Investigation of the Mathematical and Software Implementation of Triangulation Mesh Generation Algorithms in Relation to the Creation of a Slicer Program

A A Kholodilov¹, E V Faleeva¹, M V Kholodilova²

¹Department of "Computer Science and Computer Graphics", Far Eastern State Transport University, st. Serysheva 47, Khabarovsk 680021, Russia
²Computing Center of the Far Eastern Branch Russian Academy of Sciences, st. Kim Yoo Jong 65, Khabarovsk 680000, Russia

E-mail: kholodilovsapr@gmail.com

Abstract. The article discusses an algorithmic method for increasing the strength characteristics of products produced by additive manufacturing technologies through thickening the internal grid structure of filling the model when transferring to the control code during the operation of the slicer program. It shows the internal filling using the Voronoi grid and the "Exhaustion Algorithm" for the implementation of the internal filling of the three-dimensional model in the process of the slicer program.

1. Introduction

In the field of additive manufacturing, including FDM (Fused Deposition Modeling) technology, one of the research and production objectives is to improve the quality of three-dimensional printing and resistance to critical loads during the functional use of products made by means of three-dimensional printing. The technology of generating the internal filling is most often used according to a predetermined template selected by the user at the stage of setting the parameters. The comparative analysis of the existing types of infill shows the need to thicken the mesh of the internal filling of the model in places of probable fracture in order to increase its strength characteristics by means of dynamic recalculation of the mesh.

Thus, the aim of the work is to consider possible mathematical and software tools for implementing an adaptive approach to the specified layer thickness and to the internal filling, by using the capabilities of mathematical modeling. The slicer program developed in accordance with this principle processes a three-dimensional model, analyzing the outer contour and the inner filling of its structure, with a thickening of the mesh in places where thin-walled structures are present by using a Voronoi mesh, with the generation of g-code from an array of contours. In this case, for the implementation of the g-code on each of the generated contour layers, the generation of the mesh of the internal filling is carried out, with the recalculation of the parameters of this internal filling.
2. The problem of the internal filling of the model

Filling plays a very important role in printing a 3D model. It serves to reinforce the model and support it layer by layer. When printing, keep in mind that some fill types sacrifice model density in favor of faster printing [2, 3].

According to the results of previous studies, a pattern can be seen that in models of complex shapes, filling is not required in all areas with condensed filling. A situation arises similar to the problem of specified print layers with the same thickness - too thin filling will make the model brittle, and too thick filling is not necessary in all areas. If it is possible to change the thickening of the infill grid, depending on the shape of the model, when printing, the same problems are solved that arise when introducing a dynamic change of layers during slicing. The Voronoi grid (Figure 1) [4] is of the greatest interest in displaying such a filling grid.

![Voronoi grid](image1.png)

**Figure 1.** Voronoi grid.

Voronoi diagram is a geometric partition of a region into polygons with the following property: for any center pi of a system of points, one can specify a region of space, all points of which are closer to a given center of the system [5]. Such an area is called the Voronoi polyhedron, or the Voronoi area. The strict definition of the Voronoi polygon Ti is introduced as follows (1):

\[ T_i = \{ x \in \mathbb{R}^2 : d(x, x_i) < d(x, x_j) \ \forall j \neq i \} \] (1)

A circle is called empty if it does not contain a single point from the set P. The top of the Voronoi cell is the center of an empty circle passing through three or more points from a given set. In this case, the edge of the Voronoi cell is a straight line passing between two nodes pi and pj, such that the center of the empty circle passing through these points belongs to the straight line (Figure 2). An important property of the Voronoi diagram is its duality to the Delaunay triangulation.

![Voronoi diagram](image2.png)

**Figure 2.** Voronoi diagram: a - duality of Delaunay triangulation; b - Voronoi diagram.

In order to obtain the Delaunay triangulation, it is necessary to connect by segments all pairs of points whose Voronoi polygons have a common edge. The definition of natural neighbors for calculating the Sibson and Laplace interpolations is introduced through the Voronoi tiling and the Delaunay triangulation: for the Voronoi cell Ti, xi ∈ P, the natural neighbors for xi ∈ P are the vertices of the DeLone triangles incident to xi.

There are many methods for constructing a Voronoi diagram. The simplest and most well-known is the method of intersecting perpendiculars (half-planes), the idea of which is to alternately construct the polygons included in the diagram.

The next class of methods is based on an incremental algorithm. In the following order: a Voronoi diagram is constructed for a certain initial number of points, then the remaining points from the set are
inserted into it one by one and only a part of the diagram is rebuilt. With the development of computational geometry, a more complex algorithm was developed. This algorithm is based on the principle of "divide and conquer". The set of points $P$ is divided into two approximately equal subsets $P_1$ and $P_2$, and then the diagram for the original set $P$ is reconstructed using the "dividing chain". Despite the attractiveness of this method, it causes great difficulties in numerical implementation. In 1987, Stephen Fortune, based on the "sweep line" method, proposed an algorithm for calculating the Voronoi diagram. This method is fast in speed and relatively easy to implement.

In the process of considering the issues mentioned above, it was found that the problem of a statically specified layer thickness looks the most relevant in the framework of research, since it has not found sufficient reflection in foreign and Russian literary sources, which is associated with the novelty of the area under consideration [6]. Other issues related to the insufficient elaboration of the three-dimensional printing methodology - the problems of filling the model, including using the Voronoi grid, are directly related to the problem of a statically given layer thickness, but in terms of the scope of the problem, they are beyond the scope of this article. Accordingly, the problem of the adaptive approach to the dynamic layer thickness in three-dimensional printing and its relationship with reflection in the g-code comes to the fore. There are currently no software products that allow implementing the above approach on the software market. Accordingly, developments in this direction, which make it possible to automate the determination of the parameters of the outer wall and inner filling, are relevant. In addition, experimental data indicate that such a development, being introduced into production, will significantly save time, energy consumption, material costs and equipment resources, which indicates the economic feasibility of developing such software.

3. The applied algorithm for the implementation of internal filling in the slicer program

In the course of work on the slicer program, the "Exhaustion Algorithm" was chosen as an algorithm for implementing the principle of mesh thickening, in its three-dimensional implementation [7, 8].

This algorithm allows you to programmatically implement the construction of grids of equal size, including in arbitrarily specified areas. Let us pose the problem of constructing a triangulation mesh with the chosen average edge length - with a triangulation step equal to $h$.

The initial data of the algorithm are two data arrays:
1. Array of triangulation of the border, which is the initial front, which is a list of triangles-faces;
2. An array of auxiliary points $F$, for the formation of which it is placed in a given area in a parallelepiped, the so-called super-area, uniformly filled with auxiliary points $F$ with a step specified by formula 2.

$$h_F \approx \frac{h}{n}, n = 7 . .15$$

Taking into account this formula, the density of arrangement of auxiliary points is an order of magnitude higher than the density of triangulation nodes placed in the super-area. The coordinates of the nodes are calculated according to formula 3.

$$r_{ijk} = (ih_F, jh_F, kh_F)$$

This array, when implemented in software, does not require the allocation of a large amount of both operational and cache memory.

The set of nodes $F$ in software implementation is described using a three-dimensional array of one-bit Boolean variables. By specifying standard values, we describe the existence and deletion of a point through two variables 1) TRUE = the point exists; 2) FALSE = point deleted.

Taking into account this software implementation, let us sum up the calculation of the required amount of RAM, since a megabyte of RAM is required to store eight million $F$.

In the further operation of the algorithm, after the formation of a set of nodes $F$, all points that lie outside the specified area are removed from this set. The points that remain after this stage represent a structure in the form of a volumetric raster image of the specified area.

The set $F$ in this algorithm solves some of the problems in the implementation of the algorithm.
The algorithm under consideration is designed to build uniform grids, while the control function \( f_r(x, y, z) \) is taken as a constant equal to the volume of a regular tetrahedron with edge \( h \), taking into account the application of a 25% tolerance, and is denoted by the formula \( 0.15h^3 \). To indicate the progress of the algorithm execution, it is convenient to use the array \( F \), in this case, the ratio of the number of points \( F \) removed during the operation of the algorithm to the initial value of the number of existing points \( F \) will show to what part the area is exhausted. The remaining points will represent a kind of volumetric bitmap of the specified area. It is with the help of the set \( F \) in this algorithm and many of the problems mentioned above are solved. Since the algorithm is designed to generate uniform meshes, the control function \( f_r(x, y, z) \) is assumed to be a constant equal to \( 0.15h^3 \) (the volume of a regular tetrahedron with an edge \( h \) plus a tolerance of 25%).

Next, actions are performed according to the following algorithm:

1) A set of nodes \( G \) is formed, consisting of the front vertices. (i.e. initially all existing grid nodes are included in it); for each node \( G \) the solid angle is evaluated. This estimate is reduced to finding the number of existing (i.e., not removed) points \( F \) located at a distance of no more than \( \sqrt{3h} \) from a given node (which requires significantly less resources than direct computation of solid angles).

2) The vertex \( g_1 \in G \) with the smallest estimated value of the solid angle is found and such a face \((g_1, g, g_3)\) of the front is chosen, for which the sum of the estimates of the solid angles for \( g, g_2, g_3 \) is minimal.

3) A set \( G_1 \) of nodes \( G \) is formed, located from each of the vertices of the selected face on distance no more than \( \sqrt{3h} \) (this set may turn out to be empty). For each node \( g_4 \) from \( G_1 \), a tetrahedron \((g_1, g, g_3, g_4)\) is considered, and the tetrahedron is discarded if at least one of the following conditions is not satisfied:
   - strictly inside this tetrahedron there is not a single remote point \( F \) (the existence of such points on the faces is allowed);
   - inside this tetrahedron there are no other existing grid nodes, except for the vertices of this tetrahedron;
   - a tetrahedron is not intersected by any existing edge of the mesh (it is considered that a tetrahedron is intersected by an edge that is not an edge of this tetrahedron if it has at least one common point with this edge that is not a vertex of this tetrahedron);
   - the volume of the tetrahedron is not more than the maximum allowable (0.15\(h^3\)).

Of all the tetrahedra \((g_1, g, g_3, g_4)\), the tetrahedron of the best quality is selected and the transition to stage 5 is made; if there are no tetrahedra satisfying the indicated conditions, then the transition to stage 4 of the algorithm is carried out.

4) There is a point \( p \) inside the still unexhausted region such that:
   - tetrahedron \((g_1, g, g_3, p)\) satisfies all conditions of stage 3;
   - the ball \((p, 2h/\sqrt{n})\) does not have a single remote point \( F \) (this prevents the node \( p \) from being placed too close to the faces and vertices of existing tetrahedra). A variant of the search algorithm for the node \( p \) is considered below.

5) All vertices \( F \) that fall inside (and on the boundaries) of the formed tetrahedron are deleted. Then the front is updated according to the following scheme: each face of the formed tetrahedron is considered and:
   - if the face is a face of the front, then it is removed from the front;
   - If the face is not a front face, it is added to the front.

6) If there are still not deleted points \( F \) or the front is not empty, go to step 5, otherwise, the process is over.

Thus, the \( F \) array is used for several purposes at once: to estimate the solid angle; to control the correctness of the construction; to control the placement of new nodes.

It is also convenient to use the array \( F \) to indicate the progress of the execution. The ratio of the number of points \( F \) removed during the operation of the algorithm to the initial number of existing points \( F \) shows how much of the area is exhausted.
The question of finding the coordinates of a new node for constructing a tetrahedron at stage 4 of the algorithm is considered below. Let three nodes be given: \( g_1, g_2, g_3 \).

1) At the first step, the value \( r_c \) – the center of mass of the triangle (as the arithmetic mean of the coordinates of its nodes) and the unit normal \( \vec{n} \) to the plane of the face (through the normalized vector product) are found.

2) The first approximation is determined for the distance from \( r_c \) to the desired point \( p \), based on the maximum volume of the tetrahedron: \( V = S H / 3 \). The face area \( S \) was found in the previous step.

Based on the use of this method, a software algorithm for generating the outer contour and subsequent inner filling has been implemented, with the generation of both random filling points and ordered structures. Based on the generation of the structure on one layer, a general algorithm for recalculating layers was designed for transition from layer to layer (Figure 3).

![Figure 3. Examples of facet generation on control layers.](image)

4. Results of the implemented algorithm

Below, in Figure 4, a fragment of the result of dividing an object into layers with a variable section of the shell and a comparative analysis of the application of algorithms for static and dynamic construction of contours, implemented in the developed software package [9].

![Figure 4. Slicing of a structure with a variable cross-section: a - loaded model; b - static slicing; c - dynamic slicing with an adaptive approach.](image)

The G-code programmed for an externally loaded 3D object is saved to a file in a folder of the user's choice or, by default, in the program folder. Cross-checking with the help of third-party slicing programs of the obtained g-code is possible after the software session ends. An example of g-code verification in Gcode Analyzer software is shown in Figure 5.
According to the scheme of the algorithm of the developed slicer program, when processing the model, the following goes:

1) Implementation of a typical functional, specific to software complexes-slicers [10];
2) Recalculation of the number of contours and the distances between them for the outer contour, with an increase in the layer thickness by two times when switching from an array of contours of the same section to a variable one (possibly a break point);
3) Generation of a grid of internal filling with recalculation and condensation of places of probable break occurs, recalculation is carried out using iterative algorithms to generate a Voronoi grid according to the described algorithm;
4) The received g-code is unloaded for loading to a 3D printer (CNC device).

Thus, we can say that the Voronoi Triangulation Grid and the "Exhaustion Algorithm" are applicable to the development of a slicer program to implement the mesh of the internal filling of the model in the process of translating the three-dimensional model into a control program for a three-dimensional printer. The use of a mesh refinement allows in an automatic mode to generate an adaptive filling with a mesh refinement in places of probable fracture, which, in turn, will lead to an increase in strength characteristics.

5. References

[1] Kholodilov A A, Faleeva M V and Kholodilova M V 2020 Analysis of the Technology of Transferring a Three-Dimensional Model from Cad Format to the Control Code For 3D Printing International Multi-Conference on Industrial Engineering and Modern Technologies (FarEastCon) IEEE Xplore [Electronic source] URL: https://ieeexplore.ieee.org/document/9271241 (Reference date: 22.07.2021)

[2] Drobotov A V, Avdeev A R and Shvets A A 2015 Analysis of ways to increase the productivity of volumetric printing devices for the manufacture of assembly products Assembly in mechanical engineering, instrument-making vol 11 pp 3-6

[3] Problems of the quality of 3D printing [Electronic source] URL: https://3dpt.ru/page/faq (Reference date: 21.07.2021)

[4] CAD and graphics [Electronic source] URL: http://sapr.ru/article/8311 (Reference date: 21.08.2021)

[5] Floyd J 2013 3D Printing: Build Your Own 3D Printer and Print Your Own 3D Objects (California, USA: Que Publishing) p 192

[6] FDM technology. About modeling of deposited deposits [Electronic source] URL: http://www.stratasys.com/3dprinters/technologies/fdm-technology (Reference date: 21.08.2021)

[7] Galashin M P and Shcheglov I A 2006 Development and implementation of algorithms for three-dimensional triangulation of complex spatial domains: direct methods KIAM Preprint vol 10 (Moscow: Keldysh Institute of Applied Mathematics) p 32

[8] Galashin M P and Shcheglov I A 2006 Development and implementation of algorithms for three-dimensional triangulation of complex spatial areas: iterative methods KIAM Preprint vol
Acknowledgments
This work was supported by the Russian Foundation for Basic Research in the framework of Grant No. 19-37-90063 Postgraduate students (Modeling the technology of layer-by-layer fission in 3D printing of complex-shaped products).