.e - 3$ while the maximum distance is 5.2.

6 Conclusion

In this paper, we presented a brand new approach for repairing and generating meshes, 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, non-manifold vertices.

The approach makes use of the RBF Orthogonal Gradients method, recently introduced in [32], 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 global quality remeshing algorithm, and a hole filling-algorithm.

Our technique can be further improved by using the higher order version of the RBF Orthogonal Gradients method, or 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.

Acknowledgements

The rocket reinforcement model of Fig. 8 is provided courtesy of ArcelorMittal and the two raw mesh models are provided courtesy of IMATI by the AIM@SHAPE Shape Repository [6].

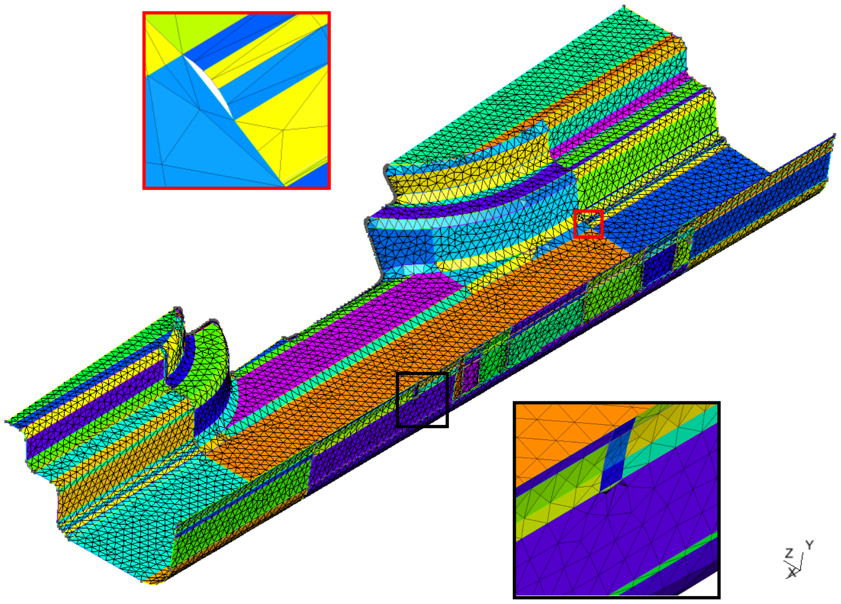
References

1. G. Barequet. Using geometric hashing to repair cad objects. *IEEE Comput. Science and Engineering*, pages 22–28, 1997.
2. G. Barequet and M. Sharir. Filling gaps in the boundary of a polyhedron. *Computer Aided Geom. Design*, 12:207–229, 1995.
3. R.K. Beatson, J.B. Cherrie, and T.J. et al. McLennan. Surface reconstruction via smoothest restricted range approximation. *Geometric modeling and computing*, 46:41–52, 2004.
4. C. Bennis, J-M Vézien, and G. Iglésias. Piecewise surface flattening for non-distorted texture mapping. *ACM SIGGRAPH Computer Graphics*, pages 237 – 246, 1991.
5. G Butlin and C. Stops. Cad data repair. In *Proceedings of the 5th International Meshing Roundtable*, pages 7–12, Pennsylvania, U.S.A., 1996.
6. J.C. Carr, R.K. Beatson, and B.C. et al McCallum. Smooth surface reconstruction from noisy range data. In *GRAPHITE'03*, pages 119–126, 2003.
7. Matthias Eck, Tony DeRose, Tom Duchamp, Hugues Hoppe, Michael Lounsbery, and Werner Stuetzle. Multiresolution analysis of arbitrary meshes. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 173–182, 1995.
8. G.E. Fasshauer. *Meshfree approximation methods with MATLAB*. Interdisciplinary mathematical sciences. World Scientific, 2007.
9. M. S. Floater and K. Hormann. Surface parameterization: a tutorial and survey. In *In Advances in Multiresolution for Geometric Modelling*, pages 157–186. Springer, 2005.
10. N. Flyer and B. Fornberg. Radial basis functions: Developments and applications to planetary scale flows. *Computers and Fluids*, 46:23–32, 2011.
11. N. Flyer and G.-B. Wright. Transport schemes on a sphere using radial basis functions. *J. Comp. Phys.*, 226:1059–1084, 2007.
12. N. Flyer and G.-B. Wright. A radial basis function method for the shallow water equations on a sphere. *Proc. Roy. Soc.*, 465:1949–1976, 2009.
13. B. Fornberg, E. Larsson, and N. Flyer. Stable computations with gaussian radial basis functions. *SIAM J. Scientific Computing*, 33(2):869–892, 2011.
14. B. Fornberg and C. Piret. A stable algorithm for flat radial basis functions on a sphere. *SIAM J. Scientific Computing*, 30(1):60–80, 2007.
15. B. Fornberg and C. Piret. On choosing a radial basis function and a shape parameter when solving a convective pde on a sphere. *Journal of Computational Physics*, 227:2758–2780, 2008.

[6] `http://shapes.aim-at-shape.net/`

16. B. Fornberg, G. Wright, and Larsson E. Some observations regarding interpolants in the limit of flat radial basis functions. *Computers and Mathematics with Applications*, 47:37–55, 2004.
17. B. Fornberg and G.-B. Wright. Stable computation of multiquadric interpolants for all values of the shape parameter. *Comp. Math. Applic. 2004;*, 48:853–867, 2004.
18. C. Geuzaine and 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):1309–1331, 2009.
19. M.R. Jones, M.A. Price, and G. Butlin. Geometry management support for auto-meshing. In *Proceedings, 4th International Meshing Roundtable*, pages 153–164, 1995.
20. E.J. Kansa. Multiquadrics, a scattered data approximation scheme with applications to computational fluid-dynamics (i): Surface approximations and partial derivative estimates. *Comput. Math. Appl.*, 19:127–145, 1990.
21. E.J. Kansa. Multiquadrics: a scattered data approximation scheme with applications to computational fluid-dynamics (ii): Solutions to parabolic, hyperbolic and elliptic partial differential equations. *Comput. Math. Appl.*, 19:147–161, 1990.
22. B. Levy, S. Petitjean, N. Ray, and J. Maillot. Least squares conformal maps for automatic texture atlas generation. In *Computer Graphics (Proceedings of SIGGRAPH 02)*, pages 362 – 371, 2002.
23. Colin B. Macdonald and Steven J. Ruuth. The implicit closest point method for the numerical solution of partial differential equations on surfaces. *Siam Journal on Scientific Computing*, 31:4330–4350, 2009.
24. J. Maillot, H. Yahia, and A. Verroust. Interactive texture mapping. In *Proceedings of ACM SIGGRAPH'93*, pages 27–34, 1993.
25. E. Marchandise, C. Carton de Wiart, W.G. Vos, C. Geuzaine, and 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):1303–1321, June 2011.
26. E. Marchandise, G. Compère, M. Willemet, G. Bricteux, C. Geuzaine, and J-F. Remacle. Quality meshing based on stl triangulations for biomedical simulations. *International Journal for Numerical Methods in Biomedical Engineering*, 83:876–889, 2010.
27. David L. Marcum. Efficient generation of high-quality unstructured surface and volume grids. *Engineering with Computers*, 17:211–233, 2001.
28. David L. Marcum and Adam Gaither. Unstructured surface grid generation using global mapping and physical space approximation. In *Proceedings, 8th International Meshing Roundtable*, pages 397–406, 1999.
29. S.M. Morvan and G.M. Fadel. Ives: an interactive virtual environment for the correction of .stl files. In *Proceedings of Conference on Virtual Design*, University of California, Irvine, 1996.
30. F.S Nourrudin and G. Turk. Simplification and repair of polygonal models using volumetric techniques. *IEEE transactions on Visualisation and Computer Graphics*, 9(2):191–205, 2003.
31. N. A. Peterson and K. Kyle. Detecting translation errors in cad surfaces and preparing geometries for mesh generation,. Technical Report UCRL-JC-144019., Center for Applied Scientific Computing, Lawrence Livermore National Labs,, Livermore, CA 94551, 2001.

32. C. Piret. The orthogonal gradients method: a radial basis function solution method for solving partial differential equations on arbitrary surfaces. *Journal of Computational Physics*, page in preparation, 2011.
33. S. Rebay. Efficient unstructured mesh generation by means of delaunay triangulation and bowyer-watson algorithm. *Journal of Computational Physics*, 106:25–138, 1993.
34. J-F. Remacle, C. Geuzaine, G. Compère, and E. Marchandise. High quality surface remeshing using harmonic maps. *International Journal for Numerical Methods in Engineering*, 83:403–425, 2010.
35. J. Schmidt, C. Piret, and N. et al. Zhang. Modeling of tsunami waves and atmospheric swirling flows with graphics processing unit (gpu) and radial basis functions (rbf). *Concurrency Comput.: Pract. Exp.*, 22:1813–1835, 2010.
36. A. Sheffer, E. Praun, and K. Rose. Mesh parameterization methods and their applications. *Found. Trends. Comput. Graph. Vis.*, 2(2):105–171, 2006.
37. X. Sheng and I.R. Meier. Generating topological structures for surface models,. *IEEE Comput. Graphics Appl.*, 15(6):35–41, 1995.
38. J. P. Steinbrenner, N. J. Wyman, and J. R. Chawner. Procedural cad model edge tolerance negotiation for surface meshing. *Engineering with Computers*, 17:315–325, 2001.
39. G.-B. Wright, N. Flyer, and D.A. Yuen. A hybrid radial basis function - pseudospectral method for thermal convection in a 3d spherical shell. *Geochemistry Geophysics Geosystems*, 11(7):1–18, 2010.
40. X.J. Wu, M.Y. Wang, and B Han. An automatic hole-filling algorithm for polygonal meshes. *Computer-Aided Design and Applications*, 5(6):889–899, 2008.
41. Y. Zheng, N.P. Weatherill, and O. Hassan. Topology abstraction of surface models for three-dimensional grid generation. *Engineering with Computers*, 17:28–38, 2001.



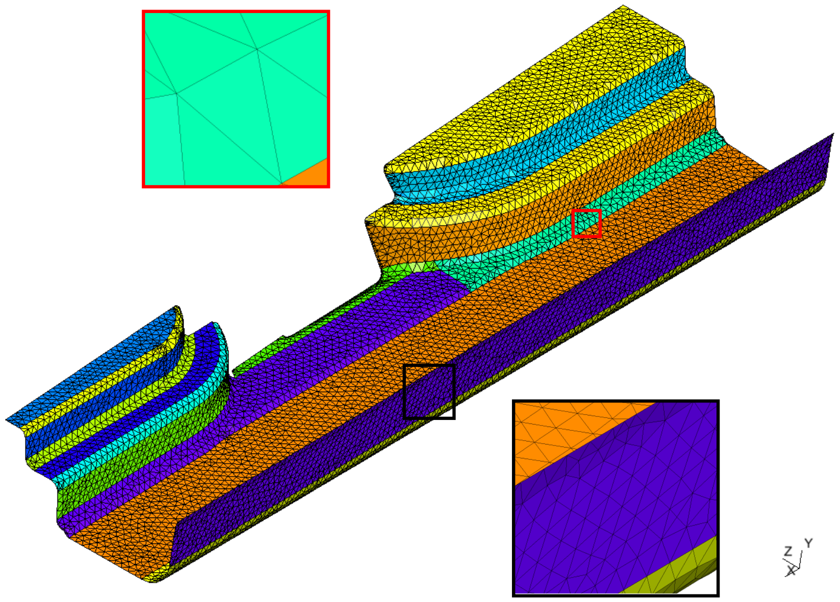


Fig. 8. Example of an initial triangulation of a CAD model from the automotive industry that contains a lot of gaps, overlaps and T-joints. The bottom figure shows the new mesh that is suitable for finite element simulations.