Repairing Non-manifold Boundaries of Segmented Simplicial Meshes

Tony L. Choque Ramos* and Alex J. Cuadros Vargas*
*Universidad Catolica San Pablo, Arequipa, Peru

***Abstract*—A digital image may contain objects that can be made up of multiple regions concerning different material properties, physical or chemical attributes. Thus, segmented simplicial meshes with non-manifold boundaries are generated to represent the partitioned regions. We focus on repairing non-manifold boundaries. Current methods modify the topology, geometry or both, using their own data structures and they are applied to a single region. The problem of modifying the topology is that if the mesh has to be post-processed, for instance with the Delaunay refinement, the mesh becomes unsuitable. In this paper, we propose alternatives to repair non-manifold boundaries of segmented simplicial meshes, among them is the Delaunay based one, we use common data structures and only consider 2 and 3 dimensions. We developed algorithms for this purpose, composed of the following tools: relabeling, point insertion and simulated annealing. These algorithms are applied depending on the targeted contexts, if we want to speed the process, keep as possible the original segmented mesh or keep the number of elements in the mesh.**

I. Introduction

Simplicial mesh generation from images can be divided in two approaches. (a) Meshes for representation, they aim to minimize the approximation error between the original image and the image represented by the mesh, for this reason the elements of the mesh usually do not meet quality criteria [1], [2]. (b) Meshes for numerical simulations, they consider certain quality criteria like number, size and shape of their elements [3], [4].

Meshes for representation frequently are generated directly from images, then if we consider a triangle or tetrahedron as a *superpixel* or *supervoxel* respectively [5], the mesh can be segmented [6], improved and became suitable for numerical simulation [7]–[9], however, it arises peaks that yield singularities, namely non-manifold boundaries.

The manifoldness quality is mandatory or very important for some applications, for instance, in surgical simulations [10], it has been explored in the field of Mesh Repairing as a consequence. Moreover, this quality is required to perform computations of smoothness on a surface, also continuous differential operators like normal and curvature are extended to the discrete case, almost the majority of geometric algorithms are not suitable if the mesh does not have this quality [11].

The significant contribution is the introduction of repairing algorithms for this atypical problem based on three tools: relabeling, point insertion and Simulated Annealing.

II. Related Work

In the field of irregular or unstructured meshes, triangular surface meshes are research objects [12]. In [13], it is proposed a method to convert a mesh with singularities into a manifold; it consists of identifying singular edges, those having $2k$ incident triangles, with k as constant, they are divided into k manifold edges, however the mesh may still contains singular vertices, they therefore are duplicated, the strategy is to produce the minimum number of duplications. Furthermore, [14] proposes a strategy based on two important operations: *cutting* and *stitching*, the former consists of cutting the surface around a singular edge, the result is a mesh with empty spaces, the latter operation involves joining two adjacent open edges while guaranteeing the manifoldness. Later, [15] uses the *cutting* and *stitching*, but the stitch operation is guided by the Simulated Annealing technique, this alternative helps to rebuild the mesh with certain guarantees on the shape.

For tetrahedral meshes, [10] introduces two conversion algorithms that are used according to their purpose: (a) Modify only the connectivity, through vertex duplications. (b) Modify the connectivity and geometry, it erodes small amounts of material around the singular object.

Finally, we present research based on tetrahedron labeling. One of the process in [11] uses a region growing of tetrahedrons labeled as *matter* or *freespace*, it proposes faster methods to detect a singular vertex based on a graph and add a tetrahedron preserving the manifold quality [16]. Similarly, in [17], tetrahedrons of a Constrained Delaunay Tessellation (CDT) are labeled as *inside* or *outside* using the minimization of a function depending on the winding number until a manifold mesh is obtained if it is possible.

III. Proposal

The input mesh M is a list of vertices $S \in \mathbb{R}^d$ and a list of cells C, each cell has $d+1$ facets (facets and cells correspond to triangles and tetrahedrons in $\mathbb{R}^3$ and edges and triangles in $\mathbb{R}^2$). Moreover, $M = \bigcup_{i=0}^{n-1} m^i$, where n is the number of submeshes, a submesh is a set of cells that do not necessarily form a closed space, it can be composed by several closed spaces, but is only made up of a single material. We can represent the submesh in M as the function $i : \sigma \in M \to \mathbb{N}$ that maps the σ cell to the material label, where $\mathbb{N}$ is the set of natural numbers with 0 depicting the background

The boundary ∂M is the list of facets which are included in exactly one cell of M. We introduce a practical use in triangulation data structure [18], the infinite vertex v_∞ ($v_\infty \notin$

$\mathbb{R}^d$, Figure 1) such that we define an infinite cell connecting a ∂M facet and v_∞, the set of infinite cells compound an abstract submesh m^{-1}, -1 is its material label, we denote the abstract mesh $M^{-1} = M \cup m^{-1}$, this will help us later for singular vertex detection.

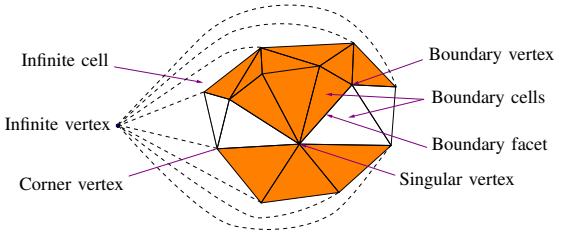


Fig. 1. Usual terms.

The boundaries of submeshes are denoted by ∂m^i, they are not assumed to be $(d-1)$-manifold, to fix them we use the relabeling (change of material label) with deterministic criteria and point insertion, but both do not assure repair all the singularities. Due to this reason, the simulated annealing technique is needed, it uses relabeling with probabilistic criteria and creating filling cells in the worst case. Our goal is that all boundaries ∂m^i would be $(d-1)$-manifold without filling cells in M.

A. Singular Vertex Identification

A topological space is $(d-1)$-manifold if every point in the space has a neighborhood homeomorphic to a $(d-1)$-ball (1-ball is an edge and 2-ball is a disk). In the discrete case, a polygonal curve is 1-manifold if all its vertices are regular, a vertex is regular if it belongs to at most two edges. In the same way, a triangle surface mesh is 2-manifold if all its vertices are regular, a vertex is regular if its opposite edges in the triangles form a simple polygon [11]. If a vertex is not regular, it is singular.

Let L be a set of cells, then L_v is all the cells in L that have v as vertex. g_v is the adjacency graph of M_v^{-1} (includes infinite cells), to find singular vertices we define the graph G_v similar to [16], it is obtained from g_v by removing the graph edges between a cell in m_v^a and other cell of m_v^b, where $a \neq b$, in Figure 2b we see the graph G_v when $d = 2$ and $d = 3$.

A component is the set of cells that depict a subgraph of G_v, m_v^i can have many components, we classify m_v^i according its number of components: .

- Triple-over-component, m_v^i has three or more components. v is a singular in ∂m^i.
- Double-component, m_v^i has two components. v is singular in ∂m^i.
- Single-component, m_v^i has one component. If $d = 2$, v is always regular in ∂m^i, on the other hand, when $d = 3$, v is regular in ∂m^i if the set of cells $M_v^{-1} \backslash m_v^i$ has a path in g_v, otherwise v is singular ∂m^i.

G_v also works for corner vertices, those that belong to an infinite cell (Figure 1). In the 3D case, finding singular vertices, involve finding singular edges. In both dimensions, v can be singular with respect to several submeshes, for this we save the material labels of the submeshes where v is singular in a vector P, decreasingly ordered according to the number of components.

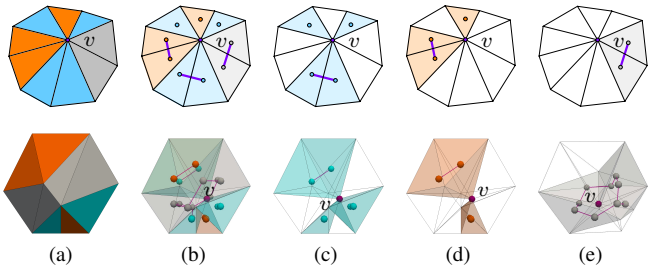


Fig. 2. A multi-material vertex in 2 and 3 dimension (a) and its G_v graph (b). A triple-over-component (c), double component (d) and single-component (e).

B. Relabeling

To repair a singular vertex v in ∂m^i, we relabel the cells around v in such a way m_v^i becomes single-component and v regular. For this we walk the vector P and perform the following:

1) Component Ordering: First, we get the label i of P. m_{vk}^i is a component, where $k = 1, 2, .., q$ and q is the number of components, for instance in Figure 3a, $q = 2$. We save each component m_{vk}^i in a vector named Q, increasingly ordered according the criterion described by the equation 1 or 2.

$$c_1 = \mu(m_{vk}^i) \tag{1}$$

Where $\mu(A)$ express the Lebesgue measure of a set A in $\mathbb{R}^d$. This criterion is convenient to modify as little as possible the segmentation.

$$c_2 = E(m_{vk}^i) \tag{2}$$

E returns the greatest distance of an edge in m_{vk}^i, it can modify the partition more than c_1, but increase the Hausdorff distance (between the output and original mesh) less than the previous.

2) Erosion or Dilation Criterion: To decide if m_v^i should be eroded or dilated we apply a criterion to each component m_{vk}^i, we compute R_v (the signed discrete curvature in a planar curve [19] and signed mean curvature in triangle surface mesh [20]) of v with respect to ∂m_{vk}^i, moreover, we compute $sum_R = \sum R_w$ where w is a regular vertex in ∂m^i and fits $w \in m_{vk}^i$ and $w \neq v$.

$$c_3 = \begin{cases} 1 \text{ if } (R_v)(sum_R) \geq 0 \\ 0 \text{ otherwise} \end{cases} \tag{3}$$

m_v^i is dilated if at least half of the elements, $\lfloor (q+1)/2 \rfloor$, in Q meet $c_3 = 1$, otherwise it is eroded. The Figure 3e presents a vertex with two components, where $c_3 = 1$ for m_{v1}^i, in this case we dilate m_v^i; conversely, m_v^i is eroded in Figure 3b, where $c_3 = 0$ for m_{v1}^i and m_{v2}^i. If m_v^i is a single-component in a 3D mesh, we always dilate. We explain each operation below.

- **Erosion.** We iterate Q to relabel each component until the next-to-last, namely, m_v^i becomes single-component. To get the new material label, we measure the boundary facets (length or area) in m_{vk}^i and identify the material label with which m_{vk}^i shares greater neighborhood or there exists greatest intersection. Finally, we update P. Notice when $d = 3$ a single-component does not mean v is regular in ∂m^i.
- **Dilation.** We take two different components k and l from Q and select a point y in m_{vk}^i (for instance, the centroid of a boundary facet) and other z in m_{vl}^i, then, we figure out a path of adjacent cells through the visibility walk [21] between y and z, Figure 3f. We relabel the cells in the path with i and update P, if there still exist more than one component, we repeat the process until m_v^i becomes single-component. For 3D special case, if m_v^i is single-component, but v is singular in ∂m^i, it is dilated by eroding the set of cells $M_v \backslash m_v^i$ using i as the new material label.

Due to relabeling can generate new singular vertices we mark all relabeled cells, so they will not be relabeled again, in order to avoid loops, thus relabeling may not work for all cases, other than scattered singular vertices.

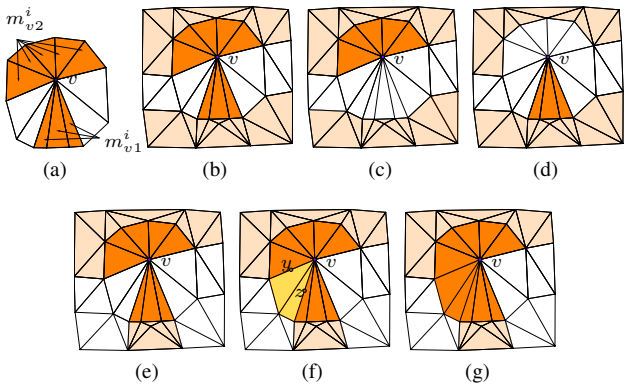


Fig. 3. A singular vertex v with its components (a), erosion case (b) repaired with relabeling using c_1 (c) and c_2 (d) criteria. Below, we show the dilation process (e-g).

C. Point Insertion

This tool is applied on a Delaunay based simplicial mesh that meets the following conditions, the cells in M are included in the convex hull of S, and the circumball (d-ball whose boundary passes through every vertex of cell) of each cell does not contain a vertex in its interior.

We select one component m_{vk}^i, in order to erode m_v^i through the insertion of w vertex. At the end of this process, the cells of m_{vk}^i are destroyed such that new cells with w as vertex are created, suppose for the 2D case, if abv is a triangle in m_{vk}^i with vertices a, b and v; after inserting w it creates a triangle abw of the same material, the same happens with all triangles in m_{vk}^i. Moreover, after inserting, the cells of other components m_{vl}^i, where $k \neq l$, and the cells outside M_v are not affected by the insertion. We try to find a position in space to insert a point w and fit the previous conditions, then, we do the following.

- **Circumball intersection.** We seek that the cells in m_{vk}^i would be destroyed, so the point to be inserted w should be inside the circumballs of all cells in m_{vk}^i, namely, w should be in the circumball intersection, if it exists (see details of computation in supplementary file).
- **Cell preservation.** In order to not affect cells of other components m_{vl}^i nor the cells outside M_v, we verify that w is not in the interior of their circumballs. If w fit this last condition, then we can erode m_{vk}^i.
- **Cell restoration.** After destroying cells whose circumballs were affected by w, we have a cavity, Figure (4c), due to M is Delaunay, we reconstruct the cavity as follows, in the 2D case there exist a new edge aw for each vertex a of the polygonal cavity, in the 3D case, there exist a new triangle abw for each edge ab in the polyhedral cavity [22], [23]; this allow us to recover the cells of m_{vk}^i where v is replaced by w as we show in the Figure 4.

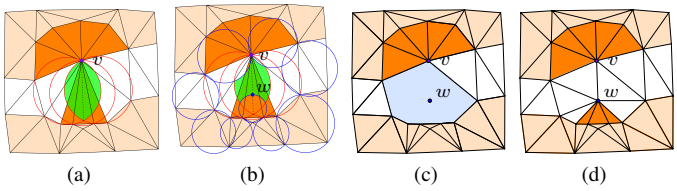


Fig. 4. Point Insertion, from left to right: Circumball intersection (red circumcircles), cells preservation (blue circumcircles) and cell restoration (c-d).

If $d = 2$, we apply the point insertion when m_v^i is double-component, Figure 5, the special case in Figure 5h is obtained by eroding the path of adjacent cells (See Section III-B2) through the point insertion.

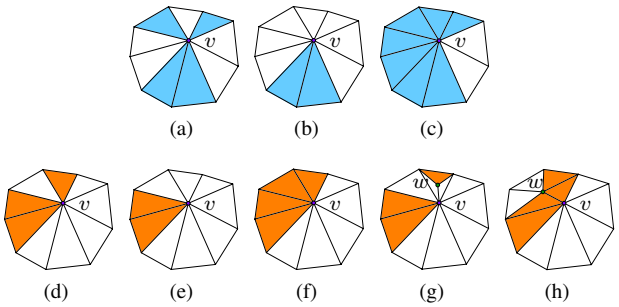


Fig. 5. Alternatives to repair a singular vertex $v \in \mathbb{R}^2$ in ∂m^i, when m_v^i is a triple-over-component (top) and double-component (bottom).

If $d = 3$, we use the point insertion when m_v^i is double-component or single-component, Figure 6. A single-component is dilated by eroding one of the components in $M_v \backslash m_v^i$ through point insertion, Figure 6j.

The point insertion tool is not always applicable, so we propose the simulated annealing as an additional tool.

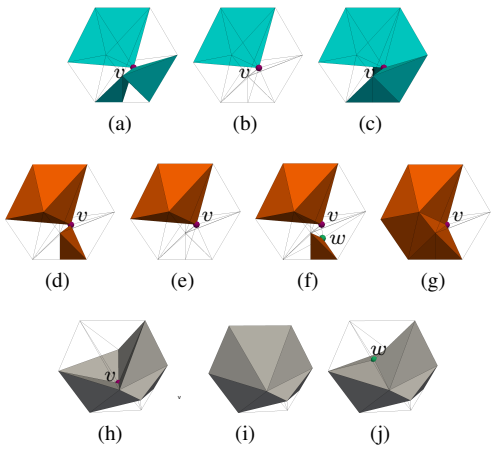


Fig. 6. Alternatives to ensure repair of a singular vertex $v \in \mathbb{R}^3$ in ∂m^i, when m^i_v is a triple-over-component (top), double-component (middle) and single-component (bottom).

D. Simulated Annealing

Is a computational stochastic technique of approximation to the global minimum of a given function $\varepsilon : Z \to \mathbb{R}$ which maps a valid state $z_t \in Z$ to the set of real numbers. To achieve the optimization, it is necessary define a set of neighboring states Z_t for each z_t.

The process begins with an arbitrary state z_t, next, we randomly select one of its neighboring states $z_u \in Z_t$, to know if z_u replace the current state, it is required to make an stochastic experiment between the probability

$$p(z_t, z_u) = e^{-\frac{\varepsilon(z_u)-\varepsilon(z_t)}{T}} \tag{4}$$

(where the temperature $T > 0$) and a random number; if $\varepsilon(z_u) < \varepsilon(z_t)$, then $p(z_t, z_u)$ is greater than 1, z_u replace the current state without making the experiment, the probability of accepting lower states decreases as slowly as T decreases, if T tends to 0 then $\lim_{T\to 0} e^{-\frac{\varepsilon(z_u)-\varepsilon(z_t)}{T}} = 0$, then there is no possibility of accepting lower states [15], [24].

Let M be the input mesh with singularities. We define the filling cell as a triangle or tetrahedron that allow us to repair the singular vertices in ∂m^i, where $0 \leq i \leq n-1$, they compound the filling submesh m^n where n is its material label, if σ_i is a cell in m^i it may become a filling cell σ_n through label change, our goal is that all ∂m_i would be $(d-1)$-manifold, although it can have filling cells.

1) The set of states: The set of states Z are all the valid segmented meshes M_t, we say that M_t is valid if all ∂m^i are $(d-1)$-manifold and it may contain filling cells in the worst case, however ∂m^n is not necessarily $(d-1)$-manifold. The input mesh is not a valid state, for it to be, it has to be preprocessed by the Random Repairing procedure, explained below, also we recover the original label of cells, it will be useful later.

2) The sets of neighboring states: The sets of neighboring states Z_t of a valid mesh $M_t \in Z$, are all the segmented meshes that can be derived of M_t, after performing the following steps:

- **Random label assignment.** We define the probability x_1 which if applied to every filling cell σ_n, if a random number is lower than x_1, then σ_n takes its original label, otherwise the material label with which shares the greatest neighborhood is assigned to σ_n. At the end of this process we recover the singular vertices, if exist.
- **Random repairing.** For each singular vertex we verify if it can be repaired without generating new singularities, for this, if a random number is lower than x_2 we erode, otherwise we dilate, for the 3D case of single-component we erode it by dilating the neighbor components. If this is not possible, we fix it creating filling cells such that M_t is valid, it consists of the same procedure than erosion, Figure 7. Creating filling cells occasionally generates new singularities, if it happens we continue creating filling cells until we get a valid state.

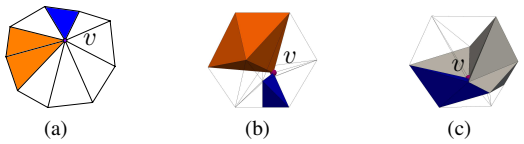


Fig. 7. Repairing with filling cells (blue). For 2D case, we repair a triple-over-component and double component in the same way as erosion (a). For the 3D case, we repair a triple-over-component and double component in the same way as erosion (b), but we repair a single-component by dilating its neighbor components with filling cells (c).

3) The cost function: The main goal is generate manifold boundaries without filling cells, then our cost function is defined as:

$$\varepsilon(M_t) = \text{ \# filling cells in } m^n; m^n \in M_t \tag{5}$$

Usually, in simulated annealing we do not know if minimum global is achieved in a given step. For our approach, we know that the minimum global of ε is 0.

4) A single simulated annealing Step: A step is performed in the following way, let M_t be the current valid mesh. First a neighbor mesh $M_u \in Z_t$ is chosen randomly. Then, the stochastic experiment is performed against the probability:

$$p(M_t, M_u) = e^{-\frac{\varepsilon(M_u)-\varepsilon(M_t)}{T}} \tag{6}$$

If the experiment is successful, the transition from M_t to M_u is accepted and M_u replace the current mesh, otherwise we keep M_t.

5) The complete simulated annealing process: Consists of s phases and each one has t steps. The phases differ between them by the drop in the temperature T. Let 0 be the first phase and $s-1$ the last phase. Then, the temperature T_k of the k phase is given by:

$$T_k = T_0(tf^k) \tag{7}$$

Where tf is the factor which controls how fast the temperature will change in each phase. The input mesh has a number of singularities, in our experiments we take $t =$ number of

singularities of the input mesh, also $T_0 = 1$ and $tf = 0.9$, for all experiments in 2D and 3D meshes we worked with $x_2 = x_1 = 0.8$, and the number of phases is $s = 100$, however, it ends before the phase 3 for all situations.

E. Repairing Algorithms

The relabeling tool imply a lower computational cost and does not change the geometry nor the topology of the mesh, but there no exists a guarantee that it fixes all the singularities. Point insertion is much more efficient than the relabeling because it certainly does not affect cells outside M_v, but it involves a high computational effort. The simulated annealing solves all the singularities by creating, in the worst case, filling cells and probably distorting the partition more than the previous tools. According to this description, we elaborate three algorithms summarized in Figure 8, each algorithm has its own qualities and are explained as follows.

1) Repairing 1 (REP1): It takes as priority the relabeling, if it is not possible we apply the point insertion, lastly we take all the remaining singular vertices that could not be repaired for the simulated annealing process. This algorithm repair the mesh with lower computational cost and try to not change the original segmented mesh.

2) Repairing 2 (REP2): The main option is the point insertion, if it is not possible we apply the relabeling and finally the Simulate Annealing. It is appropriate if we want to preserve the original segmented mesh as possible.

3) Repairing 3 (REP3): It only consists of the simulated annealing and its main advantage is that it does not insert points, namely, the number of cells in the mesh is preserved.

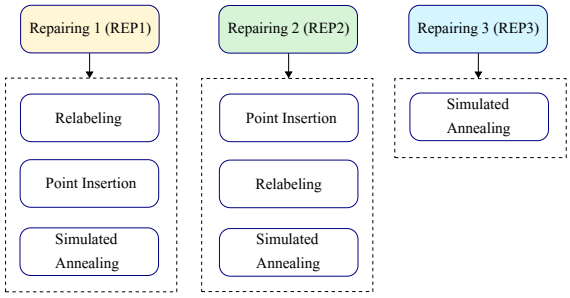


Fig. 8. Repairing algorithms description.

IV. Results

We discuss the different criteria, tools and algorithms to repair singularities of segmented simplicial meshes when $d = 2$ or $d = 3$, we have two distinctive contexts when $n = 2$ and $n > 2$. We took the meshes from [8], whose distinctive attribute is small cells near boundaries and larger cells in their interior, also they are Delaunay. The algorithm implementation was made in C++ [25] and using the *CGAL* library [26]. Our timings were obtained on an Intel(R) Core(TM) i7 2.50GHz processor with 12GB memory.

In order to evaluate different results we consider some measures. We denote the output mesh as M' with its set of vertices S', also $|S'|$ is the number of vertices in S', then the number of new vertices is $|S'| - |S|$. Furthermore, to measure the similarity between the input and output meshes we select the list of boundary facets from M' and M, next we compute $\delta = d_H/diag$ where d_H is the RMS symmetric Hausdorff distance, computed using [27] for $d = 3$ and adapted for $d = 2$ (we can see some results in Figure 9), moreover $diag$ is the bounding box diagonal of the list of original facets. δ is normally used to measure the similarity between triangle surface meshes, for our work we complement it by considering the modified space $\mu' = \frac{\mu(M)-\mu(M\cap M')}{\mu(M)}$, $\mu(M)$ is the area/volume (for $d = 2/3$) of M and $\mu(M \cap M')$ is the area/volume covered by cells of the same label and is computed using [28], we express μ' as a percentage and means the percentage of modified space. δ and μ' give us a hint about the similarity between M and M'.

A. 2D Results

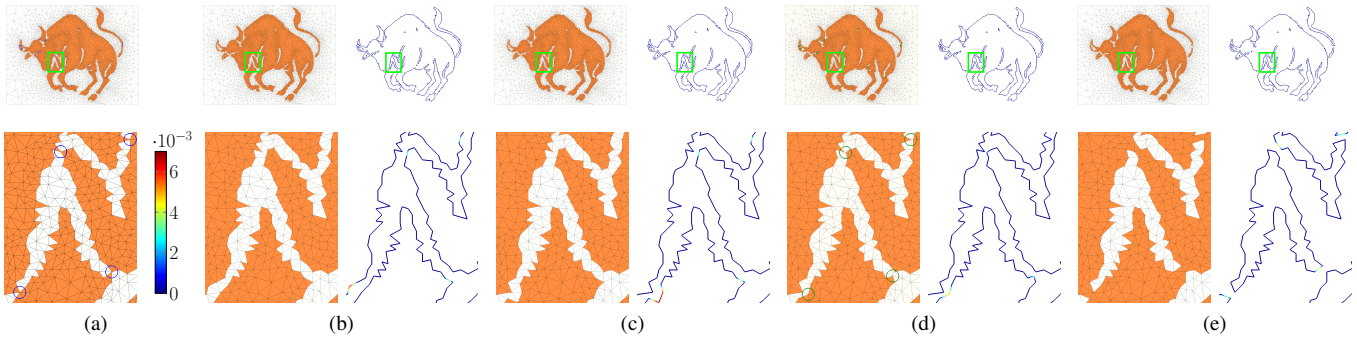
The mesh *Taurus* shown in Figure 9 contains 2 submeshes, 5505 vertices, among which 30 are singular. We could repair all the singularities using a single tool among relabeling, point insertion and simulated annealing, it is because most of the singular vertices are isolated and the components have few cells, then we can insert a point. In simulated annealing we do not use the c_3 criterion, thus, it visually distorts the segmentation more than the other tools.

In Table I, we compare the tools applied in *Taurus*. The point insertion tool is the only who creates new vertices, however in terms of noise, the point insertion is superior, because it modifies smaller area (μ') and its boundary is closer to the original boundary (δ) than the other results. On the other hand, c_1 modifies the area less than c_2 criterion, but its boundary is less approximate to the original than c_2, the tool which causes greater noise is the simulated annealing, for this model it took just one step because of scattered singular vertices. If we consider the time of execution, using c_2 has the best score, in contrast to point insertion, which has the worst result.

TABLE I
COMPARISON OF TOOLS APPLIED IN *Taurus*, FIGURE 9, IS A SEGMENTED MESH WITH 2 SUBMESHES AND 30 SINGULAR VERTICES.

Tool	$\|S'\| - \|S\|$	μ' (%)	δ ($.10^{-5}$)	Time (s)
Relabeling with c_1	**0**	0.117	9.121	0.007
Relabeling with c_2	**0**	0.120	7.562	**0.006**
Point insertion	30	**0.046**	**3.170**	0.124
simulated annealing	**0**	0.118	12.328	0.008

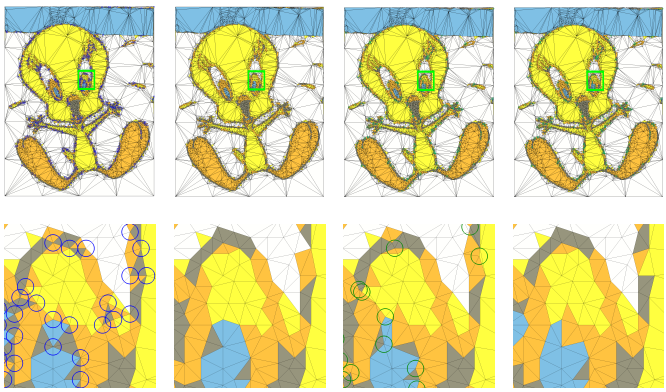
Tweety is showed in Figures 10 and 11, it is a mesh with 5 regions and 3125 vertices, among which 500 are singular, namely a little more than the sixth part of the vertices are singular, in this model we can evaluate the effectiveness of our algorithms for 2D meshes, due to dense areas with singular vertices, REP1 had to insert points in contrast to relabeling *Taurus*. REP2 has worked for the majority of cases, thus the number of new vertices is almost the same as the singularities



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

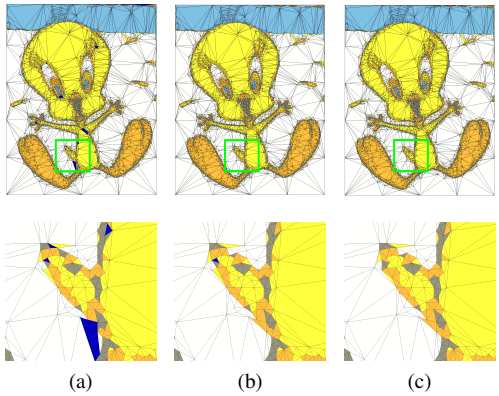
Fig. 9. *Taurus*: Singular vertices (a) are repaired applying the relabeling with c_1 (b), and c_2 (c), next, applying the point insertion (d) and finally, the simulated annealing (e). For all the cases we use c_3, except the last case. We also show the Hausdorff distance from the output set of boundary facets to the original set of boundary facets, it is normalized by the bounding box diagonal.

as we can see in Table II. This time REP3 has more steps than the previous case as we can see in Figure 11a.



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

Fig. 10. *Tweety*: It contains 5 submeshes and 500 singular vertices (a), they are successfully repaired by REP1 (b), REP2 (c) and REP3 (d).



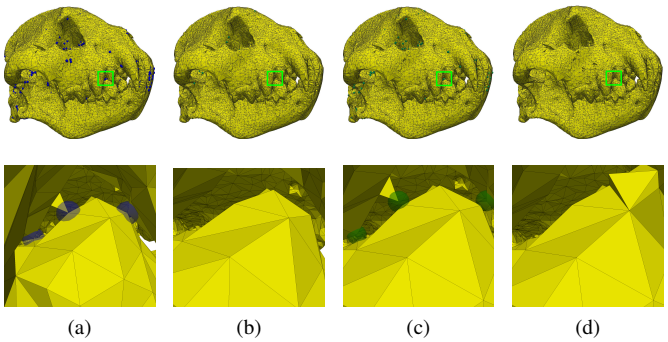
(a) (b) (c)

Fig. 11. REP3 iterations in *Tweety*. The first (a), forth (b) and the eighth step (c) which is the last.

The Table II shows some details of the repairing algorithms for 2D meshes with 2 or more submeshes, the set of meshes are *Taurus*, *Tweety*, *Titicaca Lake* and *Thundercats*, all of them were repaired successfully without creating filling cells. SA represents the number of singular vertices repaired by simulating Annealing and #sv is the number of singular vertices in the input mesh. Better results are highlighted in bold.

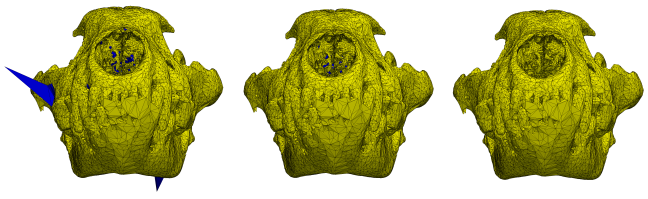
B. 3D Results

Hyena has 2 submeshes, 93808 vertices and 2139 singular vertices. In Figure 12 we see the repairing algorithms, they have the same qualities than their 2D version, but relabeling or point insertion can not repair all the singularities by themselves, each one requires at least the simulated annealing. REP3 works well as we can appreciate in Figure 13. Finally, as the 2D case, Figure 14 shows the Hausdorff distance from the output boundaries to the original boundaries normalized by $diag$.



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

Fig. 12. *Hyena*: It has 2 submeshes and contains 2139 singular vertices (a), we show the three-dimensional version of REP1 (b), REP2 (c) and REP3 (d).

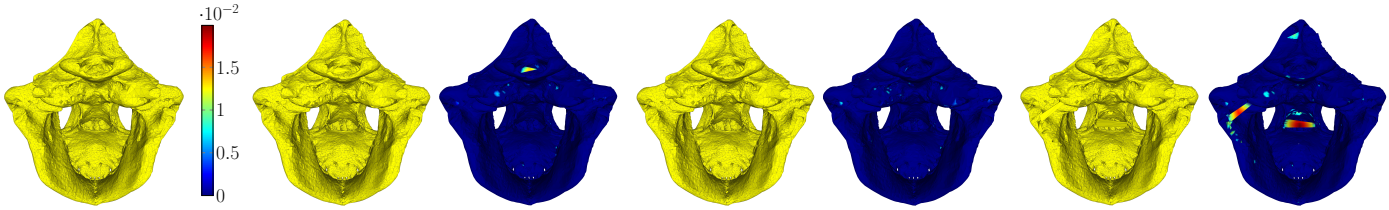


(a) (b) (c)

Fig. 13. REP3 iterations in *Hyena*. The steps number 1 (a), 5 (b) and 29 (c).

TABLE II
REPAIRING ALGORITHMS FOR 2D MESHES.

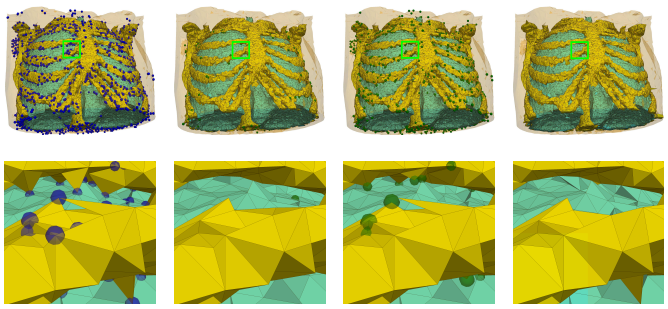
Model	n	$\|S\|$	#sv	Algorithm	Details of the algorithm					
					SA	$\|S'\|$	$\|S'\| - \|S\|$	μ' (%)	δ (10^{-5})	Time (s)
Taurus	2	5504	30	REP1	0	5504	**0**	0.130	7.562	**0.006**
				REP2	0	5534	30	**0.056**	**3.170**	0.124
				REP3	30	5504	**0**	0.152	12.327	0.008
Titicaca Lake	2	4364	17	REP1	0	4364	**0**	0.002	0.159	**0.005**
				REP2	0	4381	17	**0.001**	**0.073**	0.082
				REP3	17	4364	**0**	0.010	0.733	0.007
Tweety	5	3124	500	REP1	1	3140	16	2.054	38.490	**0.239**
				REP2	0	3527	403	**1.452**	**24.682**	2.111
				REP3	500	3124	**0**	2.440	78.417	0.073
Thundercats	2	2815	36	REP1	0	2815	**0**	0.325	44.442	**0.006**
				REP2	0	2851	36	**0.255**	**43.330**	0.291
				REP3	36	2815	**0**	0.475	266.551	3.253



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

Fig. 14. *Hyena*: Visualization of Hausdorff distance from the repaired meshes REP1 (b), REP2 (c) and REP3 (d) to the original mesh (a) normalized by the bounding box diagonal.

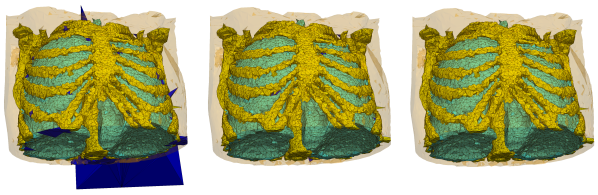
Chest has 4 submeshes with 44952 vertices among which 4903 are singular. The number of singular vertices is dense in some regions of the boundaries and some of them belong to at most 4 submeshes, this was an important model to evaluate our repairing algorithms, singularities were repaired successfully, Figure 15.



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

Fig. 15. *Chest*: It has 5025 singular vertices which are distributed in the boundaries of 4 submeshes (a). We can appreciate how singular vertices with three materials in its star are repaired with REP1 (b), REP2 (c) and REP3 (d).

The Table III shows a comparison of the repairing algorithms in 3D meshes, the set of meshes are *hyena*, *Head*, *Chest*, *Knee* and *Carp*, all the singularities in these meshes were repaired successfully without creating filling cells. The



(a) (b) (c)

Fig. 16. REP3 steps in *Chest*: it is required 10 step to repair all the singularities, we show the number 1 (a), 4 (b) and the final (c).

nomenclature is the same as the previous table (see other results in supplementary file).

V. CONCLUSION

The repairing algorithms worked for all the meshes from [8] using almost the same approach for $d = 2$ and $d = 3$. We proposed three tools: the relabeling, point insertion and simulated annealing, which were combined to produce three algorithms REP1, REP2 and REP3. Each algorithm has its own qualities and can be applied to different contexts. REP1 has lower time of execution and does not produce a lot of noise. REP2 keeps the original segmented mesh more than the others algorithm, but imply a high computational cost. Finally, REP3 does not insert any point in the mesh, but it highly distorts the segmented mesh.

TABLE III
REPAIRING ALGORITHMS FOR 3D MESHES.

Model	n	$\|S\|$	#sv	Algorithm	Details of the algorithm					
					SA	$\|S'\|$	$\|S\| - \|S'\|$	μ' (%)	δ ($.10^{-3}$)	Time (s)
				REP1	2	93953	145	0.041	0.342	**7.468**
Hyena	2	93808	2139	REP2	4	94822	1014	**0.023**	**0.130**	15.045
				REP3	2139	93808	**0**	0.053	1.183	18.884
				REP1	2	48892	290	0.119	0.651	**13.179**
Head	3	48602	4903	REP2	8	50729	2127	**0.074**	**0.344**	18.996
				REP3	4903	48602	**0**	0.155	1.053	21.853
				REP1	9	46161	1209	0.181	0.950	**27.219**
Chest	4	44952	5025	REP2	14	47770	2818	**0.141**	**0.553**	29.943
				REP3	5025	44952	**0**	0.790	2.571	37.210
				REP1	19	44110	291	0.205	0.779	**21.451**
Knee	8	43819	5162	REP2	10	46038	2219	**0.138**	**0.371**	28.689
				REP3	5162	43819	**0**	0.777	4.209	51.102
				REP1	10	32751	917	0.169	0.732	**18.921**
Carp	3	31834	3655	REP2	12	33871	2037	**0.109**	**0.374**	26.296
				REP3	3655	31834	**0**	0.253	3.263	32.282

The algorithms can also work for other simplicial meshes, beside the Delaunay one, we just do not have to consider point insertion, but they have not been tested for other meshes. We can improve some aspects like the c_3 criterion because of its computational cost or it can be set aside. With respect to the REP3 algorithm, we can add a cost function to improve the original shape while repairing the singularities like [15].

ACKNOWLEDGMENT

We would like to thank the financial support from Cienciactiva of the National Council for Science, Technology and Technological Innovation - CONCYTEC, Peru (grants FONDECYT 011-2013 Master Program and FONDECYT 142-2015).

REFERENCES

[1] M. D. Adams, "A highly-effective incremental/decremental delaunay mesh-generation strategy for image representation," *Signal Processing*, vol. 93, no. 4, pp. 749–764, 2013.

[2] K. Liu, M. Xu, and Z. Yu, "Adaptive mesh representation and restoration of biomedical images," *arXiv preprint arXiv:1406.7062*, 2014.

[3] P. Alliez, D. Cohen-Steiner, M. Yvinec, and M. Desbrun, "Variational tetrahedral meshing," in *ACM Transactions on Graphics (TOG)*, vol. 24, no. 3. ACM, 2005, pp. 617–625.

[4] F. Labelle and J. R. Shewchuk, "Isosurface stuffing: fast tetrahedral meshes with good dihedral angles," in *ACM Transactions on Graphics (TOG)*, vol. 26, no. 3. ACM, 2007, p. 57.

[5] O. Veksler, Y. Boykov, and P. Mehrani, "Superpixels and supervoxels in an energy optimization framework," *Computer Vision–ECCV 2010*, pp. 211–224, 2010.

[6] X. Chen and S. Wang, "Superpixel segmentation based on delaunay triangulation," in *Mechatronics and Machine Vision in Practice (M2VIP), 2016 23rd International Conference on*. IEEE, 2016, pp. 1–6.

[7] M. Španěl, P. Kršek, M. Švub, V. Štancl, and O. Šiler, "Delaunay-based vector segmentation of volumetric medical images," in *Computer Analysis of Images and Patterns*. Springer, 2007, pp. 261–269.

[8] A. J. Cuadros-Vargas, M. Lizier, R. Minghim, and L. G. Nonato, "Generating segmented quality meshes from images," *Journal of Mathematical Imaging and Vision*, vol. 33, no. 1, pp. 11–23, 2009.

[9] M. A. Liziér, M. F. Siqueira, J. Daniels II, C. T. Silva, and L. G. Nonato, "Template-based quadrilateral mesh generation from imaging data," *The Visual Computer*, vol. 27, no. 10, pp. 887–903, 2011.

[10] M. Attene, D. Giorgi, M. Ferri, and B. Falcidieno, "On converting sets of tetrahedra to combinatorial and pl manifolds," *Computer Aided Geometric Design*, vol. 26, no. 8, pp. 850–864, 2009.

[11] M. Lhuillier and S. Yu, "Manifold surface reconstruction of an environment from sparse structure-from-motion data," *Computer Vision and Image Understanding*, vol. 117, no. 11, pp. 1628–1644, 2013.

[12] M. Attene, M. Campen, and L. Kobbelt, "Polygon mesh repairing: An application perspective," *ACM Computing Surveys (CSUR)*, vol. 45, no. 2, p. 15, 2013.

[13] J. Rossignac and D. Cardoze, "Matchmaker: Manifold breps for non-manifold r-sets," in *Proceedings of the fifth ACM symposium on Solid modeling and applications*. ACM, 1999, pp. 31–41.

[14] A. Guéziec, G. Taubin, F. Lazarus, and W. Horn, "Converting sets of polygons to manifold surfaces by cutting and stitching," in *Visualization'98. Proceedings*. IEEE, 1998, pp. 383–390.

[15] M. Wagner, U. Labsik, and G. Greiner, "Repairing non-manifold triangle meshes using simulated annealing," *International Journal of Shape Modeling*, vol. 9, no. 02, pp. 137–153, 2003.

[16] M. Lhuillier, "2-manifold tests for 3d delaunay triangulation-based surface reconstruction," *Journal of Mathematical Imaging and Vision*, vol. 51, no. 1, pp. 98–105, 2015.

[17] A. Jacobson, L. Kavan, and O. Sorkine-Hornung, "Robust inside-outside segmentation using generalized winding numbers," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 4, p. 33, 2013.

[18] S.-W. Cheng, T. K. Dey, and J. Shewchuk, *Delaunay mesh generation*. CRC Press, 2012.

[19] M. Saba, "On the usage of the curvature for the comparison of planar curves," 2014.

[20] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr, "Discrete differential-geometry operators for triangulated 2-manifolds," in *Visualization and mathematics III*. Springer, 2003, pp. 35–57.

[21] O. Devillers and R. Hemsley, "The worst visibility walk in a random delaunay triangulation is o (n)," *Journal of Computational Geometry*, vol. 7, no. 1, pp. 332–359, 2016.

[22] A. Bowyer, "Computing dirichlet tessellations," *The computer journal*, vol. 24, no. 2, pp. 162–166, 1981.

[23] D. F. Watson, "Computing the n-dimensional delaunay tessellation with application to voronoi polytopes," *The computer journal*, vol. 24, no. 2, pp. 167–172, 1981.

[24] R. H. Otten and L. P. van Ginneken, *The annealing algorithm*. Springer Science & Business Media, 2012, vol. 72.

[25] B. Stroustrup, *Die C++-Programmiersprache: aktuell zum C++ 11-Standard*. Carl Hanser Verlag GmbH Co KG, 2015.

[26] A. Fabri and S. Pion, "Cgal: The computational geometry algorithms library," in *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*. ACM, 2009, pp. 538–539.

[27] P. Cignoni, C. Rocchini, and R. Scopigno, "Metro: Measuring error on simplified surfaces," in *Computer Graphics Forum*, vol. 17, no. 2. Wiley Online Library, 1998, pp. 167–174.
[28] D. Powell and T. Abel, "An exact general remeshing scheme applied to physically conservative voxelization," *Journal of Computational Physics*, vol. 297, pp. 340–356, 2015.