Algorithm for fixing singular defects of polygon meshes based on Half-Edge Data structure

Danylo Shovhelia[0000-0003-0571-0899], Natalya Sokolova✉[0000-0002-2207-1884]

Oles Honchar Dnipro National University, Gagarina av., 72, Dnipro, 49005, Ukraine
dshovhelia@gmail.com, n.olegowna@gmail.com

Abstract. The problem of the occurrence of singular defects in the creation of 3D models in polygonal meshes due to the peculiarities of the Half-Edge Data structure is considered. An algorithm for detecting defects in a model based on half-edge structures and mesh recovery is proposed. Software implementation of the method allows to create a model based on the source data without the formation of new defects. Appropriate tools have been developed for benchmarking, and results are presented.

Keywords. 3D printing, 3D models, polygonal mesh, singular defect, Half-Edge Data Structure, non-manifold elements.

1 Introduction

Additive Technologies (for example, three-dimensional or 3D printing) is a form of additive manufacturing technology where a three-dimensional object is created by imposing consecutive layers of material (printing, growing) according to the digital model. Digital 3D models are key components of this technology, and each specific program that deals with a 3D geometry has its quality requirements that limit the class of acceptable and supported models. In most cases, visualization is just one of many steps that make up the life cycle of a digital 3D model. 3D models need to be analyzed and processed using advanced algorithms, which usually have strict requirements for the quality and integrity of their input data. Adapting imperfect 3D models to these requirements is important.

Polygonal and triangular meshes are now a de facto standard in most 3D modeling areas. Polygonal meshes are directly supported by accelerated graphics equipment, which facilitated their expansion and use [1-4]. Most graphic formats commonly used for 3D model sharing (OFF, VRML, PLY) encode the surface mesh using indexed sets of faces. However, these file formats do not guarantee the proper simplicity complex, since they can easily encode non-manifolds and/or non-oriented polygons, isolated elements, and several other elements that are often the source of problems.

2 Formal problem statement

3D printing is one of the fastest growing technologies that captured many spheres of human life and has become their indispensable core, in particular, such as architecture, medicine and mechanical engineering. Models is a basis for 3D printing. Primitive models cannot display the whole variety of printed products, and modeling of more complex models based on a simplest one can cause a number of defects. The latter, may lead to models that could not exist in the real world. The urgent issue is the timely detection of such defects, development of algorithms for model fixation, methods restoration of polygonal meshes and their software implementation in existing tools.

3 Literature review

3.1 Defects in the representation of three-dimensional models

Specialized computer-aided design (CAD) systems are used to model the objects, which are based on a particular internal structure of the model representation. Move an object from one system to another or changing their representation can be completely unpredictable, which leads to the appearance of topological defects (defects of local connections) and consequently to geometric defects [1,5], without any possibility of their identification. Three-dimensional scanning of real-world objects can be accompanied by many defects, which are for the most part classified as geometric [6]. Such situations are unacceptable because they lead to a violation of the integrity of the printed object.

1. Local connectivity where a set of triangular polygons encoded in a file does not form a combinatorial manifold simplex complex. Correcting such problems depends largely on learning the available input geometry and most often involves creating new geometry. The following defects include [1,5]:

Isolated Vertices where a vertex that is not an element of any other simplex of a mesh. Such vertices can often be simply ignored and if necessary, be removed very easily. Many existing tools provide manual or automatic procedures for this.

Dangling Edges where edges that do not have incident triangles. A common strategy is to simply ignore or remove the edges. Edges may also be used by some approaches as a useful source of information about the assumed basic geometry to recover.

Singular (*complex, non-manifold*) *Edges* where more than two polygons have a common edge. Since the condition of the combinatorial set is required in several application contexts, there is a strong call for solutions that transform such a mesh into a combinatorial manifold. This problem is not as easy to solve as in a case of isolated vertices or edges because of the inherent ambiguity.

Singular Vertices where the vertex is not manifold in the topology of the abstract simplicial complex. Detecting such vertices is more difficult than detecting singular

edges. If it is very important to calculate the number of incident triangles for the edges, then it is necessary to calculate the number of connected components in the vicinity for the vertices. Solutions to this kind of problem are usually based on the duplication of one vertex, after which each component of the neighborhood should be reassigned to one of the copies.

2. Global Topology (general topological characteristics of the surface, including the number connected components, the genus, the number of cavities and orientation) [1,5]:

Topological Noise [7] represents a situation when reconstructing a surface, starting with point clouds or when removing an isosurface from 3D images, tiny "tunnels" that were not present in the original object are introduced into the constructed digital model due to the effects of overlay or discrete noise basic data.

Inconsistent Orientation where the order of the sequences of vertex indices indicates the orientation of the polygon. To ensure full visibility across all imaging systems, it is necessary to provide a unique consistent orientation for all mesh polygons by selecting a seed surface and extending the orientation to adjacent faces. However, some configurations are essentially not oriented, which means that, the system must necessarily cut the surface for consistent orientation.

3. Geometric defects where the geometric realization of an abstract simplicial complex is considered. Geometric defects also relate to the characterized position of the vertices. This family is further complicated by the need to handle continuous coordinates and intersections, which (for efficiency reasons) must be expressed and processed with finite precision arithmetic, which usually leads to reliability problems. The geometric effects include the following [1,5]:

Surface Holes and Gaps. When scanning an object using standard scanners and designing a surface using standard CAD systems, adjacent patches are separated by unwanted holes due to the features of the equipment and displacement of tessellation patches, which must be compensated. Such steps are known as closing the gap (the area between two triangulated surface spots that must be permanently connected but not associated with the gap) and filling the holes (undesirably missing part of the surface in the triangulated patch). The main difference between the two cases is the connection of their boundaries. The boundary of the gap usually consists of two (or more) disconnected chains of edges. The border of the hole usually consists of one or more closed edge loops.

Degenerate Elements. Degenerate triangles are zero-area triangles, and they are a source of many problems for many applications, since many useful objects (ordinary vectors, circles, bar-centric coordinates) cannot be calculated on such triangles. Applications that specialize in calculating the above objects (for example, finite element analysis or Delaunay refinement implemented in the freely available Triangle package [5,8]) fail when the mesh contains degenerate triangles or becomes unstable when it contains nearly degenerate triangles.

Self-intersections. In several application contexts, it is assumed that the input mesh represents the boundary of some solid volume, and therefore cannot self-intersect. Although it is relatively easy to test the mesh for self-intersections, solving them is a difficult problem due to inherent ambiguities. Self-intersecting meshes are typically generated by multi-patch mosaics, mesh deformation, assembly of many parts without care, or by combining trays reconstructed from partial scanning of a 3D object. Due to ambiguity, there is no common strategy for solving this problem.

Sharp Feature Chamfering. Most restoration and contouring methods limit each sample or vertex to a specific line or curve, the position of which is completely determined by the pre-set pattern. In most cases, such a pattern cannot be adjusted so that it coincides with the sharp edges and angles of the model and, therefore, virtually none of the specimens rests on such sharp features. This leads to the imposition of artifacts, where sharp edges and angles of the original shape are removed by the sampling process and replaced by unevenly triangulated chamfers, which in turn leads to poor visualization and high distortion;

Data Noise. Each scan tool has a finite precision. Thus, the output of the sample model contains additive noise from different sources. The main problem is to remove noise while maintaining the morphology of the main sampling surface, with particular attention to high-frequency details such as angles, edges, or other sharp features.

3.2 Methods of mesh repairing

Non-manifold meshes, which contain singular edges and vertices, can be classified into two classes: those that bind so-called regular sets and those that do not have regular sets [5, 9]. The former are still well defined by the continuous volume, down to the singular contacts on the non-manifold edges and vertices. Singular edges and vertices are often deliberately created to avoid duplication of vertices and for the continued need to hold these duplicate vertices sequentially. In a general case (for example, meshes containing singular edges with an odd number of incident faces), which usually has ambiguities, specialized or global methods are required.

The two algorithms proposed by Gueziec at Al. [10, 11] transform meshes that restrict non-manifold regular sets into sets of combinatorially manifold meshes. Strictly speaking, a closed-mesh mesh can be a regular set only if it has no self-intersections. However, they are only bonded and therefore can also handle self-intersecting meshes. After identifying a singular edge having 2n incident faces, it splits into n multi-row edges having 2 faces each. Such representation of marginal diversity may still contain separate vertices that must be identified and duplicated.

Rossignac and Cardoze [12] proposed a strategy for performing the minimum number of such duplications. The approach introduces additional operations to merge the edge edges of the mesh section along singular edges or by simply joining the border (clamping) or stitching adjacent curves, and thus reducing the number of locking connected components.

Extensions of these works have been studied for higher dimensions by De Floriani et al. [13-16], who propose to retain some of the innocuous features as long as the model remains a initial quasi-manifold model, which is a weaker condition than a

manifold one. For the individual case of three dimensions, Attene [1] performed a detailed analysis of existing algorithms of restoring all types of defects for polygonal meshes, according to efficiency and proposed two algorithms for removing features from tetrahedral meshes. One algorithm offers combinatorial manifolds and the other prioritizes geometry. Any non-degenerate mesh that restricts regular recruitment can be tetrahedroned and processed.

Wang Zengbo [17] presents algorithms for the rapid recovery of triangular meshes imported from STL files.

Jixin Tan and Jianxun Chen [18] describe generalized algorithms for eliminating singular defects. The latter approach involves splitting models connected by singular vertices or complex edges into two independent objects.

However, it should be kept in mind that due to topology, a half-edge data structure, does not allow the creation of singular defects, which leads to the formation of defects of another type and the inability to determine the initial ones.

3.3 Typical structures for description three-dimensional models

Objects created using polygonal meshes should store different types of elements such as vertices, edges, faces, polygons and surfaces. Polygonal meshes can be represented in a variety of ways using different methods of storing vertices, edges, and faces. These include the following [5]:

• List of faces: description of faces is made by pointers to the list of vertices.

• Winged View: Each point of an edge points to two vertices, two faces, and four (clockwise and anti-clockwise) edges to which it belongs. Large memory is required for storing mesh description data.

• Half-edged meshes: the method is similar to a winged view, except that only half of the edge is traversed.

• Four-edged meshes that hold edges, half-edges, and vertices without any indication of polygons. Polygons are not explicitly expressed in the description, and can be calculated bypassing the structure. Memory requirements are similar to half-edge meshes.

• A table of angles that stores vertices in a predefined table, and the bypass of the table implicitly specifies polygons. Table of angels do not feed the mesh completely. Most meshes require multiple corner tables (triangles).

• Vertex description: Only vertices that point to other vertices are represented. The edge and edge information are not explicit in this view. However, the simplicity of the presentation allows to perform many effective operations over the mesh.

Each presentation has its advantages and disadvantages.

The choice of data structure is determined by the application, required performance, size of the data, and operations that to be performed. Compact, simple structures are required for hardware rendering. Low-end APIs such as DirectX and OpenGL usually include a table of angles (triangles).

Half-Edge Data structure is based on a principle of splitting an edge into two multidirectional halves. Each of these halves is a basic element and stores a topology in itself, which allows to manipulate a polygonal mesh. According to the definition of

this structure, a model should not contain singular vertices and/or complex edges, since creating a connection between such elements is impossible with a basic interpretation [5].

Singular defects often occur in the following situations:

1. The process of folding vertices.
2. Boolean operations on objects.
3. Errors of tessellation algorithms (faceting of a solid-state model).
4. Changing the structure of the presentation from a more primitive (vertex representation) to a more complex (half-edge data structure).
5. Duplicate elimination in trivial data structures.

One common representation of data structures in CAD systems is the spreadsheet, which has advantages in terms of memory savings and ease of rendering. Such an implementation is presented in OpenMesh [19]. However, due to the lack of connection in such a structure, it can contain defects of any type, and attempting to create such a model in a half-edge structure can lead to its destruction. Each polygon consists of half-edges and contains so-called pointers to half-edges from the general table structure.

4 An algorithm for creating a polygon model in OpenMesh

The development team of OpenMesh uses the following algorithm to create a polygon model:

1. For a set of triple coordinates:
   1.1. Add three coordinates to the structure.
   1.2. Return the unique triangle ID.
2. From a set of identifiers to create a polygon:
   2.1. For a pair of vertex ID, check that there is a half-edge associated with that vertex, if so proceed to 2.4.
   2.2. If a half-edge has not yet been created, then create one and add it to the face.
   2.3. Select the next pair of vertices, if any, and return to item 2.1, otherwise go to item 2.7.
   2.4. Check whether a given half-edge belongs to the face. If not, return to 2.2.
   2.5. Check for the opposite half-edge of the face. If not, return to item 2.2.
   2.6. Record an attempt to attach a new polygon to a pair of connected polygons or separate it completely.
   2.7. Add the triangle ID to the data structure.
3. Return the generated polygon ID.

The disadvantage of this algorithm is that it is capable of replacing one defect with another. Instead of creating a singular edge, the resulting model will contain a hole and an island polygon. Such polygons can be evaluated as garbage and removed along with isolated vertices and hanging edges. In the end, further processing of such a model may lead to unexpected results.

It is important to consider that the sequence of polygon creation plays a large role in this algorithm.

5 Modified algorithm for model polygon creation

One of the peculiarities of implementing a half-edge data structure in Open-Mesh is the presence of edges that allow iterating over pairs of half-edges, although the edges themselves do not exist. If the general method requires counting the falling edges of a selected edge, is it possible to create an edge that will not be a structural unit of the model but will retain a link between all contenders? In other words, it is a bridge that connects all contenders for a given edge. It is an edge-bridge that stores the pointers of each half-edge attached to it.

Suppose that there is an edge between two points along which two half-edges pass (Fig. 1). In that case, the edge knows about its pair of half-edges, or a contender container consists of 2 half-edges. In this case, when trying to add one more half-edge, ie, to combine more than two polygons in one edge, the outer half-edge of the attached polygon will be added to the claim container (Fig. 2).

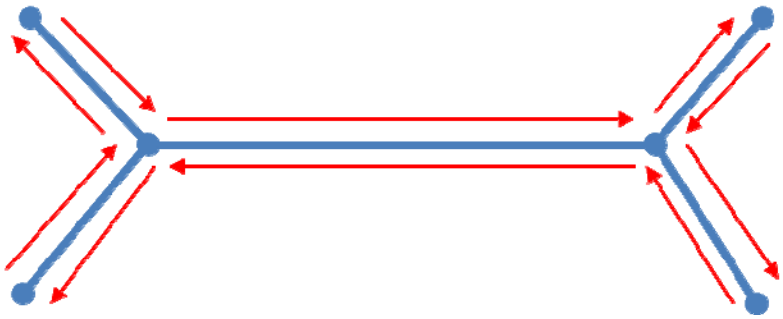


Fig. 1. Modified representation of edges

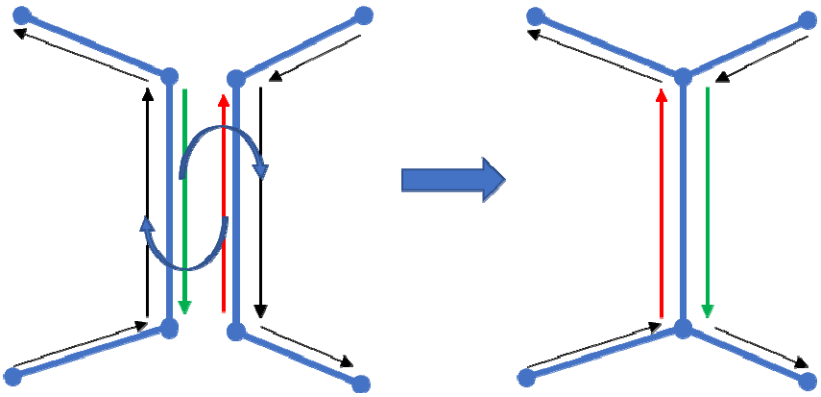


Fig. 2. Bonding of pairs

Given the fact that each half-edge can have only one partner, in that case, every 2n contender will form n pairs. From all of the above, the following changes can made to the algorithm of creating a polygon:

1. For a set of triple coordinates:
   1.1. Add three coordinates to the structure.
   1.2. Return a unique vertex ID.
2. Create a polygon from a set of vertex identifiers:
   2.1. For a pair of vertex identifiers, to check if there is a half-edge associated with that vertex, if so proceed to 2.4.
   2.2. If a half-edge has not yet been created, then create one and add it to the face.
   2.3. Select the next pair of vertices, if any, and go to item 2.1, otherwise go to item 2.7.
   2.4. Check whether the given half-edge belongs to the face: if not, go to item 2.2.
   2.5. Check if the opposite half-edge contains the faces: if not, go to item 2.2.
   2.6. Invite a container of contenders for a given half-edge.
   2.7. If the number of contenders is not even, then find a pair by method of elimination of already created pairs; otherwise, add the ID to a container. Go back to item 2.3.
   2.8. Add the triangle ID to the data structure.
3. Return the generated polygon ID.

6 Implementation of the modified algorithm

Object-oriented bridge implementation based on modified algorithm and implementation of this bridge in OpenMesh allows to eliminate defects in models based on half-edge data structure [20]. Many algorithms for preliminary analysis of meshes, at the stage of determining the integrity of the model, as well as cleaning it from debris, can lead to holes.

Let's look at the example of building an OpenMesh model. The original model was created in Solid Works 2018 SP3 (Fig.3,a). As can be seen in a figure, the model has stiffeners. The model loaded as a Triangle Soup model does not carry any information other than visual shape and cannot be used for 3D printing (Fig.3,b.).

a) b)

Fig. 3. Model # 1. a) Created in Solid Works 2018 SP3; b) 3D printing model loaded as Triangle Soup

When trying to create a model using OpenMesh, defects arise that are unacceptable with 3D printing (Fig.4).

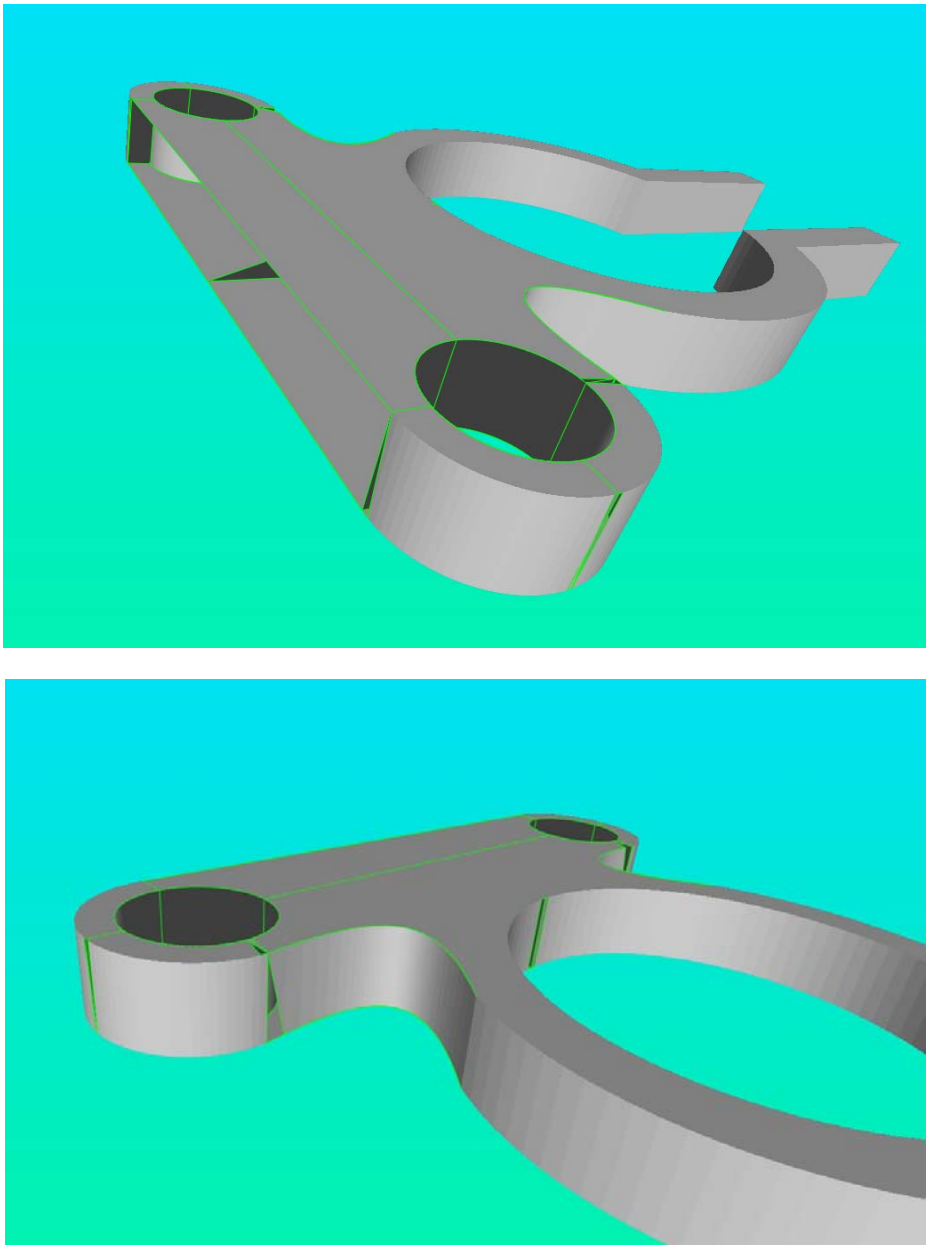


Fig. 4. A model that uses a basic Half-Edge Data Structure OpenMesh implementation.

This model is non-manifold and contains two types of defects (Singular Edges & Singular Vertices) created by Stiffener. The holes are highlighted in green (determined by the boundary edges). Because the base implementation does not support communication between more than two faces through one edge, all subsequent "contenders" for this edge will be disconnected with duplication of the corresponding points. The final analysis of a loaded model shows that the number of objects is greater than one, which is also inadmissible in some environments.

Using the modified algorithm based on the preservation of the connection between non-manifold elements gives the following result (Fig.5):

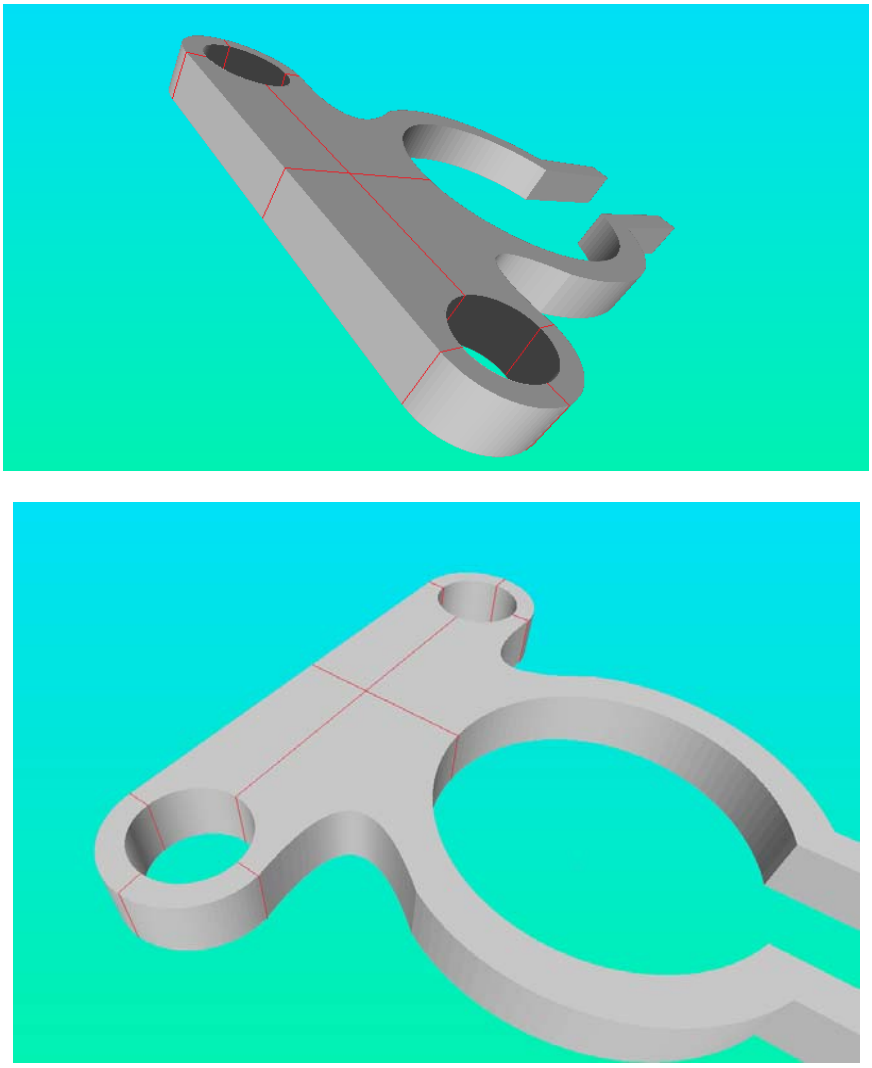


Fig. 5. Implementation of the approach with keeping the connection between non-manifold elements (Singular Edges are highlighted in red)

Another illustration compare the implementation of basic and modified algorithms.

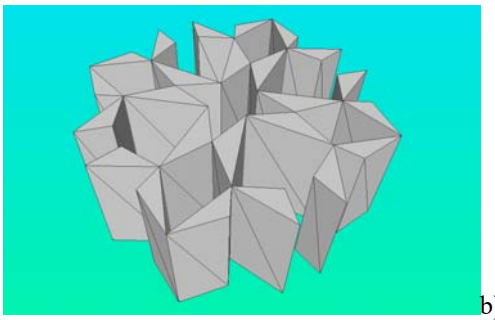
 a)  b)

Fig. 6. Model # 2. a) Created in Solid Works 2018 SP3; b) Model for 3D printing, loaded as a Triangle Soup.

This model consists of 24 triangular prisms. In the absence of communication between all the elements, it can be exploded into 24 separate parts respectively (Fig.7).

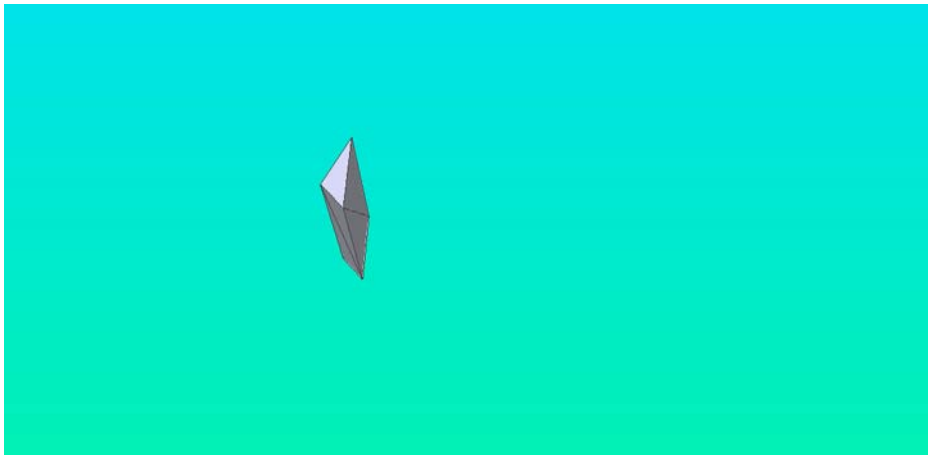


Fig. 7. Splitting the model into separate prisms in absence of communication between the elements

One solution to this problem is to replace singular defects with separate faces. After the orphaned triangles are removed, the result shown in Fig.8.

Based on the specifics of the model, about 1/8 faces will be removed. After loading this kind of object, it becomes impossible to trace the original source of data loss.

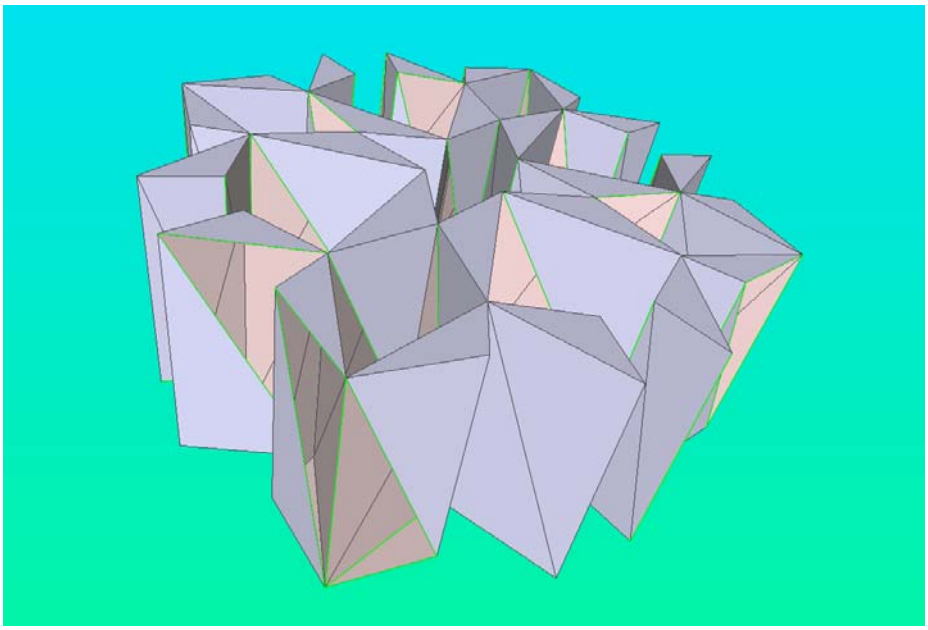


Fig. 8. Substitution of Singular Defects by Separate Edges (highlighted in green the boundary edges)

Using the algorithm of preserving the connection between the non-manifold faces and the software implementation of the bridge, the following results (Fig.9):

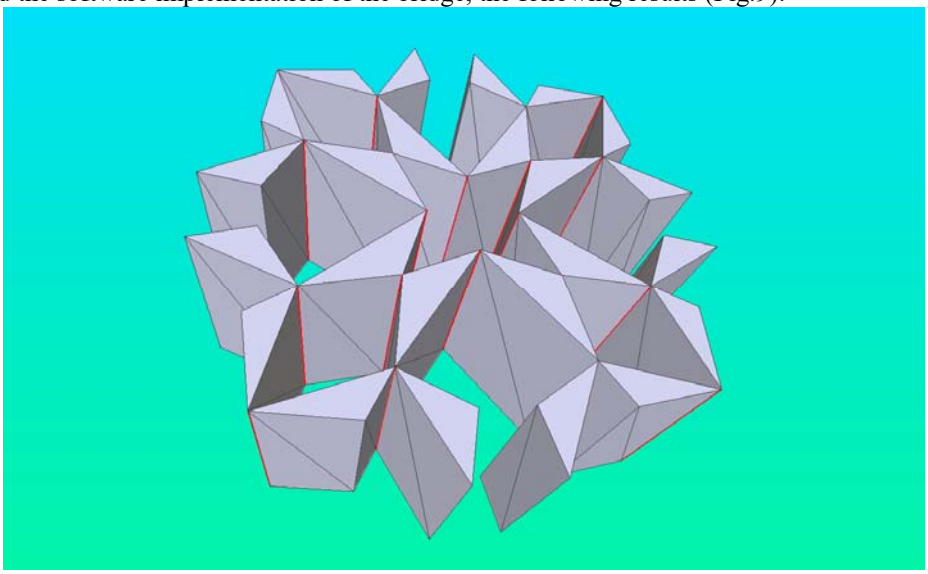


Fig. 9. Removal of singular defects by maintaining the connection between non-manifold faces (singular edges are red highlighted)

Keeping the connection between singular elements allows to load the original data without loss, as well as provide the necessary information to determine the defects of non-manifold edges and vertices.

7 Conclusion

A basic algorithm of the polygon creation implemented in OpenMesh leads to the destruction of the model if there is an attempt to create singular elements.

The newly proposed modified algorithm allows the connection between singular elements to be identified, and the overall structure of the model is preserve.

The advantage of the proposed algorithm is the simplicity and expansion of mesh recovery capabilities based on the earlier discussed algorithms.

The result of the proposed approach is the preservation of the connection in the topology during the formation of defects on polygonal meshes, which will enable detection and will make it possible to carry out various kinds of manipulations at the same time.

The disadvantage of this approach is in high memory requirements for storing links.

Further research will be devoted to improvement of the algorithm to ensure communication between all topological elements and its software implementation.

References

1. Atenne, M., Campen, M., Kobbelt, L.: Polygon Mesh Repairing: An Application Perspective : ACM Computing Surveys (scheduled to appear), 45(2):1-38. (2013). doi:10.1145/2431211.2431214
2. Lyon, M., Campen, M., Bommes, D., Kobbelt, L.: Parametrization Quantization with Free Boundariesfor Trimmed Quad Meshing. ACM Transactions on Graphics (TOG), Vol. 38, Issue 4, Article No.: 51, pp 1–14 (2019). doi:10.1145/3306346.3323019 51:1-51:14.
3. Trettner1and, P., Kobbelt, L. Fast and Robust QEF Minimization using Probabilistic Quadrics. EUROGRAPHICS 2020 Volume 39 , Number 2 (2020).
4. Schmidt, P., Born, J., Campen, M., Kobbelt, L.: Distortion-Minimizing Injective Maps Between Surfaces ACM Trans. Graph., Vol. 38, No. 6, Art. No.: 156, pp 1–15 (2019). doi:10.1145/3355089.3356519
5. Botsch, M., Kobbelt, L. Pauly, M., Alliez,P., L´evy B.: Polygon Mesh A K Peters, Ltd. Natick, Massachusetts, (2010)
6. Ju, T.: Fixing Geometric Errors on Polygonal Models: A Survey. J. Comput. Sci. Technol. 24, 19–29 (2009). doi:10.1007/s11390-009-9206-7
7. Guskov, I., Wood, Z. 2001. Topological noise removal. GI '01: Proceedings of Graphics Interface 2001, pp.19–26, (2001). doi: 10.20380/GI2001.03
8. Shewchuk,J.R. Triangle:Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. WACG 1996: Applied Computational Geometry Towards Geometric Engineering pp 203-222 (1996). doi:10.1007/BFb0014497
9. Ying, L., Zorin, D.N.: Nonmanifold subdivision. VIS '01: Proceedings of the conference on Visualization, pp.325–332, (2001)

10. Gueziec, A., Taubin, G., Lazarus, F., Horn, B.: Cutting and Stitching: Converting Sets of Polygons to Manifold Surfaces. IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS, Vol. 7, NO. 2,pp. 136-151(2001)
11. Gueziec, A., Taubin, G., Lazarus, F., Horn, B.: Converting sets of polygons to manifold surfaces by cutting and stitching. SIGGRAPH '98: ACM SIGGRAPH 98 (1998). doi:10.1145/280953.281628
12. Rossignac, J., Cardoze, D.: Matchmaker: manifold BReps for non-manifold r-sets. SMA '99: Proceedings of the fifth ACM symposium on Solid modeling and applications, pp. 31–41 (1999). doi:10.1145/304012.304016
13. De Floriani, L., Hui, A.: A Scalable Data Structure for Three-dimensional Non-manifold Objects. In: Proc, of the Symp. on Geom. Proc., pp. 72-82 (2003)
14. De Floriani, L., Magillo, P., Puppo, E., Sobrero, D.: A Multi-resolution Topological Representation for Non-manifold Meshes. CAD Jour. 36(2), pp.141-159 (2004). doi:10.1145/566282.566307
15. De Floriani, L., Greenfieldboyce, D., Hui, A.: A data structure for non-manifold simplicial d-complexes. SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing, pp. 83–92 (2004). doi:10.1145/1057432.1057444
16. De Floriani, L., Hui, A.: Data Structures for Simplicial Complexes: an Analysis and a Comparison. In: Proc, of the Symp. on Geom. Proc., pp. 119-128 (2005)
17. Zengbo, W.: Fast topological reconstruction algorithm for a STL file. Journal of Computer Applications, 34(9), pp.2720—2724 (2014).
18. Tan, J., Chen, J.: Research and Application on Model Repairing Algorithm of 3D Modelling Technology. 6th International Conference on Mechatronics, Computer and Education Informationization (MCEI 2016). doi: 10.2991/mcei-16.2016.48
19. OpenMesh https://www.graphics.rwth-aachen.de/software/openmesh/
20. Shovgelia, D.G., Sokolova, N.O.: Obnaruzhenie defektov v trekhmernykh geometricheskikh modelyakh, predstavlennykh na osnove Half-Edge Data Structure. Prykladni Pytannia Matematychnoho Modeliuvannia 1(2), pp.155-161 (2019). doi: https://doi.org/10.32782/2618-0340-2019-3-14