Asesorías y Tutorías para la Investigación Científica en la Educación Puig-Salabarría S.C.
José María Pino Suárez 400-2 esq a Lerdo de Tejada. Toluca, Estado de México. 7223898475
RFC: ATI120618V12

Revista Dilemas Contemporáneos: Educación, Política y Valores.
http://www.dilemascontemporaneoseducacionpoliticayvalores.com/
Año: VI Número: Edición Especial Artículo no.:94 Período: Diciembre 2018.

TÍTULO: Formación del Modelo Poligonal Topológicamente Conectado en el software de La Leva para Máquinas Cnc.

AUTORES:

1. Rustam R. Miftakhov.

2. Evgenij v. Zubkov.

RESUMEN. Este documento describe un problema real del campo del modelado 3d en sistemas automatizados para la preparación de programas de control para máquinas CNC, ampliamente utilizado en empresas de construcción de maquinaria. La eficiencia del sistema CAM se mejoró al garantizar la exactitud de la topología y la calidad de los modelos poligonales utilizados. El algoritmo general de la solución consta de nueve etapas para garantizar la flexibilidad de diseño y depuración. Se realizó un experimento con varios modelos poligonales, que confirmó la corrección y eficiencia del algoritmo desarrollado. El algoritmo propuesto tiene un carácter universal desde el punto de vista de su uso en el modelado 3d.

PALABRAS CLAVES: algoritmo, modelo poligonal, modelado 3d, máquinas de construcción, máquinas CNC.

TITLE: Formation of the Topologically Connected Polygonal Model in CAM software for CNC machines.

AUTHORS:

1. Rustam R. Miftakhov.

2. Evgenij V. Zubkov.

ABSTRACT: This document describes a real problem in the field of 3d modeling in automated systems for the preparation of control programs for CNC machines, widely used in machine building companies. The efficiency of the CAM system was improved by guaranteeing the accuracy of the topology and the quality of the polygonal models used. The general algorithm of the solution consists of nine stages to guarantee the flexibility of design and debugging. An experiment was carried out with several polygonal models, which confirmed the correction and efficiency of the developed algorithm. The proposed algorithm has a universal character from the point of view of its use in 3d modeling.

KEY WORDS: algorithm, polygonal model, 3D modeling, machine building, CNC machines.

INTRODUCTION.

The Russian company SPRUT-Technology is engaged in the development and implementation of systems in the field of production automation. One of the developed software products of this company is the first in the country system for the preparation of control programs for CNC machines on personal computers and the world's first CAM-system – “SprutCAM” [Information about “SprutCAM”. (Accessed 11.06.2018)], available: http://www.sprut.ru/products-and-solutions/products/sprutcam/?tab=137

The main target group of the system is small and medium business enterprises. During the analysis, it was found out, that 3D models of parts, for which the NC program is preparing, are not always topologically connected and correct. In the “SprutCAM” system, there is an algorithm that solves this problem, but its speed and quality of the output is not always enough when working with large and complex models.

In this way, it was necessary to improve the algorithm of formation of topologically connected polygonal model in the “SprutCAM” system.

DEVELOPMENT.

Methods.

Main definitions.

Polygonal model (mesh). It is a collection of vertices, edges and faces that defines the shape of a polyhedral object in 3D computer graphics and solid modeling. The faces usually consist of triangles (triangle mesh), quadrilaterals, or other simple convex polygons [The definition of “Polygon mesh”, available: https://en.wikipedia.org/wiki/Polygon_mesh (Accessed: 11.06.2018)].

Topology of the polygonal model. It is a certain, specific disposition of vertices, edges and polygons.

Correct topology of the polygonal model. It is such disposition of vertices, edges and polygons, which satisfies the rules and principles of the correct topology of the model [T. Ju, 2009].

The basic rules of correct topology are:

R1. Amount of edges of each polygon must be equal to the amount of its neighbouring polygons (“neighbours”).

R2. Each edge of a polygon must have only one paired edge (“pair”) that belongs to one of its neighbours.

R3. The order of vertices bypass and the normal vector of each polygon must coincide with the order of vertices bypass and the normal vector of each neighbour.

R4. Each polygon must not intersect any other polygon.

Neighboring polygons (neighbors). It is a polygons, which have a common edge and the same order of vertices bypass. The order of vertices bypass determines the direction of normal vector.

Paired edges (pairs). It is edges, which belongs to neighbours.

Description of the algorithm.

This article describes the algorithm, which allows restoring the correct topology of the model, in accordance with the resulted rules of the correct topology. It consists of nine stages.

Stage 1. Caching polygons.

At this stage, a cache of polygons is created [A. Skvortsov, 2002]. Before its creation, the future dimensions and number of cells are calculated [R. Miftakhov, E. Zubkov, 2017]. After creation, each model polygon is added to the cache. Using the cache allows to speed up all following stages of the algorithm.

Stage 2. Primary formation of pairs.

At this stage, the neighbours are searched for each polygon. It should be kept in mind that the data structure allows you to store no more than one neighbour for each edge. The order of actions is:

1. Execute a cycle for all polygons in the model.
2. Using the cache, find all nearest polygons for each edge of the current polygon.
3. Execute a cycle for all found polygons.
4. Check each edge of the found polygon with the current edge. If they are pairs, then put into the memory the checked edge and break the cycle 3.

After this stage, you can determine polygons that do not correspond to the first rule of the correct topology R1 (“polygons not with all neighbours”).

Stage 3. Stitching polygons.

Stitching. It is a specific action, which is executing for polygons not with all neighbours to satisfy the rule R1; i.e., to close possible holes in the model [M. Wagner, U. Labsik, G. Greiner, 2003]. The efficiency of closing holes depends on the correct choice of tolerance [S. Bischoff, L. Kobbelt, 2005]. This stage consists of two sub-stages:

3.1 Stitching vertices.

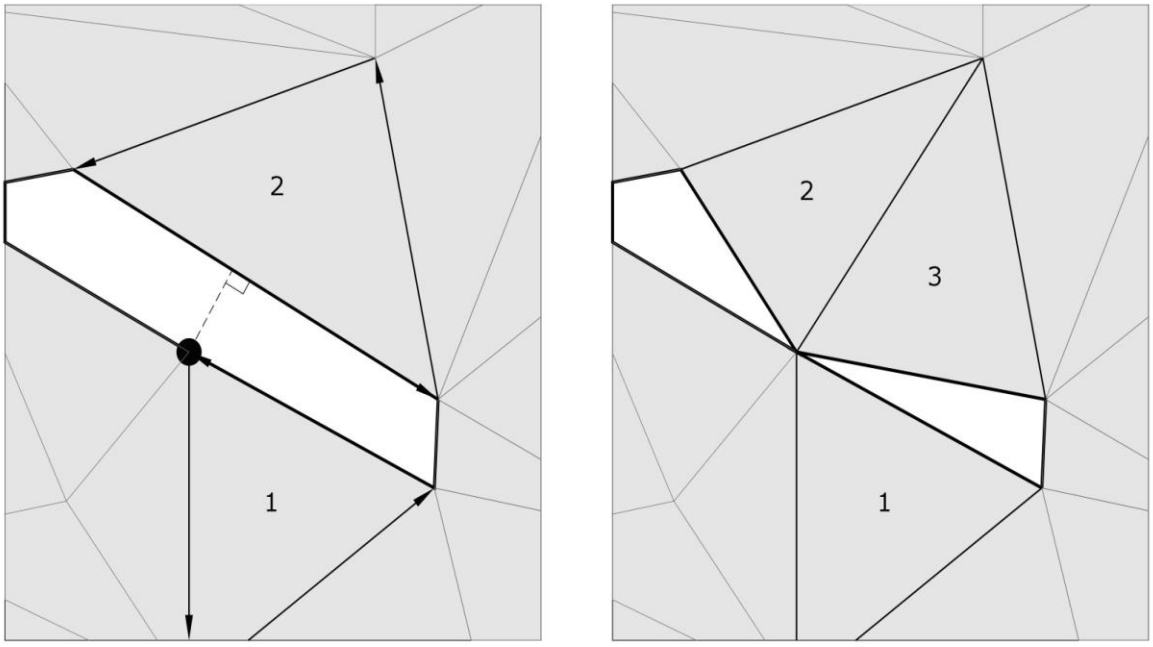
1. Execute a cycle for all polygons not with all neighbours.
2. Using the cache, for each vertex of the current polygon find all polygons, distance to which is less or equal to tolerance.
3. Execute a cycle for all edges of found polygons, which does not have a pair.
4. Check every vertex of such edges for belonging it to the region with tolerance radius around the current vertex.
5. Calculate average of vertices, which belong to the region with tolerance radius around the current vertex.
6. A new vertex is calculated from vertices, which belong to the region with tolerance radius around the current vertex, by the formula of the average.
7. After that all these vertices are replaced by the calculated.

3.2. Stitching edges.

1. Execute a cycle for all polygons not with all neighbours.
2. Using the cache, for each vertex of the current polygon find all polygons, distance to which is less or equal to tolerance.
3. Execute a cycle for all edges of found polygons, which does not have a pair.

4. Check such edges for existing of the projection of the current vertex onto the edge.
5. If the projection exists and the distance to then, edge is less than the tolerance; then, connect the current vertex and the edge according to the specific method (Fig.1).

Fig.1 Stitching edges (1 – current polygon; 2 – found polygon; bold point – current vertex; 3 – new polygon).



Stage 4. **Searching for intersections of polygons**.

At this stage, the searching for such a defect in the model and its correction is occurred [J. Wonhyung, S.Hayong, K.C. Byoung, 2004]. It allows correspond to the rule R4.

There are two types of intersecting polygons:

1. 3D intersection [T. Moller, 1997].
2. Intersecting coplanar polygons [C. Sabharwal, J. Leopold, D. McGeehan, 2013].

The solution to this problem is to split two polygons along the intersection line. For the first type of intersection 2 intersecting polygons at the input, 4 non-intersecting polygons at the output.

In the second type of intersection, there is no intersection line, since the polygons lie in the same plane. Such a problem is solved as follows:

1. Execute a cycle for all edges of the first polygon.
2. For each edge, create a plane that will pass through this edge and perpendicular to the polygons plane.
3. Intersect and split the second polygon with the created planes.

The second type of intersection is sometimes called “overlap”. It is necessary to keep in mind, that after splitting the overlapped polygons, two identical polygons can be formed, one of which should be removed. This stage can be successfully optimized using parallel computations [R. Miftakhov, E. Zubkov, 2017]. This optimization will significantly reduce the execution time of the whole algorithm, since this stage is one of the most time-consuming.

Stage 5. Creating duplicate polygons.

This stage is necessary to solve the problem of incorrect orientation of normal vectors of the neighboring polygons and correspond to the rule R3. The duplicate polygon is completely identical to the existing polygon in the model, however it has an opposite order of vertices bypass and normal with the opposite direction. Hence, surfaces that have the opposite direction of the normal will be successfully corrected.

Stage 6. Searching for all pairs.

At this stage, all possible pairs are searched for all edges in the model. Due to this stage, it becomes possible to choose the next polygon (“transition”), when forming solid models in the following stages.

The difference from the first stage is that all possible pairs for each edge are put into the memory, not just the first pair.

The differences in the algorithm are such that the cycle does not stop when a pair is found, and also a data structure is used that allows storing a lot of edges (for example, an array or a tree).

Stage 7. Searching for connected components (shells).

The task of this stage is very similar to the task of finding connected components in a graph, therefore this name was chosen.

Connected component of an graph. It is a subgraph in which any two vertices are connected to each other by paths, and which is connected to no additional vertices [The definition of "Connected component" in graph theory, Available: http://enacademic.com/dic.nsf/enwiki/153930 (Accessed 11.06.2018)].

In the context of this article, instead of definition of "connected component", a definition of "shell" will be used.

Shell of the model. It is such set of model polygons that for any two polygons in the set, there is a transition, but there is no transition from the polygon of this set to the polygon not from this set. A transition is a path from one polygon to another, passing through neighboring polygons.

At the end of this stage, each polygon will contain the information about the number of the shell to which it belongs

Stage 8. Formation of the correct models.

Using information about the found pairs and shells, the formation of models is executed, which will correspond to the basic rules of the correct topology.

Polygons can have many neighbours belonging to one edge, which does not correspond to the rule of the correct topology of the model R2. For the correct resolution of such ambiguity, it is necessary to form the maximum possible contour of polygon transition (Fig.2). The order of actions is:

1. Execute a cycle for all shells.
2. Create a new model that will represent the current shell.

3. Choose the first polygon in the shell according to the following rule: its coordinates must be minimal and the normal vector is directed outwards.
4. Add the polygon to the created model (model can be used as a queue).
5. Execute a cycle for all polygons of the created model (queue).
6. Execute a cycle for all edges of polygon.
7. Choose the neighbor that makes the minimum angle with the current polygon.
8. Add the chosen polygon to the model (queue), if it has not been added yet.

At the end of cycle 5, the next shell is processed in cycle 1 and so on until all the shells are processed.

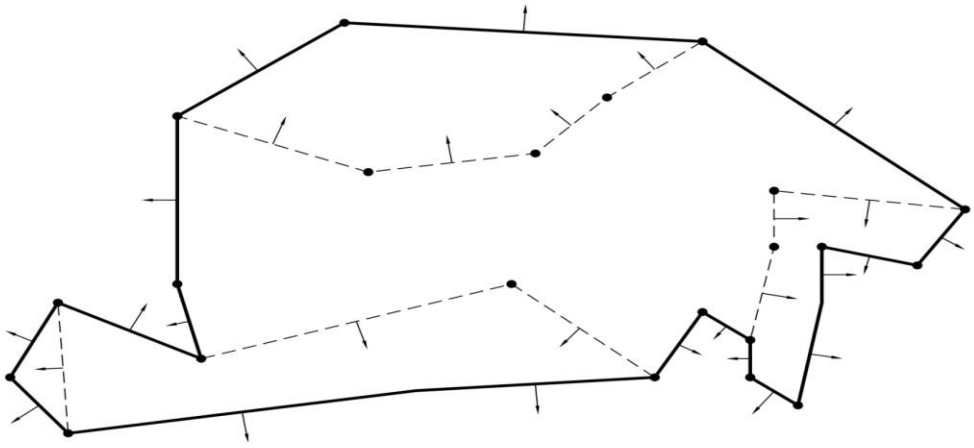
At the end of this stage, topologically correct models will be formed at the output, the number of which will correspond to the number of found shells.

In most cases, among the formed models, one has the largest amount of polygons, while the others are quite insignificant. Therefore, one model remains as a result, and insignificant models are deleted. In this case, it is necessary to take into account not only the amount of polygons, but also the size of the model. Such a problem can be solved as follows:

1. In advance, select a constant that represents the number expressed in percentages
2. Using the constant, calculate a number equal to the fraction of the total amount of polygons. It will be called the minimum allowed number of polygons.
3. Execute a cycle for all formed models.
4. Remove model, in which amount of polygons is less than the minimum allowed number of polygons.
5. Unite all remaining models.

Unification of the remaining models is necessary, because sometimes several models are equally significant and they can’t be neglected.

Fig.2 Maximum possible contour of polygon transition (shown as a solid line; dotted lines – discarded polygons; the arrows show the direction of the normals).



Results and Discussion.

To test the efficiency and reliability of the algorithm, experiments were conducted with various polygonal models. One of these models is a part for stamping bottles. The other part is a wheel (Fig.3). The algorithm described in this paper restores the correct topology of these models successfully. The statistical data is given in table 1. The execution time of the algorithm for these models is shown in table 2. These models have a high resolution, which usually affects the speed. In this algorithm, using the cache and various optimizations (for example, parallel computations), it is possible to achieve acceptable execution time results.

Fig.3. Models for experiments.

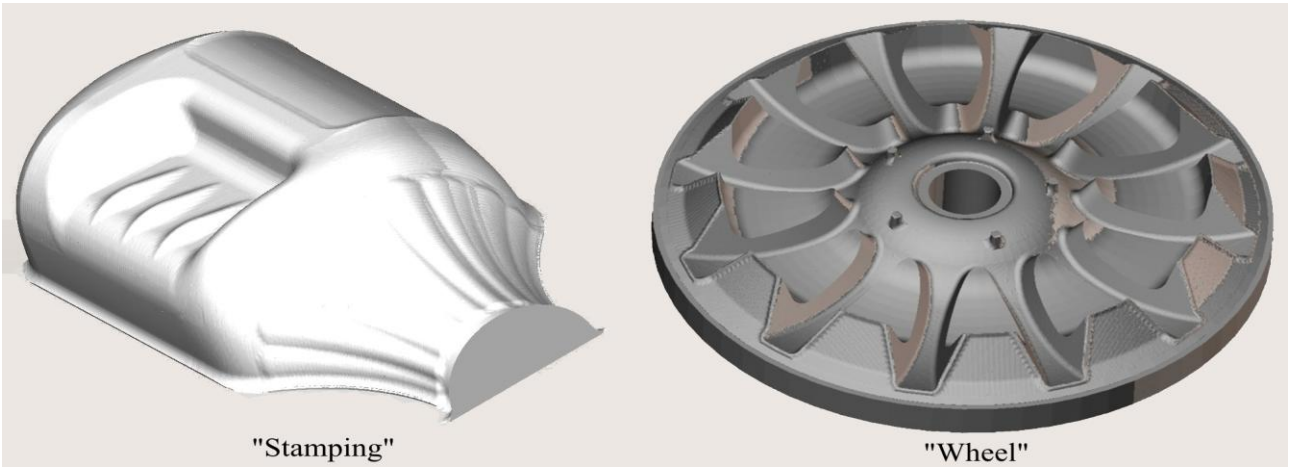


Table 1. Statistical data on models.

Name	Data	
	“Stamping”	**“Wheel”**
General information		
Amount of vertices	253720	378200
Amount of polygons	510058	756773
Size of model (X, Y, Z)	32000, 17093, 8213	32000, 32000, 6092
Tolerance	0.6	0.6
Stage 1. Caching polygons		
Cache size for polygons (X x Y x Z = amount of cells)	485 x 259 x 125 = 15701875	451 x 451 x 86 = 17492486
Stage 2. Primary formation of pairs		
Amount of edges without pairs	1054	428
Stage 3. Stitching polygons		
Amount of stitching vertices	17763	1137
Amount of removed degenerate polygons	9394	765
Amount of stitching edges	0	5
Amount of added polygons	0	5
Stage 4. Searching for intersections of polygons		
Amount of intersected polygons	699	7173
Amount of intersected coplanar polygons	779	480
Amount of removed polygons	2672	1045
Stage 5. Creating duplicate polygons		
Amount of created duplicate polygons	512235	771603
Stage 6. Searching for all pairs		
Amount of edges with one pair	2141	1286
Amount of edges with two pairs	1496098	2300680
Amount of edges with more than one pair	4648	26625
Stage 7. Searching for connected components (shells)		
Amount of shells	120	77
Stage 8. Formation of the correct models		
Amount of formed models	1	1
Amount of polygons in the resulting model	998094	1533957

Table 2. Execution time of the algorithm for models.

Name	Execution time, sec	
	“Stamping”	**“Wheel”**
Total execution time of the algorithm	8.62	14.336
Stage 1. Caching polygons	0.791	1.33
Stage 2. Primary formation of pairs	0.671	1.34
Stage 3. Stitching polygons	0.631	0.438
Stage 4. Searching for intersections of polygons	2.548	4.04
Stage 5. Creating duplicate polygons	0.495	0.886
Stage 6. Searching for all pairs	0.215	1.248
Stage 7. Searching for connected components (shells)	0.343	0.439
Stage 8. Formation of the correct models	2.974	4.615

In summary, according to the results of experiments, it was noted that this algorithm is very reliable. This means that for almost any input data, a correct model will be obtained at the output, which corresponds to the basic rules of the correct topology. It should be noted that the quality of the resulting model directly depends on the correctly chosen tolerance.

In the experiments described above, the tolerance was insufficient. You can see this if you compare the amount of polygons in the input model and in the formed. In the latter, they are approximately twice as large. It means, that there were unclosed holes in the resulting model, because of insufficient tolerance (such a model is called “thin-walled”). Nevertheless, the result corresponds to the basic rules of the correct topology.

CONCLUSIONS.

To sum up, this article describes a new algorithm for solving one of the most difficult and indefinite tasks in 3D modeling – formation of the correct topology of the polygonal model. The results of the experiment showed that the algorithm has high reliability and speed. Comparison of the algorithm

used in the “SprutCAM” system with the developed algorithm showed that the speed of the latter is on the average 2-3 times higher.

Acknowledgements.

I would like to thank my scientific advisor, Evgenij Zubkov, for his continuous support and insightful comments, and my colleagues, Vitaliy Burkov and Aleksandr Groshev, for their essential ideas, help and encouragement.

The work is performed according to the Russian Government Program of Competitive Growth of Kazan Federal University.

BIBLIOGRAPHIC REFERENCES.

[1] A. Skvortsov (2002). Delaunay triangulation and its using. Tomsk: Tomsk university publishing house, P.128.

[2] C. Sabharwal, J. Leopold, D. McGeehan (2013). Triangle-Triangle Intersection Determination and Classification to Support Qualitative Spatial Reasoning”, Polibits, vol. 48, pp. 13–22.

[3] Information about “SprutCAM”. (Accessed 11.06.2018) http://www.sprut.ru/products-and-solutions/products/sprutcam/?tab=137

[4] J. Wonhyung, S.Hayong, K.C. Byoung (2004). Self-intersection Removal in Triangular Mesh Offsetting; Computer-Aided Design & Applications, vol. 1, №. 1-4, pp. 477–484.

[5] M. Wagner, U. Labsik, G. Greiner (2003). Repairing non-manifold triangle meshes using simulated annealing; International Journal of Shape Modeling, vol. 9, №. 2, pp. 137–154.

[6] R. Miftakhov, E. Zubkov (2017). Using the caching method for accelerating polygonal models algorithms; Innovations in science and practice, №. 2, pp. 90–94, November 2017.

[7] R. Miftakhov, E. Zubkov (2017). The optimization of polygons intersection searching using parallel computing”, IITMA: materials collection,. pp. 157–159.

[8] S. Bischoff, L. Kobbelt (2005). Structure Preserving CAD Model Repair", Computer Graphics Forum, vol. 24, №. 3, pp. 527–536.

[9] T. Moller (1997). A Fast Triangle-Triangle Intersection Test; Journal of graphics tools, vol.2, №.2, pp 25-30.

[10] The definition of "Connected component" in graph theory. (Accessed 11.06.2018). http://enacademic.com/dic.nsf/enwiki/153930

[11] The definition of "Polygon mesh". (Accessed 11.06.2018). https://en.wikipedia.org/wiki/Polygon_mesh

[12] T. Ju. (2009). Fixing Geometric Errors on Polygonal Models: A Survey; Journal of Computer Science and Technology, vol. 24, №. 1, pp. 19–29.

DATA OF THE AUTHORS.

1. Rustam R. Miftakhov. Kazan Federal University. Email: raf.rust@gmail.com

2. Evgenij V. Zubkov. Kazan Federal University. Email: raf.rust@gmail.com

RECIBIDO: 2 de noviembre del 2018. **APROBADO**: 16 de noviembre del 2018.