Reconstruction of Volumes from Soup of Faces with a Formal Topological Approach

Hakim Belhaouari[1], Sebastien Horna[1],

[1]University Of Poitiers, XLIM UMR CNRS 7252, first.last@univ-poitiers.fr

Corresponding author: Hakim Belhaouari, hakim.belhaouari@univ-poitiers.fr

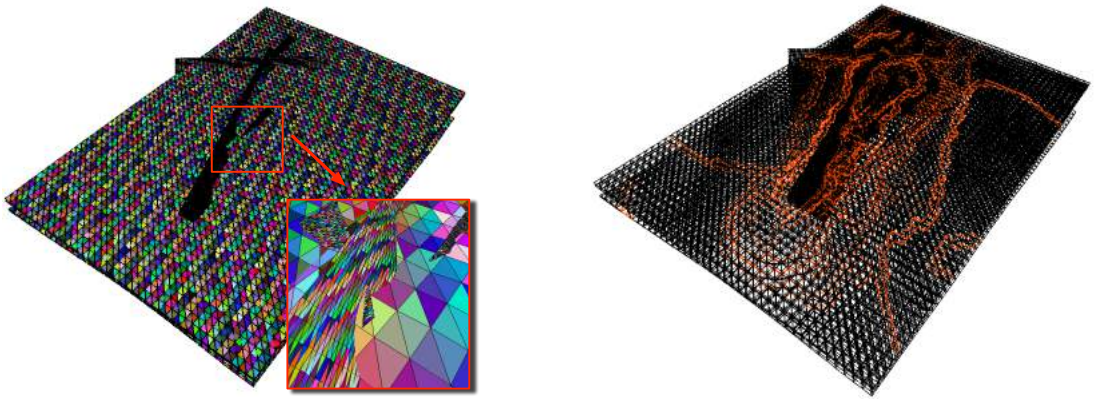
Abstract. Digital 3D models are widely used in many application fields. While the models are required by several applications, such as architecture, geoscience or simulation, they often lack vital information for the final business application. Models are therefore often rebuilt to meet the requirements of a specific application. Unfortunately, this reconstruction is a tedious task and consumes time and resources. This paper presents a general algorithm for reconstructing 3D objects with 3D topological information and without geometrical inconsistencies. Firstly, a cellular subdivision is performed by relying on geometrical input data and then topological relationships between cells are explicitly defined. A 3D generalized map (3-G-map) data structure handles reconstruction information. Secondly, faces are split along their intersections in order to satisfy a space partition constraint and built a set of volume without overlapping. One of our notable contribution consists in solving the problem of 3D intersection by using topological information to improve speed performance and to handle geometrical imprecision.

Keywords: 3D modeling, Topological reconstruction, Volumes and faces intersection
DOI: https://doi.org/10.14733/cadaps.2019.972-984

1 INTRODUCTION

Nowadays, digital 3D models are key components in many industrial and scientific sectors, such as architecture, geoscience, simulation and obviously computer graphics. Numerous tools propose to build virtual objects, either by construction from scratch, or by reconstruction from acquisition data (point cloud, photos, and so on). However, each application has its own quality requirements that restrict the class of acceptable and supported models [8]. For example, the models used within 3D printing framework must have a volume representation to ensure thick zones in fragile parts. In simulation, the neighborhood relationships between volumes and faces are necessary to compute energy transfers (acoustics, heat or lighting) [13, 22]. In geometrical and topological modeling, complex operations require a volumetric and spatial closed description of objects (Boolean operations, hole-filling).

Although 3D modeling is more and more successful, 3D models often correspond to a geometrical representation: either by a set of faces, or (in rare cases) by a set of volumes. No additional topological information is available, except the face neighborhood of volumes (according to the used modeler). Hence, the produced objects may still contain inconsistencies, especially interpenetrations between elements of the same dimension (volume to volume, face to face or edge to edge). Indeed, a geometrical correction and a dedicated information reconstruction are relevant in most application contexts (especially simulation applications that are usually based on volume information). For example, Figure 1 represents a real set of geological surfaces, which cannot be used for detecting petroleum reservoir through flow simulation, due to numerous face intersections and overlapping of the model, as illustrated in the zoomed part of Fig. 1(a). Classical filling-hole algorithms cannot be applied in order to avoid mix of semantics between faults and horizons and modifications on these meshes must be limited. Figure 1(b) highlights in red all interprenetration errors. As standard 3D operations cannot be used to correct the model, a dedicated process must be used to build a valid model [9, 16, 21, 24].



(a) (b)

Figure 1: Reservoir model generated from geological data (set of geological faults and horizons): (a) Direct interpretation of seismic and drilling survey with overlap faces; (b) Detection in red of all geometrical errors that forbid flow simulation.

Based on previous researches [6, 7, 15, 20], a large survey of model repairing methods is proposed in [8]. More precisely, authors distinguish two main categories for repairing models: global volumetric and surface approaches. Volumetric approaches consist in reconstructing a new valid mesh after transformation of existent mesh into an intermediate mesh: as oriented manifold [15], binary space partition tree [20], Nef polyhedra [17]. These methods are often too time consuming and relies on chosen intermediate structure. On the contrary, it solves highly inconsistent model in a robust way [8, 22]. Surface approaches work directly on the input mesh, and attempt to repair locally (self-)intersections *one by one* [6, 22]. Thus, modifications on original mesh are very limited to inconsistent parts. However, the resolution of intersecting geometry is numerically unstable which may produce artifacts. Finally, above mentioned methods can not respect at same time robustness, accuracy or efficiency properties. More, some methods have proposed to efficiently remove some simple inconsistencies (complex edges or singular vertices), but they are not fully automatic [15, 20]. Moreover, the produced object is not based on a formal topological model.

Several modelers or dedicated software offer correction operations. MeshLab [12] allows to detect polygon intersections and to delete corresponding faces: thus, a hole is created then filled with others techniques. ReMESH [4] exploits MeshFix, a correction library developed from the work of [2, 3]. This software is based on an algorithm that strives to convert a low-quality digitized polygon mesh to a single manifold and watertight mesh triangle without degenerate or intersecting elements. However, this method is based on triangles processing and can only handle one connected component. Whereas, the proposed approach in this paper allows to work on any polygons (not limited to triangles and/or quadrangles) and produce a set of volumes from intersections of several connected components if necessary.

In the same vein, a comparison with Boolean operations between two volumes is often made (corresponding to two connected components). Our purpose is clearly different, because Boolean operations produce, from surface objects (i.e. 2D topological objects with immersions in 3D), a unique volume built from them. In addition, processed volumes must be spatially closed (they have an interior and an exterior) in various simulation as visibility problem in rendering [18]. Thus, our approach keeps all topological volumes called co-refinement (similar to Minkowski sum, excepted objects are not necessary convex).

In this paper, we present a topological reconstruction method of volumes from soup of polygons in a robust, accurate and efficient way. The main purpose is the direct production of a 3D model with all topological information in any dimension (i.e. neighborhood of edges, faces and volumes) and all embedding information (color, domain properties and so on) for various simulation algorithms. The used topological structure loads directly a mesh without intermediate conversion. To achieve the reconstruction process, cleaning and repairing stages are added, in particular to handle the possible overlapped elements (edges, faces, volumes). One of our scientific contribution is this correction method that relies on the topological information in order to improve efficiency and to control geometrical imprecision. More, embedding information (domain specific data) are handled automatically along reparation in order to maintain consistent business model. Finally, the topology provides enough information on corrected objects to be used directly in many application domains (3D printing, simulation, visualization).

The next section presents the fundamental background, in particular the topological model used in our work. Our contributions are detailed in Sections 3 and 4, where the first one shows the importation from raw data and the topological reconstruction process in accordance with the closure criteria and the second focuses on the repairing operations. Some results are given and discussed in Section 5. Finally, Section 6 concludes and provides some prospects.

2 FUNDAMENTAL BACKGROUND

Many data structures exist in computer graphics to represent cells subdivisions with their adjacency relations [10], among which generalized-maps (G-maps) are a formalization of all these concrete data structures [19]. G-maps also present a formal and mathematical definition that allows us to address robustness problems. Similar topological models have already been used in topological reconstruction [14, 18] and refinement process [9, 11].

2.1 Generalized map

The main advantage of G-maps model is a uniform description through dimension of structure, operations and additional defined properties. Here we use a non-oriented graph [5], where nodes (denoted by the set G) are basic elements called darts and arcs represent the adjacency relationship. Each arc is labeled depending on the dimensional relationship. Commonly, we use notation α_i for representing a link between two cells of dimension i and d for the basic element (darts). Thus, a $3-$G-map represents intuitively a 3D object from its decomposition into topological cells (volumes, faces, edges, vertices).

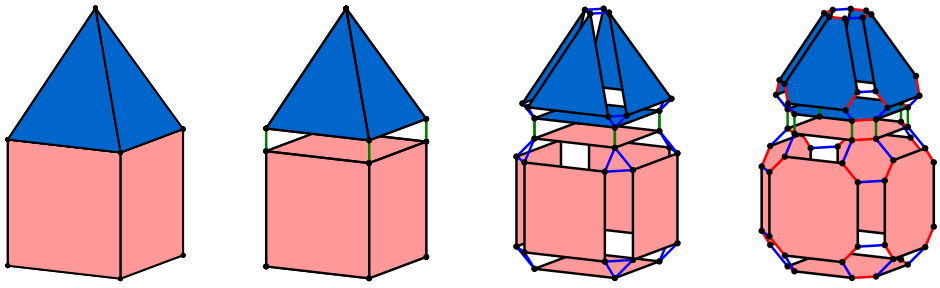


Figure 2: Cell decomposition of a geometric 3D object: (a) 3D topological object; (b) Volume decomposition: two adjacent volumes are linked along an α_3 relation (green lines); (c) Face decomposition: two adjacent face are linked along an α_2 relation (blue lines); (d) Edge decomposition: two adjacent edge are linked along an α_1 relation (red lines) and edge is α_0 relation (black lines) and it forms final $3-$G-map.

Figure 2 illustrates the topological decomposition of a 3D object composed of a pyramid over a box. The dimensional relationship are colored lines between cells. In the remain article we display them or not depending on point of interest in order to not overload figure. And we use sparingly exploded view to more appreciate topological relationship.

2.2 Additional required properties

G-map may represent oriented and non-oriented objects. Here, we want to handle real-world objects as a set of disjoint volumes. We must define a set of properties for identifying an acceptable object. That is why we use a 3D space subdivision that helps us to define formally closed and oriented 3D partition.

The space partition criteria relies on a mix of geometry and topology, where we define C_i as the geometrical space of a $i-$cell (e.g. C_3 defines the interior space of a volume, C_2 defines the area of a face, C_1 is the length of an edge, C_0 is the vertex itself). Formally, a 3D object is well formed if and only if a space partition P satisfies the Equation 1. In other words, two cells i must not occupy the same space and/or must be connected along cells of lower dimensions.

$$\forall i \in \{0,1,2,3\}, \forall C_i^1, C_i^2 \in P, C_i^1 \cap C_i^2 = \emptyset \vee \exists j \in [0;i[, C_i^1 \cap C_i^2 = C_j \text{ where } C_j \in P \tag{1}$$

The object closure property is well known in the Computer Graphics community but with $3-$G-map this property differs and becomes a topological closure defined in the Equation 2. Less formally, it means there is no self-loop in the graph thus opened faces are prohibited. This property relates to the dangling face, a specific case where $d.\alpha_2 = d.\alpha_3$ that corresponds to a geometric hole. In other word, after topological reconstruction holes are detected immediately.

$$\forall d \in G, \forall i \in [0;3], d.\alpha_i \neq d \tag{2}$$

The last property is the orientation: G-maps may represent oriented (or not) objects. For representing orientation, we include a special boolean embedding (often named *orient*). Formally, the property behind is defined in the Equation 3 where orient accesses to the special boolean and it forces that two adjacent darts have an opposed value for any dimensions.

$$\forall d \in G, \forall i \in [0;3], d.orient \neq d.\alpha_i.orient \tag{3}$$

3 TOPOLOGICAL RECONSTRUCTION PROCESS

Our contribution lines up to repair/clear soup of faces to obtain volumes. Initially, the load process reconstruct basic faces (at this time we check if faces are flat with at least three vertices or edge length is null, and correct them if necessary) and affect initial embedding information depending on the model (for instance orientation, color or nature of element as material). Then the topological reconstruction process computes the topological link between faces that have a common geometric edge. The second step consists in correcting interpenetration elements was detail in next section.

3.1 Pre-process: import data

3D models are often recorded in file format that relies on faces indexed by vertices (eg. surfaces as the OBJ, OFF, PLY, or any supported format from Assimp library).During the parsing of vertices, we use a balanced tree (AVL Tree[1]), where keys are points and each stored value is a pair compound of current vertex index and a list of darts which are at this 3D coordinate. The keys are ordered according to their components (x,y,z). When a new point is inserted/computed, the process searches if it already exists in the tree, then it is associated with the existing index else it is registered. The advantage of this structure is the efficiency of standard operations (especially insertion and finding redundant elements).Probably, any other acceleration structure could be use to manage all geometrical points as Kd-tree or BVH. However, AVL Tree is more flexible for dynamic insertion that avoids complex reconstruction (if needed) and it offers practical internal use for associating index between geometry and darts. This study is not the main focus of this article.

When the process completes the registration of all vertices. The first step of the reconstruction is creation of all topological faces. Figure 3 shows all steps of the topological reconstruction process. After, registration in the tree, object looks like in Figure 3(a), where data has been corrected if needed. Topologically, each face counts $n * 4$ darts (where n is number of edges), and topological faces are created by α_0, α_1 and α_3 links as illustrated in Figure 3(b) (a single face with two sides – called facets, one for each incident volume). This step produces (by construction) $0-$closed, $1-$closed, and $3-$closed topological faces (actually α_0, α_1 and α_3 links two different darts); an orientation mark is fixed on the face darts in order to satisfy the orientation property. Next step of the process consists in closing α_2-links (connecting topological face) in accordance to the orientation property through an angle arrangement algorithm.

3.2 Angle arrangement

At this stage, α_2-links are open and do not follow closure property (see Eq. 2). The angle arrangement corrects it, produces volumes and satisfies partially consistent space partition. In practice, the angle arrangement connects around geometrical edges, incident topological faces in a specific order (see Fig. 3). Here, AVL tree is heavily used for retrieving quickly topological faces (representing darts) that are incident to a geometric edge. For that, it retains darts that are α_0-linked in dart list of endpoints of the geometrical edge. Then, the process chooses randomly an incident face as referee and sorts according to their angle around edge from referee (see Fig. 3(c)). Finally, representing darts are successively α_2-sewed around the edge depending on the angle and orientation marks (see Fig. 3(c_1 and c_2)). This angle arrangement handles dangling faces (object border) with the same process without trouble as shown in Figure 3(c_3).

The topological reconstruction process guarantees the respect of closure and orientation properties. Indeed, topological faces construction (steps 1) directly defined the closing of the topological links α_0, α_1 and α_3 while respecting the orientation constraints. In a second step, the angle arrangement algorithm computes the α_2 links while also respecting the orientation property. Thus the angle arrangement process guarantees the production of an oriented and closed 3D model. In practice, one topological face is divided into two facets (linked along α_3 relationship). During the reconstruction process, when two topological faces must be sewed, the first one is chosen by selecting facet with open α_2 relationship. Then, the second one is selected depending on facet which has an opposite orientation.

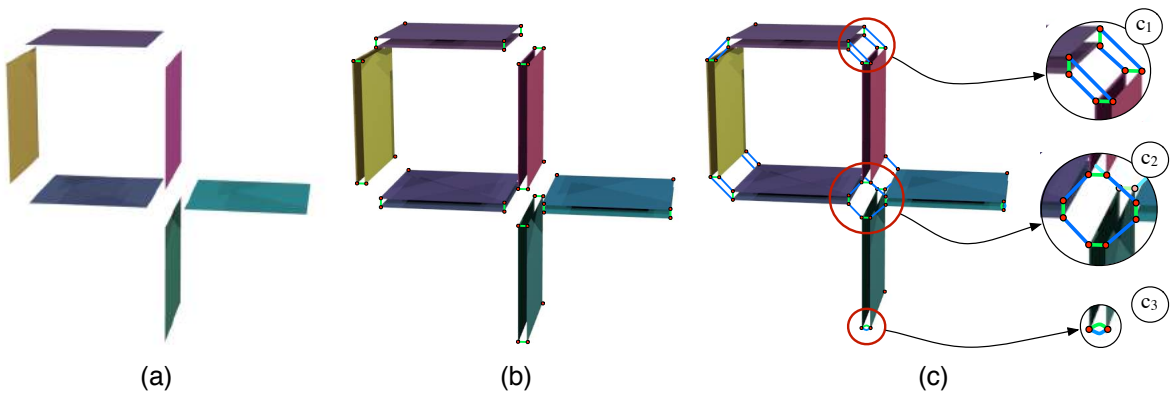


Figure 3: Topological reconstruction (exploded view): (a) original soup of faces; (b) each geometrical face correspond to a single topological face: with α_0, α_1 and α_3 closed, α_2 opened; (c) closure of α_2 relationship: all faces are linked according to angle around geometrical edge; (c_1) two faces are directly sew according to the orientation; (c_2) several faces are sewed according to orientation and angle arrangement; (c_3) dangling face: geometrical edge (E) with a unique incident face where for all dart d in the edge $d.\alpha_2 = d.\alpha_3$.

Figure 4 presents result of the topological reconstruction process. From a set of polygon, our topological reconstruction process produces a set of connected volumes (see Fig. 4(b)). Thanks to topological information, unclosed volumes which contains dangling face are directly detected (see Fig. 4(c)). In spite of all reconstructed volumes are delimited by a set of faces linked to other faces or not, the space partition criteria is partially satisfied. Geometrically, when we consider two faces, they may cross themselves. Actually, interior or exterior volume intersects interior or exterior of another volume. It may be similar for lower dimensions. Thus, we need the co-refinement process in order to complete this property. This is the purpose of next section.

4 3D CO-REFINEMENT

The second step consists in cleaning/repairing because up until this point, the reconstruction process does not completely satisfy the partition criteria. Unfortunately, the original model includes several other inconsistencies as shown in Figure 4(d). We shall now focus on presenting the 3D co-refinement, which splits objects correctly until satisfaction of the partitioning constraints. Volume intersections (and self intersection) are solved by 3D co-refinement of lower dimension cells. We therefore use two types of co-refinement according to the dimension of the treatment for our object embedded in 3D. One of our notable contribution consists in solving the problem by using the topological information to accelerate computations by reducing possible trouble concerning numeric precision. Indeed if a cut occurs on a topological edge, all incident faces will cut their edge simultaneously, thus a unique computation cuts multiple faces at once and avoid redundant computation. Finally, the 3D co-refinement process is composed of two steps. Firstly, the $1-$co-refinement guarantees the insertion of all vertices corresponding to edge intersections. Secondly, the $2-$co-refinement corresponds to split faces on their intersections.

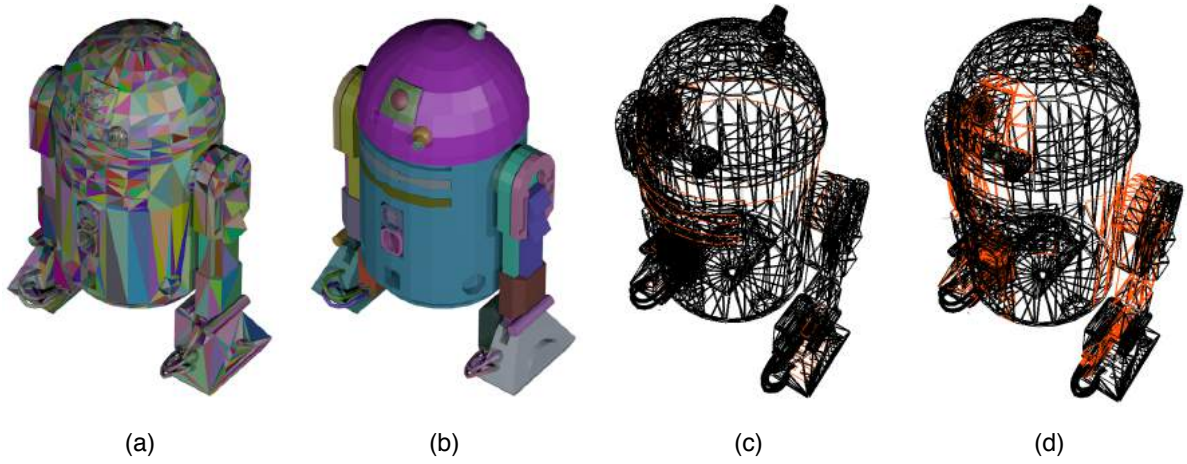


Figure 4: Topological reconstruction result: (a) initial set of geometric faces; (b) after the topological reconstruction, several volumes are built (one per color); (c) immediate detection of dangling face; (d) visualization of face intersections (in red).

4.1 Split edges ($1-$co-refinement)

Edges intersection process ($1-$co-refinement) relies heavily on computing intersections between topological edges. As a topological edge regroups multiple geometrical edges (one per incident face), the number of tested elements is reduced. Furthermore, the insertion of vertex impacts automatically all incident faces without additional computation. This operation is common in topological model and is very efficient. Finally, $1-$co-refinement process consists in the computation of intersections for all pairs of crossed edges. We consider only intersection points which are strictly inside processed edges. The corresponding topological edges are split according to the new inserted vertex.

4.2 Split faces ($2-$co-refinement)

The $2-$co-refinement handles face intersection in 3D. This feature takes into consideration the support plane besides each face polygon. In 3D, The intersection of two planes is a line, if they are not parallel. When the intersection exists, it is possible to split really both faces along the *splitting line* with some constraints.

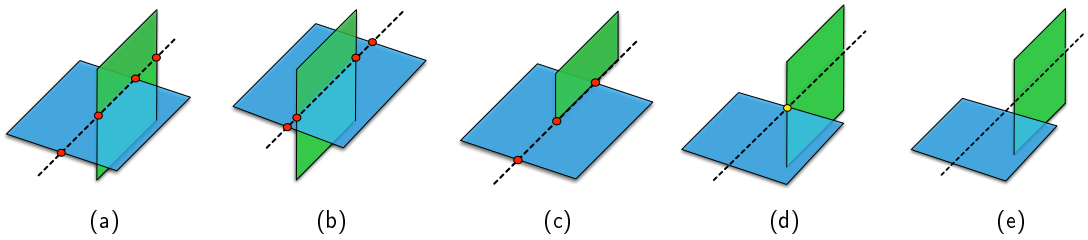


Figure 5: All cases of true intersections in 2-co-refinement: (a) and (b) split operation with insertion of points in both faces; (c) limit split case when green face border touches blue face; (d) face intersection corresponds to a single vertex; (e) no intersection occurs.

Concretely the split decision becomes a $1D$ problem depending on the intersection between the splitting line and each edge of faces. The study of cross interval and origin of the stored information indicates if the split is required or not. The Figure 5 shows different configurations. Figures 5(a-c) present cases that require a split because at least one intersection point of the green face (in red circles) is surrounded by two intersection points of the blue face. Figure 5(d) is not modified by the $2-$co-refinement, the intersection point is a single vertex that is detected and treated by the $1-$co-refinement (no face split operation occurs at this stage). Obviously, the Figure 5(e) shows case where no relevant point is surrounded, then no split operation occurs.

In a real complex model, due to the great number of polygons, a face may be intersected by several other faces. A naive approach consists in applying the previous process on each pair of face. But this mechanism drives to unstable computation due to the multiple application of the 2-co-refinement. To achieve that, one of our contribution is to buffer all splitting lines in an ad-hoc structure without effective split. A splitting line stores all intersection vertices along its director vector from all edges of each intersected face. Then, stored intersection vertices are sorted along the director vector of the splitting line. Figure 6 illustrates different examples where the intersections of at least two faces generate splitting lines. An analysis of this line with all vertices delimit for each face a segment with a set of additional vertices that income from other faces and are inside the segment. Then, faces are finally cut along this resulting segment with all including intersection vertices. Thus, if several faces share a same split line, then all intersections vertices may be reported on each concerned face. Figure 6(a) illustrates three faces that share a same splitting line where all red points are stored and impact other faces that contain them before effective split operation (see Fig. 6(b)). A more complex case treats intersection between splitting lines. The process is similar to the previous case but intersection vertex of splitting line is added to the basic process. Figure 6(c) illustrates a non-trivial case where three splitting lines cross themselves into the red point. Figure 6(d) presents effective split operation where each face part added the crossed vertex.

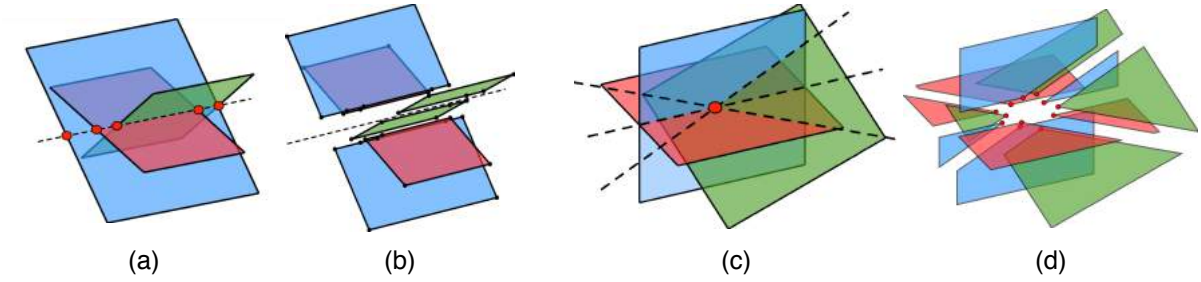


Figure 6: Intersection between several faces: (a) 3 faces are intersected on the same splitting line; (b) the 3 faces are cut, and all included vertices are added on new faces; (c) 3 faces are intersected on different splitting lines; (d) the 3 faces are cut, and splitting line intersection point is added on each new face.

Figure 7 shows a typical case where topology improves and reduces computations. A face is α_2-sewed and another face intersects the first without intersecting adjacent faces (see Fig. 7(a)). Then, the effective split occurs on intersected face (normally) and topology forces addition of new vertices on adjacent faces automatically (see Fig. 7(b)). This implicit operation avoids a recall to the $1-$co-refinement and the angle arrangement for repairing locally this trouble.

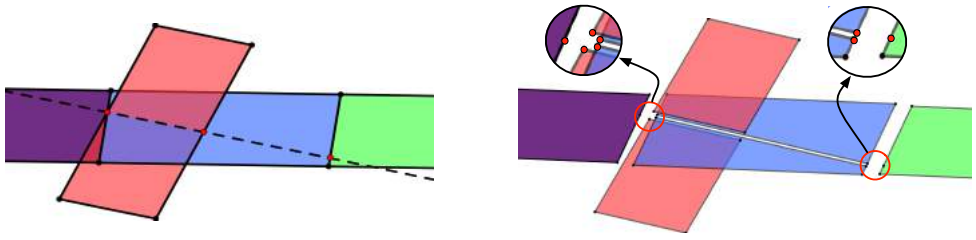


Figure 7: Topological reconstruction impact: (a) 4 faces where three are α_2-sewed and red one crosses only the blue one; (b) After effective split: topology inserts automatically new vertices on adjacent faces.

4.3 Merge of faces

After $2-$co-refinement, the partition constraint may not be complied as some faces cover the same space. The merge operation relies on the property that after the 2-co-refinements, some faces have strictly the same geometry and topology. But embedded values may be different as color or any additional semantics information (material tag, and so on). The detection of such faces therefore becomes easy cause topology gives a simple search around the face and compare each encountered vertex.

Here, the merge operation for overlapped faces depends heavily on the business logic. For instance, in rendering, this operation may mix light properties from both faces in order to create a new material with both properties. Alternatively, in architecture, both overlapped faces may have different semantic label and the operation tags a special label. In the end, an architect decides on the final label.

Our software offers possibilities to provide an operation for mixing embedding values in accordance to the user's wish. For instance, a user provides an operator that computes average color from RGB component of both redundant faces.

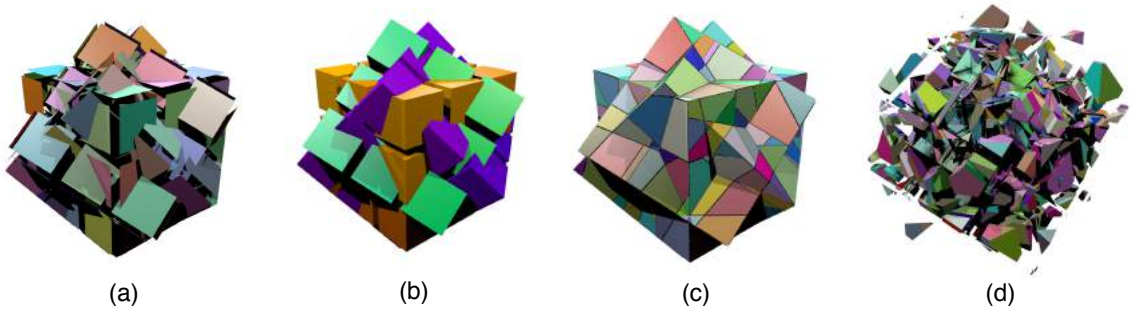


Figure 8: Global process: (a) 3 Rubik's cubes overlapped (total: 324 faces); (b) result after first topological reconstruction, 81 volumes are built; (c) result after 3D co-refinement process, all faces and edges are split along intersections, the model is composed of 3998 faces and 618 volumes; (d) exploded view of all obtained volumes.

To conclude this section, Figure 8 illustrates the final full reconstruction process. Initially, a first reconstruction is performed after importation of raw data (see Fig. 8(a-b)). Then, we exploit the topology in order to improve accelerator structure and we apply the cleaning/repairing stage for ensuring all the properties of our model (see Fig. 8(c)). Finally, the process applies a last reconstruction in order to consider new edges and faces for satisfying the partition criteria (see Fig. 8(d)).

5 RESULTS AND DISCUSSION

This section presents results of our method and to prove generality of our approach we take different meshes from various domains (CAD, lighting simulation, geology, artistic). Object examples are extracted of community meshes and a real geological data, with additional embedding when needed in accordance to business logic of their original domain. The chosen objects are illustrated in Figure 9. Each model presents particularities and highlights specific part of our method. Among all examples, we count small and big objects, and more or less correct objects. For instance, Figure 9(a) is our synthetic model because it contains few faces but a lot of intersections. The Stanford Bunny (see Fig. 9(d)) is well formed cause it has been scanned from real ceramic figurine. Whereas, Figures 9(c,f,g) are artistic creations with many unintentional errors. Figure 9(b) illustrates gears created from CAD tool but positioning of them has been done by hand and drives to numerous errors on tiny edges. Figure 9(e) is a real mesh created from noised data that induce interpenetration errors.

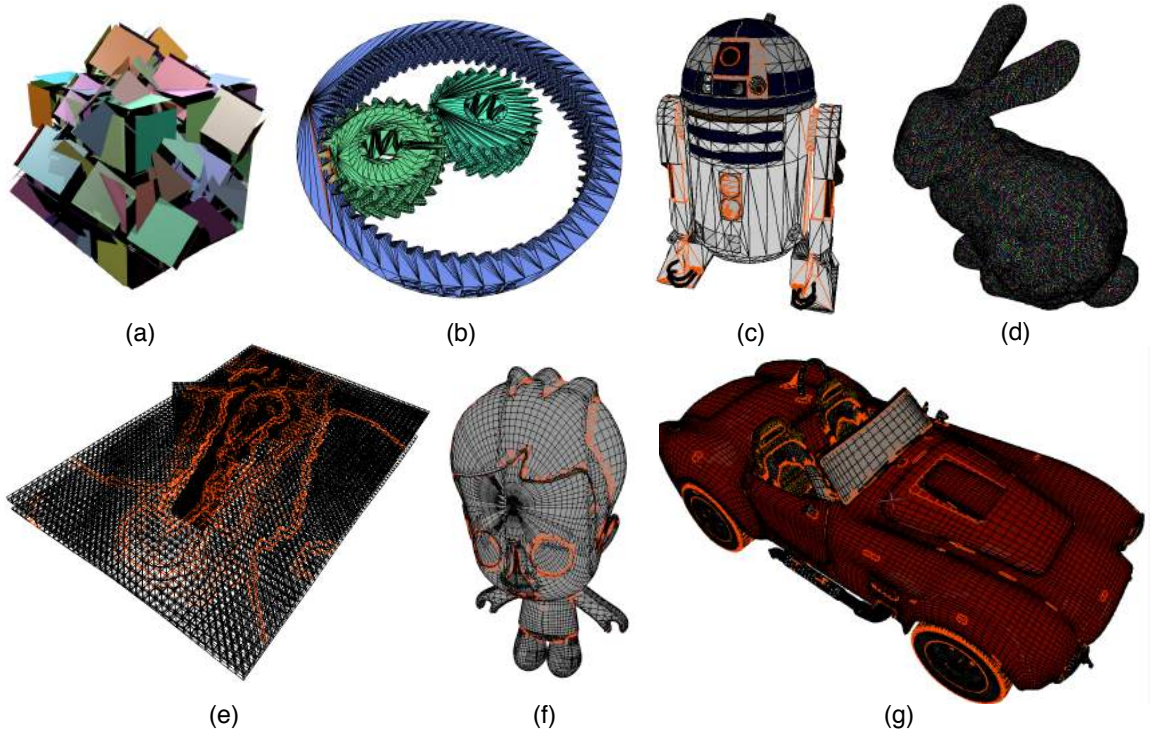


Figure 9: Used meshes for experimenting our process, (self-)intersections are underlined in red: (a) 3 intersected Rubik's cubes, many faces are intersected several times; (b) CAD gears; (c) droid; (d) Stanford Bunny; (e) geological model; (f) artistic toy; (g) fan cobra car.

Our chain process has been implemented with two different tools. The first one is Jerboa [5], a Java library dedicated to G-map. This tool allows rapid prototyping algorithms to assess their feasibility and efficiency but it does not offer good performance in terms of computation time. The second is Moka [23], a G-map library in C++ where many common operations are implemented in an efficient way. Table 1 presents execution time of our process on meshes of the Figure 9 with Moka. This study has been performed on an Intel i7-2600 with 16Go RAM under Linux OS. Regular grids are used for optimizing computation time. The purpose of this structure is to avoid unnecessary tests for remote edges/faces, since their geometrical location in the scene do not allow possible intersection.

Meshes	nb. Faces	Topo. Reconstruction	nb. Volumes Built	Co-refinement With Topo.	intersections Processed
Fig. 9(a)	324	0.02s	81	0.03s	2,674
Fig. 9(b)	40,916	3,5s	6	32s	843
Fig. 9(c)	6,872	0.4s	66	0.21s	2,165
Fig. 9(d)	69,451	3.2s	1	20s	0
Fig. 9(e)	61,720	3.5s	10	1.5s	537
Fig. 9(f)	65,439	15s	107	45s	7398
Fig. 9(g)	342,000	26s	1,740	23min12s	59,966

Table 1: Execution time for the full 3D-reconstruction and 3D-co-refinement process.

The topological reconstruction looks fast but computation time is not directly related with the number of faces. For example, Figures 9(d) and 9(f) (line 4 and 6) have different time for similar size. The explanation comes from the fact that Figure 9(d) has a simpler configuration on geometrical edge whereas Figure 9(f) has many adjacent faces to a common geometric edge. Indeed, angle arrangement process time depends on the number of faces considered around edges.

In addition, the co-refinement does not depend directly of the mesh size. For instance, line 1 and line 3 count similar faces but the co-refinement step takes more time for the first than the second one (ditto line 4 and 5). However, the time for co-refinement depends drastically on complexity of the tested mesh. The count of intersection affects co-refinement process. Nevertheless, this indicator seems not unique (e.g. line 1 counts four times more intersections than line 5) and the size of the mesh affects obviously the process.

These models has been tested on usual tools. In particular, the software MeshLab and ReMESH detect and select correctly intersected faces. And it successes to remove duplicate faces, single vertices and other errors. A first comparison allows to control that our method detects the same number of geometric inconsistencies. Although MeshLab detects these errors a little faster than our method, but it does not correct it directly. The only solution is deletion of these faces, then filling the created hole. This approach does not respond to our needs. On correct mesh as the Stanford Bunny, performance time are similar (2 seconds) and our reconstruction does not add too much additional time.

6 CONCLUSION

In this paper, we present a topological reconstruction method of volumes from a soup of polygons (from various input data source: obj, off, others geometrical format and G-map format). This framework relies on the topological structure G-map and divides reconstruction into two major steps: topological 3D angle arrangement, then cleaning/repairing operations (called co-refinement). Furthermore, the proposed angle arrangement allows to reduce number of effective test and to avoid multiple reapplication of the whole process. The co-refinement gathers all split operations in order to decrease the number of artifact and to ensure best computation on initial mesh directly. All these operations lead to satisfy a specific topological model with strong criteria that ensure high quality of the produced object. Finally, the obtained objects are formed from disjoint volumes where all topological and embedding information are consistent. And they may be directly used for various simulations algorithms.

In future work, we would add an automatic detection and correction of holes (topology helps us with the notion of dangling face) to ensure the common spatial volume closure. Furthermore, the current process may *oversplit* some faces, especially during the 2-co-refinement. The process could add a topological simplification to avoid this trouble and/or to reduce the count of initial primitive. Finally, the last (and complex) feature consists in detecting inclusion of volumes (without edges or faces connection).

ACKNOWLEDGMENTS

This research is a part of collaboration between Total S.A., the company Geosiris and XLim Laboratory. The project proposes a novel process for increasing mesh quality of subsurface in order to accurate fluid flow simulation for detecting petroleum reservoir. We would like to thank Total S.A. and Geosiris for providing us gladly real geological data from the Alwyn site located in the Northwest Europe.

ORCID

Hakim M. Belhaouari, http://orcid.org/0000-0003-4454-7756
Sebastien M. Horna, http://orcid.org/0000-0002-9170-6513

REFERENCES

[1] Adelson-Velskii, G.M.; Landis, E.M.: An algorithm for the organization of information. Soviet Mathematics Doklady, 3(2), 263–266, 1962.

[2] Attene, M.: A lightweight approach to repairing digitized polygon meshes. The visual computer, 26(11), 1393–1406, 2010. http://doi.org/10.1007/s00371-010-0416-3.

[3] Attene, M.; Campen, M.; Kobbelt, L.: Polygon mesh repairing. ACM Computing Surveys, 45(2), 15:1–15:33, 2013. http://doi.org/10.1145/2431211.2431214.

[4] Attene, M.; Falcidieno, B.: Remesh: An interactive environment to edit and repair triangle meshes. In IEEE International Conference on Shape Modeling and Applications (SMI 2006), 41–41. IEEE, 2006. http://doi.org/10.1109/SMI.2006.29.

[5] Belhaouari, H.; Arnould, A.; Le Gall, P.; Bellet, T.: JERBOA: A graph transformation library for topology-based geometric modeling. In 7th International Conference on Graph Transformation (ICGT 2014), vol. 8571 of LNCS. Springer, York, UK, 2014. http://doi.org/10.1007/978-3-319-09108-2_18.

[6] Bischoff, S.; Kobbelt, L.: Structure preserving cad model repair. Computer Graphics Forum, 24(3), 527–536, 2005. http://doi.org/10.1111/j.1467-8659.2005.00878.x.

[7] Bischoff, S.; Pavic, D.; Kobbelt, L.: Automatic restoration of polygon models. ACM Transactions on Graphics, 24(4), 1332–1352, 2005. http://doi.org/10.1145/1095878.1095883.

[8] Botsch, M.; Kobbelt, L.; Pauly, M.; Alliez, P.; Levy, B.: Polygon Mesh Processing. AK Peters, 2010. ISBN 978-1-56881-426-1.

[9] Brandel, S.; Schneider, S.; Perrin, M.; Guiard, N.; Rainaud, J.F.; Lienhard, P.; Bertrand, Y.: Automatic building of structured geological models. Journal of Computing and Information Science in Engineering, 5(2), 138–148, 2005. http://doi.org/10.1115/1.1884145.

[10] Brisson, E.: Representing geometric structures in d dimensions: Topology and order. Discrete and Computational Geometry, 9(4), 387–426, 1993. http://doi.org/10.1007/BF02189330.

[11] Cazier, D.; Dufourd, J.F.: A formal specification of geometric refinements. The Visual Computer, 15(6), 279–301, 1999. http://doi.org/10.1007/s003710050178.

[12] Cignoni, P.; Callieri, M.; Corsini, M.; Dellepiane, M.; Ganovelli, F.; Ranzuglia, G.: MeshLab: an Open-Source Mesh Processing Tool. In V. Scarano; R.D. Chiara; U. Erra, eds., Eurographics Italian Chapter

Conference. The Eurographics Association, 2008. ISBN 978-3-905673-68-5. http://doi.org/10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136.

[13] Crozet, S.; Léon, J.C.; Merlhiot, X.: Fast computation of local minimal distances between cad models for dynamics simulation. Computer-Aided Design and Applications, 15(4), 585–600, 2018. http://doi.org/10.1080/16864360.2017.1419646.

[14] Diakité, A.A.; Damiand, G.; Van Maercke, D.: Topological Reconstruction of Complex 3D Buildings and Automatic Extraction of Levels of Detail. In Eurographics Workshop on Urban Data Modelling and Visualisation. Eurographics Association, Strasbourg, France, 2014.

[15] Gueziec, A.; Taubin, G.; Lazarus, F.; Horn, W.: Converting sets of polygons to manifold surfaces by cutting and stitching, 383–390. Visualization 1998. IEEE Proceedings, 1998. http://doi.org/10.1109/VISUAL.1998.745327.

[16] Guiard, N.: Construction de modeles geologique structuraux. Ph.d. thesis, Ecole des Mines, FR, 2006.

[17] Hachenberger, P.; Kettner, L.; Mehlhorn, K.: Boolean operations on 3d selective nef complexes: Data structure, algorithms, optimized implementation and experiments. Computational Geometry, 38(1), 64–99, 2007. http://doi.org/10.1016/j.comgeo.2006.11.009.

[18] Horna, S.; Damiand, G.; Meneveaux, D.; Bertrand, Y.: Building 3D indoor scenes topology from 2D architectural plans. In International Conference on Computer Graphics Theory and Applications (GRAPP). Spain, 2007.

[19] Lienhardt, P.: N-dimensional generalized combinatorial maps and cellular quasi-manifolds. International Journal of Computational Geometry And Applications, 4(3), 275–324, 1994. http://doi.org/10.1142/S0218195994000173.

[20] Murali, T.M.; Funkhouser, T.A.: Consistent solid and boundary representations from arbitrary polygonal data. In Proceedings of the 1997 Symposium on Interactive 3D Graphics, 155–163, 1997. http://doi.org/10.1145/253284.253326.

[21] Perrin, M.: Shared Earth Modeling: Knowledge Driven Solutions for Building and Managing Subsurface 3D Geological Models. Technip, 2013. https://hal-mines-paristech.archives-ouvertes.fr/hal-01518015.

[22] Skorkovská, V.; Kolingerová, I.; Benes, B.: A simple and robust approach to computation of meshes intersection. In Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, 175–182. INSTICC, SciTePress, 2018. ISBN 978-989-758-287-5. http://doi.org/10.5220/0006538401750182.

[23] Vidil, F.; Damiand, G.: Moka, 2003. https://xlim-sic.labo.univ-poitiers.fr/logiciels/MoKa.

[24] Zhou, Q.; Grinspun, E.; Zorin, D.; Jacobson, A.: Mesh arrangements for solid geometry. ACM Transactions on Graphics, 35(4), 39:1–39:15, 2016. http://doi.org/10.1145/2897824.2925901.