Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

RESEARCH ARTICLE

Color printing based on 3MF: color diffusion from the surface to the interior of voxel model

Liu Ruiying[1] | Yang Weidong[1] | Li Haixia[1] | Wang Yuanyuan[1] | Wang Haoyu*[2]

[1]School of Mechanical Engineering, Hebei University of Technology, Tianjin, China
[2]Robotics and Mechatronics, Central Connecticut State University, New Britain, United States

Correspondence
Wang Haoyu, Robotics and Mechatronics, Central Connecticut State University, New Britain 06050, United States. Email: wanghao@ccsu.edu

Abstract

Full-color 3D printing technology has become more and more popular for industrial manufacturing applications. The voxelization of color 3D models and the expression of appearance colors become more and more important. To illustrate the complex appearance and internal characteristics of 3D models, we present a voxelization algorithm of the color 3D models for color 3D printing in this paper. Specifically, we use the 3D model of 3MF with color appearance features for surface voxelization first. Then, we propose an improved scan line filling algorithm to realize the voxelization of the interior of the model and obtain the color voxel model. Finally, to improve the high quality of color printing, we propose a surface color inward diffusion algorithm, which can make the surface color of the voxel model diffuse inward and further express the surface color of the voxel model. The experimental results show our proposed method can obtain an effective color voxel model and accurately express the color of the voxel model surface, which can enhance the color effect of the color model and realize the control of voxels.

KEYWORDS:
3D printing, voxelization, color representation, inward diffusion

1 | INTRODUCTION

Full-color 3D printing technology has become the trend of additive manufacturing[1], which has big demand in education, art, manufacturing, construction, and other fields.[2,3,4,5] The traditional 3D printing process only focuses on discretizing a 3D model into two-dimensional slices and ignores the multi-color and multi-material nature of a 3D model.[6,7] This does not meet the development requirements of color 3D printing. To illustrate the complex appearance and internal characteristics of a color 3D model during 3D printing, the complete and accurate expression of printing location and color information is required.

At present, the STL format is widely used in model information representation for 3D printing. It uses triangular mesh as the minimum unit to represent the surface model. Although STL format is widely used and supported by CAD and other software, it has disadvantages such as a large data file, low accuracy, high redundancy. In addition, STL cannot address properties such as color and material. To solve this problem, new file formats such as 3MF, AMF, and OBJ were introduced.[8,9,10]. Among them, the 3MF format has the characteristics of information completeness, high readability, easy verification, great extensibility, and high compatibility. The 3MF format can represent complete information of a color 3D model, which includes not only the geometric information of the triangular mesh of the 3D model, but also the information about the color, material, texture, and other attributes of the model. This makes 3MF a far superior format than STL for color 3D printing.

In the field of 3D printing, voxels are the smallest three-dimensional volume that a 3D printer can reliably print materials. The process of converting a geometric model into a voxel model is called voxelization which is the pre-processing for any color 3D model printing process.[11] Generally, voxel data is stored as an n-dimensional matrix.[12] Therefore, the voxel model can describe the three-dimensional structure discretely to accurately control the internal structure of the model during printing.

For any 3D model voxelization process, the result of the voxelization within the model determines the result of the voxelization of the entire model. STL slicing algorithm[13] has been used to obtain the slice contour of the 3MF format of a 3D model and the scan line filling algorithm[14] has been used to generate rasterized slices to create a voxel model. However, the current scan line filling algorithm does perform well in color 3D printing because it doesnt consider the relationship between the surface voxels and the internal voxels of the model.

A printing system using powder binder and lamination technology can print color 3D models. But it cannot provide the resolution or the smooth surface of one printed by a multi-nozzle system and is limited only to near opaque materials. Therefore, it has little potential in printing with complex appearance characteristics in the future.

Contrary to a general 3D printing system which only needs to define the number and thickness of slices, a multi-jet system machine using translucent materials for color 3D models also relies on the accurate definition of color of the voxels under the surface of the material for better visual effects.[15,16] So, it is needed to determine the boundary coloring width of each slice.[17] The color definition of the internal voxels within a certain range under the surface of the color 3D model is very important.

Brunton[18] used the error diffusion halftone method to set the color values of voxels on the contour lines inside the model, which has a high time complexity because the size of voxels gradually decreases with the refinement of the model, which will lead to a large number of voxels in the voxel model, and the time cost of model voxelization increases exponentially. The calculation time for adjusting the color of the internal voxels under the model surface using the error diffusion halftone method is also greatly increased. Hu[19] and Xiao[20] both proposed the method of transforming triangular meshes into triangular prisms and made the color surface of the color 3D model equivalent to a thin-wall model, to improve the model surface color. However, when a voxel is inside an overlapped region of triangular prisms, they simply set the color value of the voxel to the color value of the triangular mesh with the shortest distance from it. In addition, they did not deal with voxels between two triangular meshes with an angle greater than 180°. This causes the color difference at the junction of the triangular mesh on the model surface to be too large and the color cannot be accurately illustrated. In the process of color 3D model voxelization, all factors affecting the accurate expression of color need to be considered. Therefore, the inner diffusion algorithm of surface color and the second diffusion algorithm of color was proposed by this paper.

To sum up, this paper studied the voxelization algorithm of the color 3D model in 3MF document format. The main contributions include: (1) parsing the 3MF document; (2) according to the characteristics of the color 3D model, an improved method of scan line filling algorithm was proposed; (3) the methods of color inward diffusion and second diffusion of surface voxels were proposed.

This paper consists of six parts: the first part includes the research background, purpose, significance, previous research results, and problems; the second part is the analysis of the color 3D model in 3MF document format and the surface voxelization of the model; the third part is about the internal voxelization of the model and the generation of voxel model; the fourth part is about the color expression of voxel model; the fifth part is the experimental verification; the sixth part is the summary.

2 | SURFACE VOXELIZATION OF COLOR 3D MODEL

Voxelization is the process of gradually discretizing the model into individual voxels, including surface voxelization and internal voxelization. The surface voxelization process of the color 3D model in 3MF document format includes: first, the data structure of the color 3D model in 3MF document format is analyzed to obtain the digital model; secondly, the model is sliced from bottom to top along the positive direction of $z-axis$ according to the thickness of each layer until the three-dimensional model is discretized into two-dimensional slices; finally, the contour of each slice is rasterized into raster points. All raster points generated by the contours of all slices are surface voxels of the model for surface voxelization of the model.

2.1 | Analyzing color model of 3MF document format

The 3MF document format uses common structures such as Open Packaging Convention, ZIP, and XML to simplify development process. The 3MF document is opened in the form of ZIP to obtain the digital model of the color 3D model based on the XML

format. The resulted XML document includes the information of color, material, and texture of the 3D model and the vertices of the triangular mesh. Like STL, 3MF uses triangular meshes to represent the model. However, according to the topological relationship between triangular meshes[21], 3MF uses the index of vertices to represent triangular meshes. 3MF eliminates the redundancy caused by the three vertices of the triangular mesh being stored many times. So, the storage space occupied by the data is reduced by two-thirds compared with that of STL. The triangular mesh in the 3MF document format is a composite unit composed of attributes such as vertices, edges, faces, texture, and materials.

In the data structure of the triangular mesh of the 3MF shown in Figure 1, <texture> is a container that stores the texture characteristics of the 3D model. The elements in the container are the UOV coordinates of the vertices of the texture triangle. The order of these elements forms an implicit index starting from 0, which will be referenced by the elements in other containers, such as <triangles>. The elements in <vertices> are the position coordinates of the vertices of the triangular mesh in the right-handed coordinate system. The implicit index formed by these elements starting from 0 will be referenced by the elements of other containers.

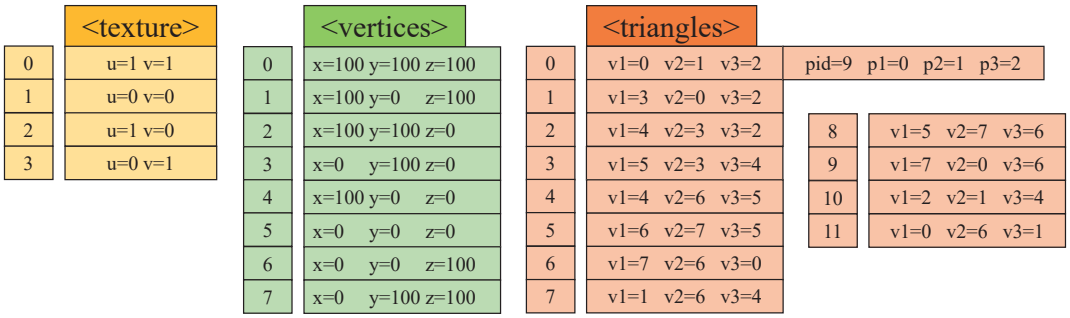


Figure 1 The data structure of triangular meshes.

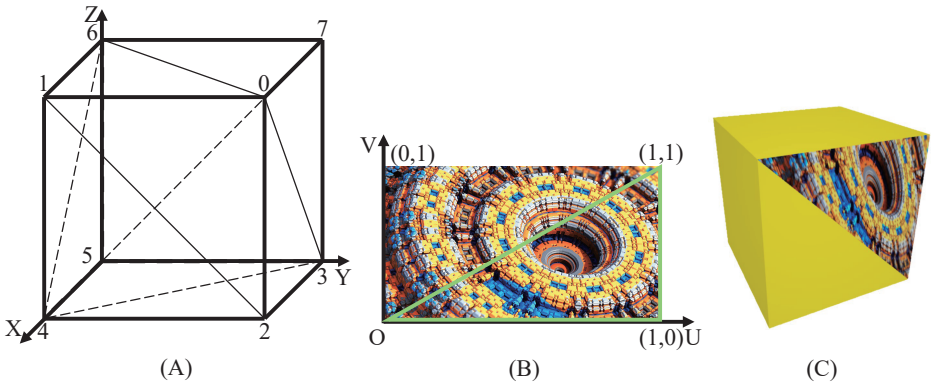


Figure 2 Color cube. (A) Triangular mesh model, (B) texture, (C) thumbnail.

Each element in <triangles> represents a triangular mesh. The values of $v1$, $v2$, and $v3$ in the element are the index of the triangle mesh vertex coordinates in the <vertex> container. Indexes that reference three vertices form a triangular mesh. Pid is the ID of the 2D texture of the model; $p1$, $p2$, and $p3$ are the indexes of the vertices of the texture triangle in the <texture> container; and a texture triangle corresponding to the triangle mesh is formed by referencing the three vertices of the texture triangle.

In the 3MF document, there is a topological relationship between the data of triangular meshes. For example, in the <triangles> container, $v3$ and $v1$ of the element with index 0 and $v2$ and $v3$ of the element with index 1 refer to the same vertices in the <vertices> container, indicating that the triangular meshes corresponding to these two elements share one edge and two

vertices. The cross product of the direction vectors of any two edges of the triangular mesh is the normal vector of the triangular mesh. For example, in the 0th element of <triangles>, the three vertices of the triangular mesh are sorted in the order of $v1$, $v2$, and $v3$, and the direction vector of the edge $\overrightarrow{v1v2}$ is $(v2 - v1)$. the direction vector of the edge $\overrightarrow{v2v3}$ is $(v3 - v2)$, and the normal vector n of the triangular mesh is $(v2 - v1) \times (v3 - v2)$.

Figure 2 is a triangular mesh model, texture, and thumbnail of the model of the color cube. In Figure 2A, the 8 vertices of the triangular mesh model of the cube correspond to the $0-th \sim 7-th$ elements of the <vertices> container in the triangular mesh data structure. These 8 vertices are repeatedly referenced by the elements in the <triangles> container to form 12 triangular meshes. Figure 2B is the 2D texture of the triangular mesh surface. The 2D texture is mapped to the surface of the triangular mesh by constructing UOV coordinates. The UOV coordinates of the lower left corner of the 2D texture is (0,0) and the upper right corner is (1,1). Figure 2C is a thumbnail of the model.

2.2 | Slicing of the 3MF model

The surface of the color 3D model has color texture, so when layering, not only the geometric triangular mesh of the model but also the texture triangle corresponding to the triangular mesh needs to be sliced. Texture triangles are divided according to the division scale of the geometric triangle mesh.

The slicing process for the color 3D model of 3MF includes the following main steps:

(1) Each triangular mesh constituting the model is traversed in turn and slices are made from bottom to top along the positive direction of the $z-axis$ according to (the thickness of the layers to generate the disordered line segments of the slice contour. The textured triangle is segmented incrementally or decremental according to the ratio of (layer thickness/triangle mesh side length). The line segment of the texture obtained by the texture triangle segmentation and the line segment of the slice contour are bounded together. Which means the data structure of the line segment of the color slice contour should include the position coordinates (x, y, z) and texture coordinates (u, v) of the two endpoints.

(2) Sorting the disordered line segments according to the z coordinate value; the line segments with the same z value are the line segments of the slice contour of the same layer slice.

(3) Because the model is sliced along the positive direction of the $z - axis$, the direction vector $\vec{D}$ of the line segments of the slice contour is the cross product of the unit vector in the positive direction of the $z - axis$ and the normal vector of the triangular mesh to which this line segment belongs, that is, $\vec{D} = (0, 0, 1) \times \vec{N}$. The two endpoints of this line segment are sorted along the direction pointed by the direction vector $\vec{D}$.

(4) According to the topological relationship between triangular meshes, the line segments of the slice contour of the same layer are joined to generate a closed planar polygon which is the slice contour with positive direction.

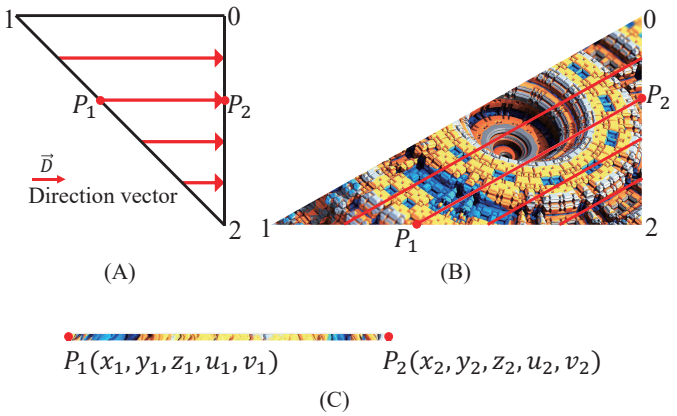


Figure 3 Segmentation of triangle mesh and texture triangle. (A) Triangle mesh, (B) texture triangle, (C) segment of slice contour.

In the <triangles> container of the cube in Figure 2, the triangular mesh with the index value of 0 has the attribute of the texture. As shown in Figure 3, while the triangular mesh of Figure 3A is sliced, the texture triangle of Figure 3B is also sliced according to scale, The points P_1 and the point P_2 have both position coordinates (x, y, z) and texture coordinates (u, v). The points P_1 and the point P_2 of the slice contour are sorted according to the direction of the direction vector $\overrightarrow{D}$. The segment of the slice contour obtained by the triangular mesh segmentation and the segment of the texture obtained by the texture triangle segmentation together constitute the line segment of the color slice contour as shown in Figure 3C.

2.3 | Surface voxelization

The slice contour of the model is further discretized to generate surface voxels of the model. This process is called surface voxelization of the model. The model composed of surface voxels is called the surface voxelization model. The slice contour is rasterized according to the resolution of a given printer, that is, the size of voxels. The Bresenham algorithm[22] was used to rasterize the line segments of the slice contour and the texture. The rasterization increment of the line segments of the slice contour is the size of the voxels. The rasterization increment of the line segments of the texture is equal to (the length of the line segments of the texture/the number of raster points generated by the line segments of the slice contour). The center position coordinates and texture coordinates of the surface voxels can be obtained by rasterization calculation.

Each surface voxel of the model corresponds to a pixel in the 2D texture. The pixels are sorted from top to bottom and from left to right in the picture. Therefore, the pixels corresponding to the texture coordinates of the surface voxels need to be converted into the positions of the points in the texture picture to correctly obtain the color values of the surface voxels. Figure 4 is a diagram to illustrate the positions of pixels in a picture. Each grid in the diagram represents a pixel. In a picture file, the upper left corner is the first pixel and the lower right corner is the last pixel. The position B_i of the pixel in the picture is obtained through equations (1)-(3), So, the color values of the four RGBA channels of the pixel can be read from the data of the picture file. The value of the alpha channel representing the transparency of the picture is generally set to 255 which represents complete opacity. The transparency of the texture picture of the color 3D model only represents the transparency of the model when visualized in a computer software. It has nothing to do with the transparency of the translucent material used during printing.

$$x = u \times (w - 1) \tag{1}$$

$$y = v \times (h - 1) \tag{2}$$

$$i = w \times (h - 1 - y) + x \tag{3}$$

Where w is the width of the picture, h is the height of the picture, x is the number of the column where the pixels are located, and y is the number of the row where the pixels are located. The texture picture contains $w \times h$ pixels.

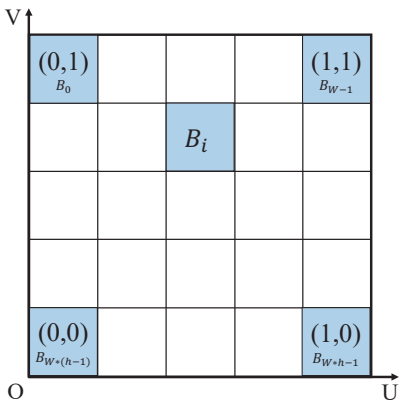


Figure 4 Position of the pixel in texture picture

In addition to the center coordinates of the surface voxels, the color values of the four RGBA channels of the pixels in the texture picture corresponding to the surface voxels are also included in the data structure of the surface voxels generated by the slice contour rasterization. As shown in Figure 5A,B color cube of the size of $100mm \times 100mm \times 100mm$ was voxelized when the voxel size was $1mm$ and $0.5mm$ respectively. The smaller the voxels, the more refined is the appearance of the voxel model and the smoother is the surface of the final printed color 3D model.

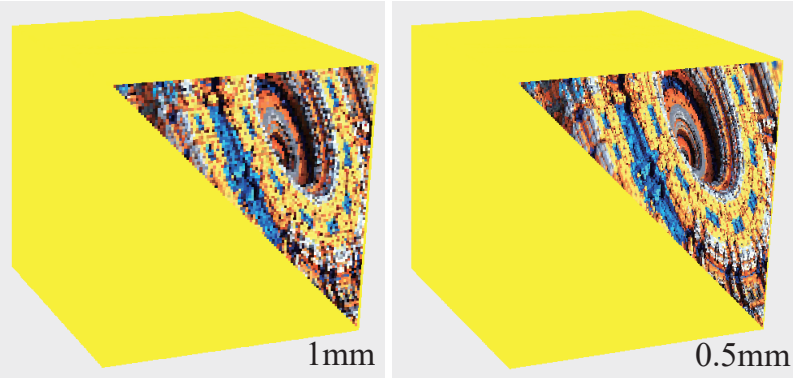


Figure 5 Surface voxelization model of color 3D model. (A)$1mm$, (B)$0.5mm$.

3 | INTERNAL VOXELIZATION OF COLOR 3D MODEL

The final voxel model can be obtained by filling the interior of the surface voxel model obtained in Section 2.3. The surface voxels with the same coordinate value of z are located on the contour of the same slice. The scan line filling algorithm was used to fill the inside of the slice contour to generate a rasterized slice. The raster points filled in the rasterized slice are the internal voxels with height z. The process of filling the contour of each slice is the process of voxelization inside the model. Each layer of rasterized slices can be regarded as a color bitmap image with z coordinates.

Because the surface of the color 3D model has color texture, the scanning line filling algorithm needs to be improved to make it suitable for the process of internal voxelization of the color 3D model. The surface voxels of the model were combined with the improved scan line filling algorithm to fill the interior of the model and generated the internal voxelization of the model.

3.1 | Improvement of scan line filling algorithm

The scan line filling algorithm normally has four steps: (1) Calculate the intersection of the scan line and each side of the polygon slice contour; (2) Sort the intersection points on each scanning line in the order of increasing x coordinate value; (3) Pair the intersection points according to the odd-even rule to determine the filled interval of the polygon; (4) Record the interval to be filled. For the singular points generated because the scanning line accidentally intersects with the contour edge, the point with the largest y value on the contour line needs to be ignored when calculating the intersection point[23].

The spacing between scan lines in the scan line filling algorithm is equal to the size of a voxel. The calculation process of the intersection of scan lines and each edge of the slice contour is the same as the process of the Bresenham algorithm using raster points to approximate a straight line. Therefore, the process of rasterizing the slice contour using the Bresenham algorithm mentioned in Section 2.3 is the process of calculating the intersection of scan lines and the model slice contour in the scan line filling algorithm.

In the first step of using the scan line filling algorithm, since the interval of the scan lines is fixed, and the interval of the raster points on the slice contour in the direction parallel to the $x - axis$ or the $y - axis$ is same, rounding needs to be considered. In this process, as shown in Figure 6A, during the rasterization of line segment P_6P_0, the x coordinates of two adjacent raster points are different but the y coordinates are same.

In the second and third steps of the scan line filling algorithm, after the points on the scan line are sorted, the interval between the odd and even points is to be filled. Therefore, on a line segment of the same rasterized slice contour, only one of the two adjacent points in the horizontal direction can be kept. Otherwise, when filling the inside of the slice contour, there will be a problem of vacancy or wire drawing. As shown in Figure 6B, the points P_0 P_6 are sorted anticlockwise to form a polygon. The scanning line of $y = 8$ intersects with the edges P_0P_1, P_1P_2, P_5P_6, and P_6P_0 of the polygon. Because the interval between raster points needed to be rounded, five raster points were generated. The five points were sorted according to the x coordinate value. The intervals to be filled were determined according to the odd-even rule and are between points 1 and 2 and between points 3 and 4. The space between points 4 and 5 is blank. This resulted in unfilled areas inside the polygon.

Therefore, in the improved scheme of the scan line filling algorithm, it was proposed that on the line segment of the same rasterized slice contour, the appropriate point is selected as the filling mark point from the adjacent points in the horizontal direction to avoid problems in filling. As shown in Figure 6C, after line segment P_6P_0 was rasterized, one of points 3 and 4 who have the same coordinate value of y was selected as the filling mark point after the singularity point with the highest y value was removed. In Figure 6D, the four filling mark points on the scanning line with $y = 8$ were sorted according to the value of x. The filling areas are between points $1'$ and $2'$ and points $3'$ and $4'$.

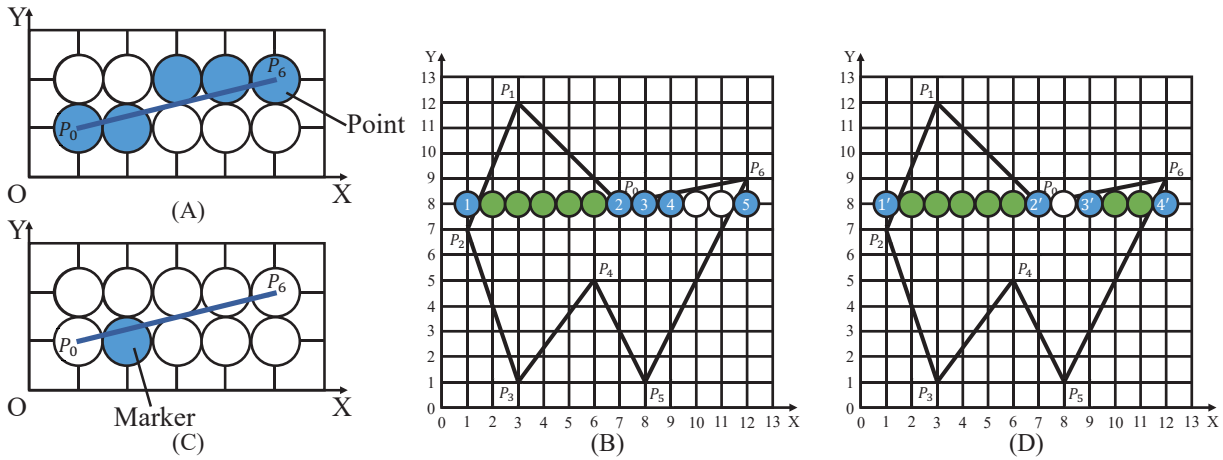


Figure 6 Scan line filling algorithm. (A) Rasterization of lines, (B) raster points of polygon outline, (C) fill mark points, (D) fill mark points of polygon outline.

In the process of obtaining the voxels inside the model using the scan line filling algorithm, if the points on the slice contour are arbitrarily selected as the filling mark points, the surface of the part of the model will lose the color attribute. Therefore, the following method of selecting the filling mark points was proposed. First, the slope of a contour line segment is used to tell if raster points with same y values exist. If so, the normal vector of the contour line segment is used to select the raster point which is closer to the inside of the slice as the filling mark point.

Figure 7 is the result of the rasterization of straight line segments with different slopes. When the slope k of the straight line is $k \to \infty$, $k \geqslant 1$, or $k \leqslant -1$, the coordinate values of y of the generated raster points are not the same. Therefore, after the singularity points are eliminated, the remaining raster points can be used as the filling mark points. When the slope k of the straight line is $-1 \leqslant k \leqslant 0$, $k = 0$,or $0 < k < 1$, adjacent raster points with the same coordinate value of y exist in the generated raster points. Therefore, how to select the filling mark points in the improved scan line filling algorithm in these three cases is discussed as follows.

When the slope of the straight line is 0, the y coordinate values of all the generated raster points are the same and the number of points is uncontrollable. So, the raster points on the line segment of the slice contour with the slope of 0 cannot be used as the filling mark points.

As shown in Figure 8, Figure 8A shows the normal vector $\overrightarrow{N}$ of the triangular mesh and Figure 8B shows the projection vector $\overrightarrow{N_P}$ generated by the projection of the triangular mesh normal vector $\overrightarrow{N}$ onto the XOY plane. It is the normal vector $\overrightarrow{N_P}$ of the line segment of the slice contour, as shown in Figure 8C.

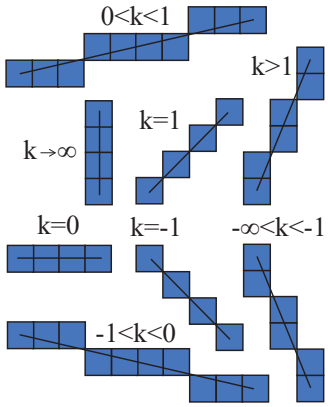


Figure 7 Line rasterization with different slopes.

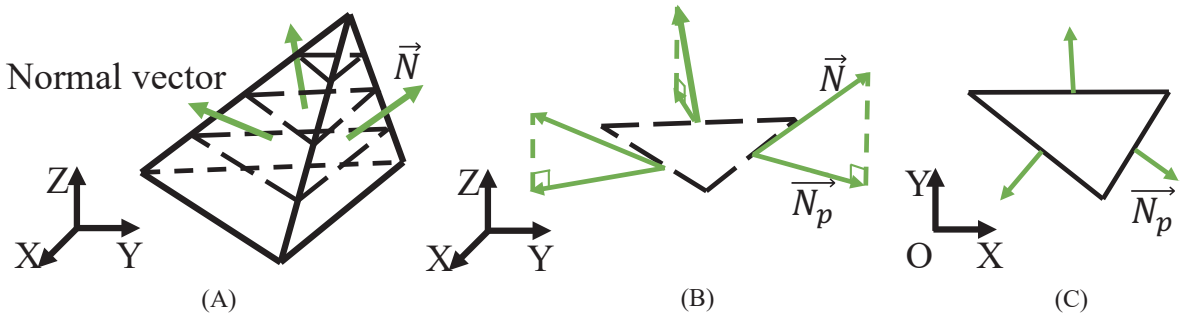


Figure 8 3D model slice contour. (A) The normal vector of triangular meshes, (B) projection vector, (C) normal vector of the line segment.

The slope of the line segment of the slice contour is $-1 < k < 0$, as shown in Figure 9A,B. When the included angle between the normal vector n of the line segment and the positive direction of the $x-axis$ is greater than 45° and less than 90°, the filling mark point of the line segment should be the raster point with the smallest value of x (closest to the inside of the slice) among the raster points with the same coordinate value of y. When the included angle between the normal vector $\overrightarrow{N_P}$ of the line segment and the positive direction of the $x-axis$ is greater than 225° and less than 270°, the filling mark point on the line segment is the raster point with the largest coordinate value of x among the raster points with the same coordinate value of y.

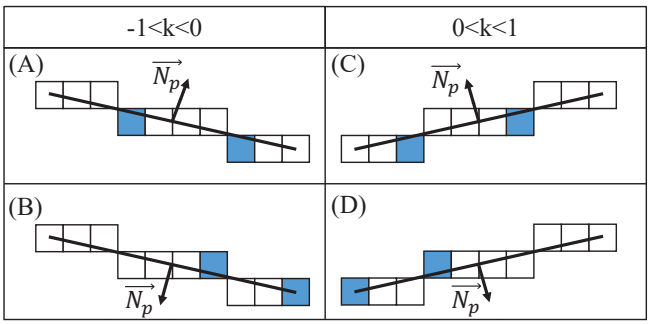


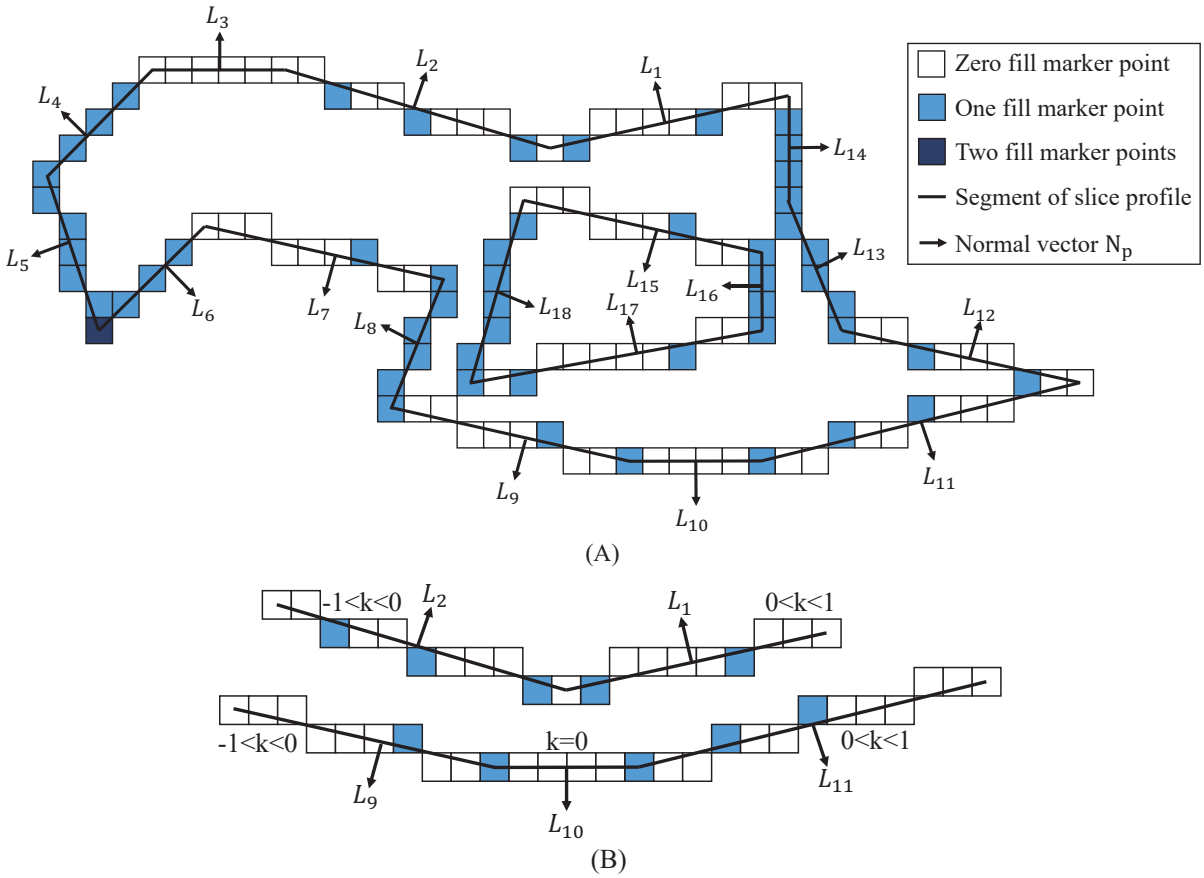
Figure 9 Filling marker points.

As shown in Figure 9C, D, the slope of the line segment of the slice contour is $0 < k < 1$. When the included angle between the normal vector $\overrightarrow{N_P}$ of the line segment and the positive direction of the $x - axis$ is greater than 90° and less than 135°, the raster point with the largest coordinate value of x among the raster points with the same coordinate value of y is the closest to the inside of the slice and should be selected as the filling mark point. When the included angle between the normal vector $\overrightarrow{N_P}$ of the line segment and the positive direction of the $x - axis$ is greater than 270° and less than 315°, the raster point with the smallest coordinate value of x among the raster points with the same coordinate value of y is the closest to the inside of the slice and should be selected as the filling mark point.

After eliminating the singularity points, the filling mark points of the irregular polygon in Figure 10A were determined according to the above mentioned filling mark point selection process. As shown in Figure 10A, in an irregular polygon with inner and outer boundaries, the line segments $L_1 - L_{14}$ were sorted counterclockwise to form the outer boundary of the polygon, while the line segments $L_{15} - L_{18}$ were sorted clockwise to form the inner boundary of the polygon. The line segment $L_1 - L_{18}$ were rasterized to generate raster points and the filling mark points of the polygon were selected according to the improved scanning line filling algorithm. Filling mark points are represented as blue or dark blue raster points in the figure. White raster points represent zero filling mark points; blue raster points represent one filling mark point; and dark blue raster points represent two filling mark points (two filling mark points with the same geometric position but belonging to two different contour line segments).

As shown in Figure 10B, on the line segments L_1, L_2, L_9, L_{10}, and L_{11}, there were first removed the raster point with the largest coordinate value of y which is a singularity point on each line segment. Then the improved scan line filling algorithm was used to select the filling mark points on the polygon contour line segments when the slope k is $-1 < k < 0$, $k = 0$, or $0 < k < 1$.

Figure 10C is to illustrate a rasterized polygon generated by filling the interior of the polygon according to the odd-even rule using the improved scan line filling algorithm described above. In a color 3D model, the points inside the rasterized slice contour are the internal voxels of the model.



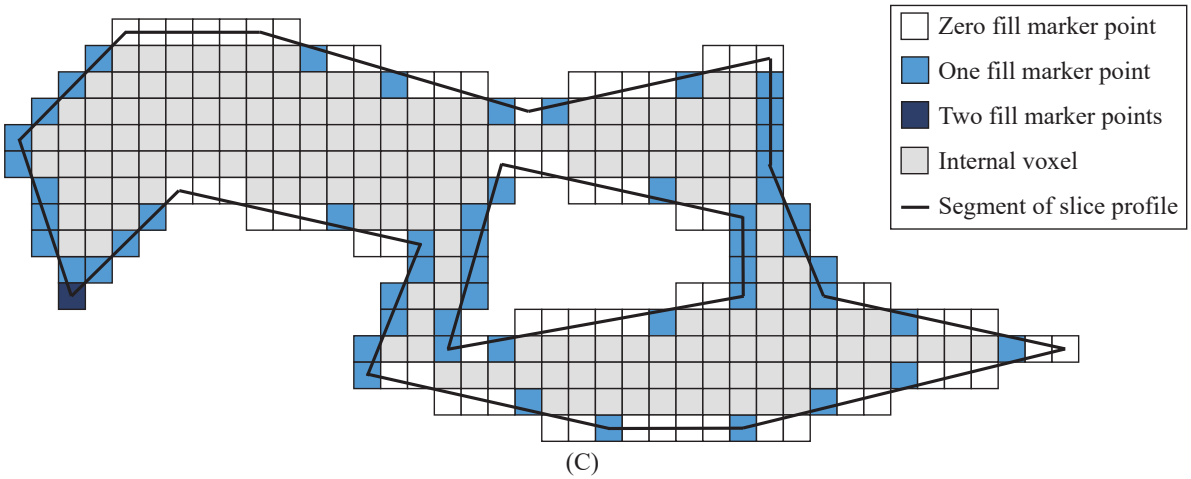


Figure 10 Irregular polygon. (A) Filled marker points in the contour, (B) partially filled marker points, (C) rasterized polygon.

3.2 | Voxel model

After the slice contour of each layer of the color 3D model is filled by the improved scan line filling algorithm, the rasterized slice is generated. The color contour of the rasterized slice is made of surface voxels and the interior is made of internal voxels. Figure 11 is the case where the ring is filled according to the improved method. Figure 11A is the ring model; Figure 11Bis the rasterized contour made of surface voxel; and Figure 11C is the rasterized slice. It can be seen from Figure 11 that the rasterized slice of the ring did not lose the color information of the surface and also realized the complete filling of the interior.

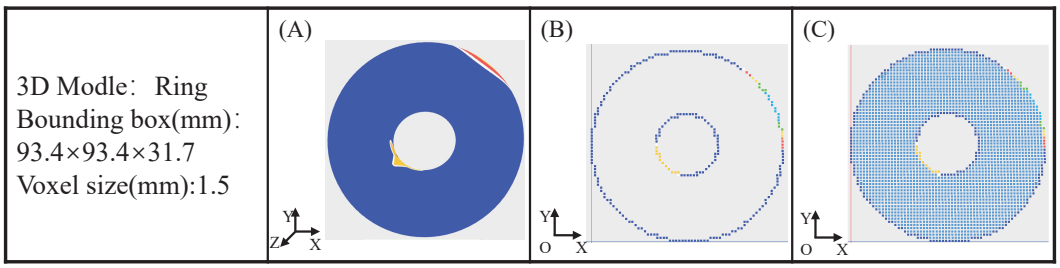


Figure 11 Ring slice contour filling. (A) Ring, (B) contour rasterization, (C) internal filling.

In the process of internal voxelization of the model, to ensure that the color information of the surface is not lost and avoid the problem of gaps or filaments in the model, in the process of slice rasterization, the improved scan line filling algorithm was used to select the filling mark points in the slice contour to ensure that the interior of each slice was filled. The internal voxelization of the model could obtain the geometric positions of all internal voxels. In the post-processing of the model, all physical attributes such as color and material could be bound to the structure representing the internal voxels. In the process of printing, precise control and management of the geometric and physical properties of each internal voxel of the model were achieved.

4 | COLOR REPRESENTATION OF VOXEL MODEL

To accurately reproduce the color of the model, the color values of the internal voxels under the surface of the color 3D model were calculated by using the inward diffusion algorithm of the surface color. When there is a concave region formed by a triangular mesh with an angle greater than 180° in the model, the color value of the internal voxels near the concave region was calculated using the color secondary diffusion algorithm.

4.1 | Inward diffusion algorithm of surface color

The color of the voxel on the surface of the color 3D model diffuses in the opposite direction of the normal vector $\overrightarrow{N_P}$ of the triangular mesh to which the voxel point belongs. The color value of the internal voxel within the diffusion range gradually becomes close to the base color inside the model. As shown in Figure 12, point C is a surface voxel, its color will gradually diffuse along the opposite direction of $\overrightarrow{N}$, and the color value of the internal voxel will gradually turn white. As shown in Figure 13, in addition to the gradual change of the color value, when the internal voxel of the model is within the inward diffusion range of the multiple surface voxels, the final color value will be jointly affected by the multiple surface voxels. Therefore, the weight of the influence of each point on the point should be calculated according to the distance between the internal voxel and each surface voxel. Finally, the final color value of the internal voxel is obtained through weighted calculation.

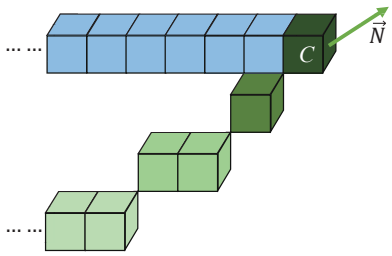


Figure 12 The color of surface voxels diffuses inward.

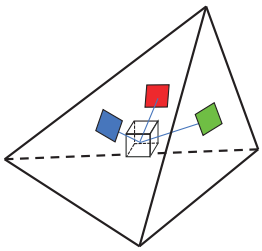


Figure 13 Affected internal voxels.

The position of the internal voxels is affected by the color of the surface voxels during the inward diffusion:

The size of the voxel is m, and the triangular grid normal vector $\overrightarrow{N} = (x_n, y_n, z_n)$. The range within which the color of the surface voxels diffuses inward, the number of internal voxels affected in the diffusion process, is called the threshold value n_0.

Modulus of the normal vector of triangular mesh:

$$\left\|\overrightarrow{N}\right\| = \sqrt{x_n^2, y_n^2, z_n^2} \tag{4}$$

The unit vector of the normal vector of triangular mesh:

$$\vec{e} = \frac{\overrightarrow{N}}{\left\|\overrightarrow{N}\right\|} = \left(\frac{x_n}{\left\|\overrightarrow{N}\right\|}, \frac{y_n}{\left\|\overrightarrow{N}\right\|}, \frac{z_n}{\left\|\overrightarrow{N}\right\|}\right) \tag{5}$$

The position of the center of the affected internal voxel is $C - m\vec{e}, C - 2m\vec{e}, C - 3m\vec{e}, \cdots, C - n_0 m\vec{e}$.

The internal voxel is affected by T surface voxels; i represents the current internal voxel affected by the $i-th$ surface voxel; d_i is the distance between the current internal voxel and the $i-th$ surface voxel, $0 \leqslant d_i \leqslant n_0$;the center coordinate of the $i-th$ surface voxel is $P_i(x_{P_i}, y_{P_i}, z_{P_i})$; color value $RGB_{P_i}(R_{P_i}, G_{P_i}, B_{P_i})$ are color values of the red, green and blue color channels of

RGB_{P_i}. The base color inside the model is $RGB_B = (R_B, G_B, B_B)$. R_B, G_B, and B_B are the color values of the three channels of the base color. Suppose the base color inside the model is white and the color value of white is $RGB_B(255, 255, 255)$. The farther the internal voxel P is from the point P_i, the less it is affected by the point P_i color value. Thus, the closer the color value of the point P is to the base color inside the model. For the internal voxel with a distance of d_i from the $i-th$ surface voxel, the color value affected by the $i-th$ surface voxel is $RGB_i(R_i, G_i, B_i)$, while R_i, G_i, and B_i are the values of the three channels of the color value RGB_i of the current internal voxel affected by the ith surface voxel.

The color value of the internal voxel within the diffusion threshold of the $i-th$ surface voxel:

$$R_i = R_{P_i} + \frac{(R_B - R_{P_i}) \times d_i}{n_0} \tag{6}$$

$$G_i = G_{P_i} + \frac{(G_B - G_{P_i}) \times d_i}{n_0} \tag{7}$$

$$B_i = B_{P_i} + \frac{(B_B - G_{P_i}) \times d_i}{n_0} \tag{8}$$

Internal voxel P final color value:

$$RGB = \sum_{i=1}^{T} RGB_i \times p_i \tag{9}$$

p_i is the weight of the surface voxel affecting the color value of the current internal voxel. The farther the current internal voxel is from the surface voxel, the smaller is the weight of this surface voxel.

$$p_i = \frac{(n_0 - d_i)^2}{\sum_{j=1}^{T}(n_0 - d_i)^2} \tag{10}$$

4.2 | Color secondary diffusion algorithm

When there are concave areas formed by triangular meshes with an included angle greater than 180° in the model, the included angle between the line segments of adjacent slice contours will be greater than 180°. As shown in Figure 14, according to the method that the color of the voxels on the surface of the model diffuses inward along the opposite direction of the normal vector of the point, the voxel P' in the model is located in the areas where diffusion cannot be performed between adjacent slice contours. Therefore, when the included angle between the line segments of the slice contour is greater than 180°, the intersection point of two line segments will be calculated as two overlapping voxels.

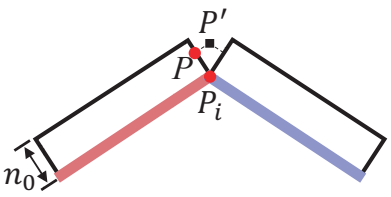


Figure 14 Concave contour edge.

As shown in Figure 15, the $i-th$ surface voxel P_i is at the end point of the slice contour line segment. The voxel P is within the diffusion range in the opposite direction of the normal vector of point P_i. When there is an increment in the vertical direction in the diffusion direction of point P_i, point P whose color value is affected by the point P_i will undergo secondary diffusion in the positive and the negative directions of the $x-axis$. Similarly, when there is an increment in the horizontal direction in the

diffusion direction of point P_i, point P will undergo secondary diffusion in the positive and negative directions of the $y-axis$. Now if color at point P is allowed to diffuse, point P' will be within the diffusion range. The point P' is in the area between the line segments of the slice contour with an included angle greater than 180° and cannot be reached through diffusion from point P_i. In the secondary diffusion of the color value of the point P, the secondary diffusion threshold is n_0. The color value RGB_i of the internal voxels within the secondary diffusion range affected by the point P_i will become close to the color value RGB_B of the internal base color of the model.

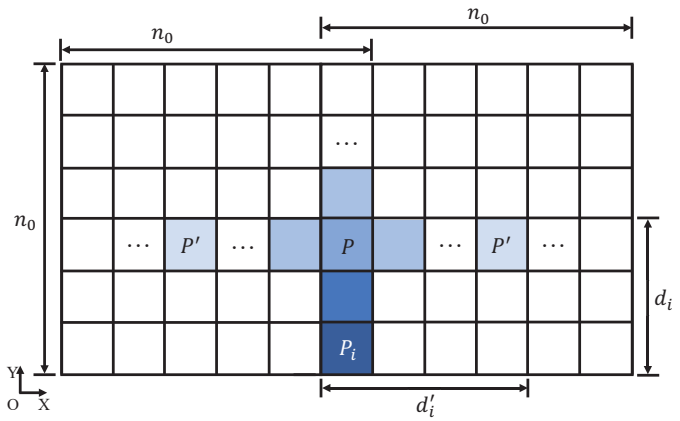


Figure 15 Second diffusion of color values.

The distance d_i in equations (7)-(9) will be changed into the relative distance D_i between the current internal voxel P' and the surface voxel P_i. D_i is the sum of the distance d_i between point P and point P_i and the relative distance $(n_0 - d_i) \times \frac{d_i'}{n_0}$ between point P' and point P. The relative distance between point P' and point P means the number of points that have not been diffused when the color value of point P_i is diffused to point P. It is the distance $(n_0 - d_i)$ from the last point to the point P within the range of the diffusion threshold of the point P_i, multiplied by the ratio of the distance d_i' between point P' and point P and the diffusion threshold n_0. The calculation formula of the final color value RGB_i' of the point P' is equation (14). The weight calculation formula is equation (15).

The color value of the inner voxel within the second diffusion threshold range of the $i-th$ surface voxel is:

$$RGB_i' = RGB_{P_i} + \frac{(RGB_B - RGB_{P_i}) \times D_i}{n_0} \tag{11}$$

$$D_i = d_i + (n_0 - d_i) \times \frac{d_i'}{n_0} \tag{12}$$

The final color value of internal voxel P':

$$RGB' = \sum_{i=1}^{T} RGB_i' \times p_i' \tag{13}$$

p_i' is the weight of the surface voxel affecting the color value of the current internal voxel. The farther the current internal voxel is from the surface voxel, the smaller the weight of this surface voxel.

$$p_i' = \frac{(n_0 - D_i)^2}{\sum_{j=1}^{T}(n_0 - D_i)^2} \tag{14}$$

5 | RESULTS AND DISCUSSION

In the visual studio environment, C + + language was used to develop the software to realize the voxelization of color 3D models proposed in this paper. The processor is Intel (R) core (TM) i7-8700k 3.70ghz and the running memory is 20GB. Visualization of 3MF model and voxel model was implemented using OpenGL under MFC framework. In the following, regular, irregular, and complex models were voxelized, and the color of the surface of the voxel model was diffused to the inside of the model using the algorithm proposed in Section 4.

5.1 | Regular model

Figure 16A is a color cube, and Figure 16B is the texture expansion diagram of the cube. The size of the cube is $13.5mm \times 13.5mm \times 13.5mm$ and the size of the voxel is $0.3mm$. Figure 17A is the surface voxel of the cube. When the threshold value of the inward diffusion of the surface color is $n_0 = 0$, as shown in Figure 17B, in the voxel model of the cube, the bottom color of the surface voxels is white. In addition, there is only one layer of color voxels on the surface. When the threshold value of diffusion is $n_0 = 10$, as shown in Figure 17C, the color of the voxels on the surface of the cube was diffused to the inside within a certain range, which enhanced the expression of the color model on the surface color.

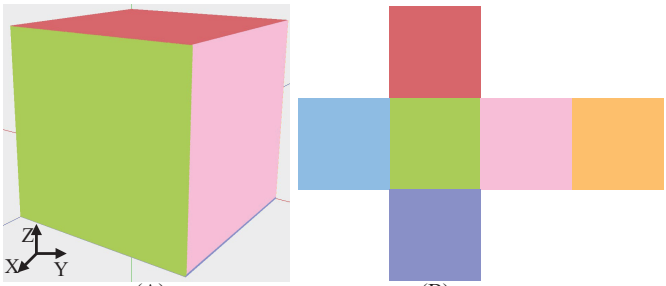


Figure 16 Cube with six different colors. (A) Cube, (B) texture.

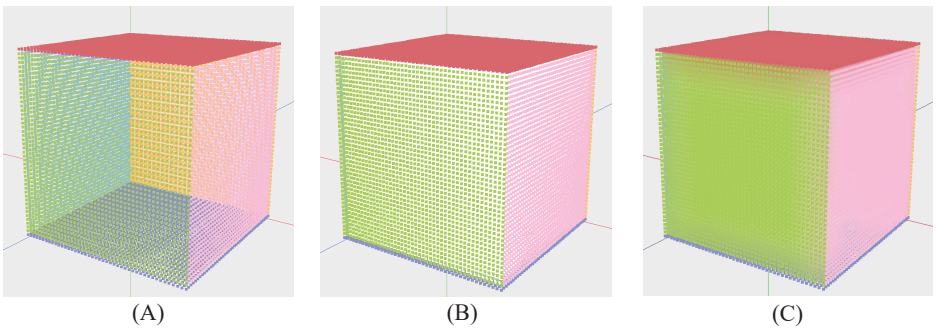


Figure 17 Voxelization of the cube. (A) Surface voxelization, (B) voxelization with a threshold of 0, (C) voxelization with a threshold of 10.

When the threshold n_0 is 0, 10, and 20 respectively, the cross-section of the cube parallel to the YOZ and XOY planes is shown in Figure 18. Figure 18A,B,and C are the cross-sectional view of the cube parallel to the YOZ plane, and Figure 18D,E,and F are the cross-sectional view of the cube parallel to the XOY plane.

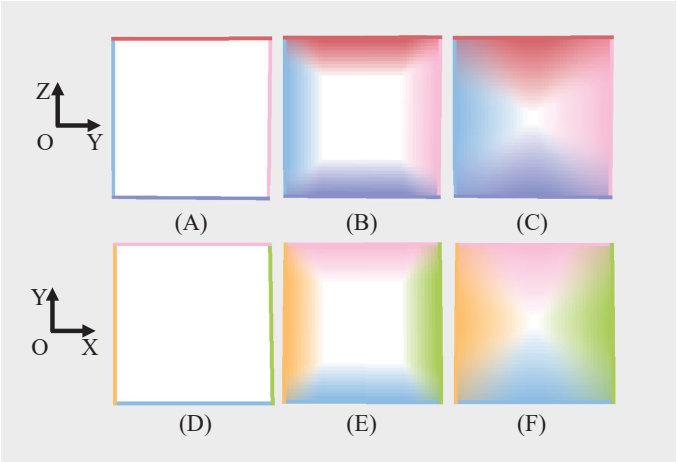


Figure 18 Section of the cube. Section parallel to YOZ plane (A) $n_0 = 0$, (B) $n_0 = 10$, (C) $n_0 = 20$, section parallel to XOY plane (D) $n_0 = 0$, (E) $n_0 = 10$, (F) $n_0 = 20$.

5.2 | Irregular model

When there is a concave region in the model, as shown in Figure 19A, the slice of the 3D model layered along the positive direction of the $z - axis$ is a concave polygon and the included angle between the line segments of the slice contour is greater than 180°. As shown in Figure 19B, the color of the surface voxels diffused toward the inside of the model along the opposite direction of the normal vector. The color value of the internal voxels in the area near the common point of the two contour lines with an internal angle greater than 180° is the base color, which resulted in discontinued color propagation. In Figure 19B, the common point of the two contour line segments with an internal angle greater than 180° was taken as two surface voxels, and they carried color in inward diffusion and secondary diffusion and calculated the color value of the internal voxels between the contour line segments with an internal angle greater than 180°.

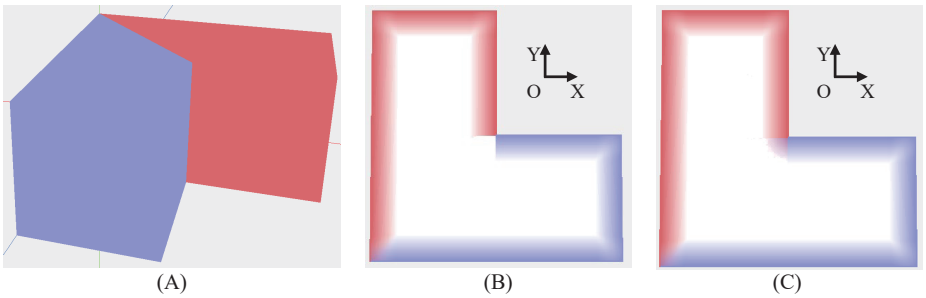


Figure 19 Irregular model. (A) Concave model, (B) inward diffusion, (C) secondary diffusion.

5.3 | Complex model

Figure 20A and Figure 20B respectively show the results of voxelization of the complex color model cat and dog when the voxel sizes are $0.03mm$ and $0.1mm$ respectively. They also illustrate the slice pictures of the voxel model at one-half of the $z - axis$ positive direction under different surface color inward diffusion thresholds. The slice of the cat in Figure 20A is composed of a plurality of polygons and the slice of the dog in Figure 20B is a concave polygon. The complex 3MF model used the improved scan line filling algorithm and the color inward diffusion algorithm to generate the voxel model, which achieved the expected result.

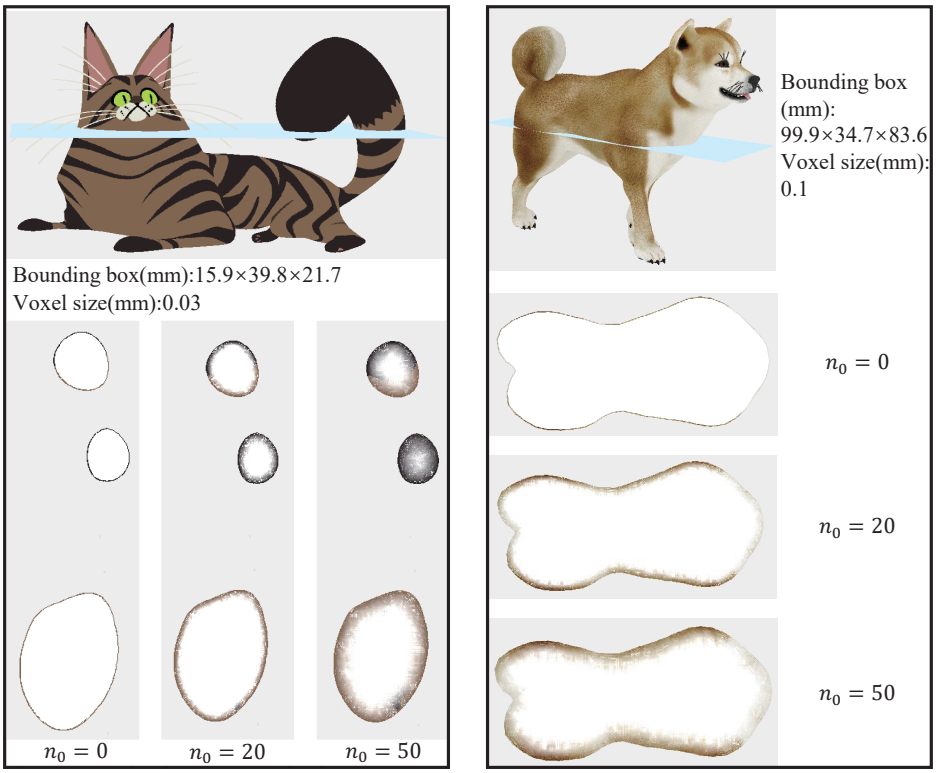


Figure 20 Slice of color voxel model in half. (A) Cat, (B) dog.

However, when there are thin and narrow areas in the model, although the algorithm of surface voxel inward diffusion is still effective, the color of the voxel points in the thin and narrow areas will be mixed. Furthermore, the color will tend to become same. Therefore, the algorithm needs further improvement to realize self-adaptability.

5.4 | Experimental analysis

Four color models of the cube, concave model, cat, and dog were used as experimental subjects. Voxels of different sizes were used for voxelization, and the color of the surface was diffused when the diffusion threshold was 0, 10, 20, 30, 40, and 50, respectively. Table 1 shows the voxelization data of the four models, including the size of the Axis-aligned Bounding Box, the voxel size, the number of surface voxels and internal voxels, and the running time of the model voxelization when the color is expressed. When the diffusion threshold n_0 is 0, the color of the model surface is not diffused to the inside. With the increase of diffusion threshold n_0, in addition to voxelization, the internal voxels also need to express the color from diffusions. Finally, the time required to convert the model into a voxel model is the time when n_0 is 0 plus the time when the color is expressed.

Table 1 Test data of voxelization.

Model	Bounding Box(mm)	Voxel size(mm)	Surface Voxel	Internal voxel	Voxelization time(s)					
					$n_0 = 0$	$n_0 = 10$	$n_0 = 20$	$n_0 = 30$	$n_0 = 40$	$n_0 = 50$
Cube	$13.5 \times 13.5 \times 13.5$	0.05	440092	19610100	98.86	117.14	131.85	147.31	176.16	200.70
Concave model	$20.0 \times 20.0 \times 10.0$	0.05	563774	23920000	119.2	146.05	179.69	217.36	256.7	315.27
Cat	$15.9 \times 39.8 \times 21.7$	0.03	1512231	96904240	485.7	615.89	829.5	1147.96	1501.86	2027.43
Dog	$99.9 \times 34.7 \times 83.6$	0.1	1335049	71377010	384.65	460.02	661.8	941.26	1284.65	1734.92

With the increase of n_0, the running time required for model voxelization in color expression gradually increases. The running time of voxelization of the four models and the diffusion threshold is fitted, as shown in Figure 21A-D. The fitted curve is a parabola. The relationship between the number of voxels in the model and the coefficients of the parabolic equation are shown in Table 2.

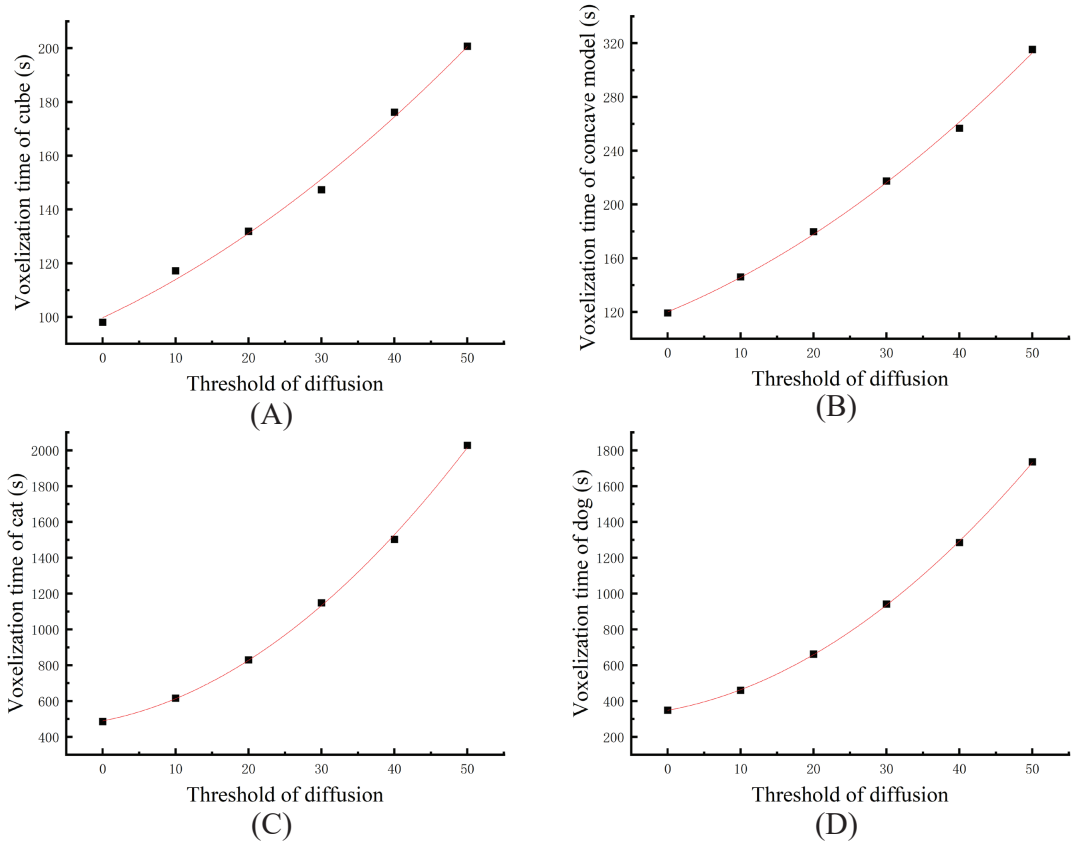


Figure 21 Fit the curve. (A) Cube, (B) concave model, (C) cat, (D) dog.

Table 2 Relationship between the number of voxels and the coefficient of the equation fitting the curve.

Model	Surface Voxel	Internal voxel	$y = Ax^2 + Bx + C$		
			A	B	C
Cube	440092	19610100	0.01491 ± 0.0054	1.27172 ± 0.28107	99.72668 ± 2.98815
Concave model	563774	23920000	0.03238 ± 0.00543	2.23802 ± 0.2831	120.07561 ± 3.00969
Cat	1512231	96904240	0.45323 ± 0.03164	7.86696 ± 1.6481	489.25296 ± 17.52129
Dog	1335049	71377010	0.40374 ± 0.01091	7.4836 ± 0.5682	348.02832 ± 6.04059

The model with more voxels takes longer to realize color expression. The more voxels in the model, the degree of curve bending is larger but very gentle. The coefficient A of the quadratic term of the curve is positively correlated with the number of

surface voxels and internal voxels of the model. These data show that, for models with complex surfaces, the number of voxels in the model is a decisive factor in the running time of the voxelization process when the color expression is accurately realized.

First, we compared our method with that using the error diffusion halftone algorithm. Brunton traverses the inner voxels at the same distance from the surface, spreads the color of the voxels to the adjacent voxels, and deepens the color depth with the increase in the number of layers. Our method is to diffuse the color of the surface voxels in the opposite direction of the normal, and then calculate the color value of the internal voxels affected by the surface according to the weight to achieve color expression.

It is assumed that the number of surface voxels of the model is S_0, and the number of voxels on the layer with the same distance from the surface is S_1, S_2,$\cdots$, and S_n. Using the error diffusion halftone algorithm, the color of the current voxel is diffused to adjacent voxels, the number of adjacent voxels is Q, and the time of color diffusion can be approximately $Q\times(S_1+S_2+\cdots+S_n)$. The time required by our algorithm can be approximated as $n\times S_0+(S_1+S_2+\cdots+S_n)$. When $(Q-1)\times(S_1+S_2+\cdots+S_n)$ is greater than $n\times S_0$, our algorithm may take less time.

6 | CONCLUSIONS

We proposed an efficient voxelization algorithm for color 3D models based on a 3MF document format. According to the characteristics of the color model, the scanning line filling algorithm was improved. The filling mark points were added to ensure that the color of the model surface is not missed. At the same time, the problem of internal missing in the process of model voxelization can be avoided. To enhance the color effect of the color model, we proposed an algorithm of color inward diffusion of surface voxels to express the color of the voxels under the surface of the model and achieve accurate control of voxels.

In the field of voxel-level additive manufacturing, it will be very easy to define the material distribution and color value management based on the color voxel model. It includes defining the distribution of complex functional gradient materials and color management during color 3D printing. However, with the refinement of the model, the size of voxels gradually decreases, which will lead to a large number of voxels in the voxel model. The time required for calculation also increases exponentially. Therefore, the algorithm of voxelization should be optimized in the future. To speed up the process of slicing and rasterization of the model, the results of the previous layer can be mapped to the current slice as the initial value which can be tracked, and re-paired with the filling mark points set by the user. It should reduce the time of calculating the center coordinates of voxels in the process of voxelization. In addition, voxels inside the slice contour can be merged. Voxels with the same material and color values can be merged into voxels of larger size to reduce the memory space required by the calculation. This will improve the overall efficiency of model voxelization.

ACKNOWLEDGMENTS

The authors are thankful for the support from the National Natural Science Foundation of China,under the Grant No.52175313.

Author contributions

Liu Ruiying: Data curation (lead); resources (lead); methodology (equal); software (lead); validation (lead); visualization (lead); writing-original draft (lead). Yang Weidong: Conceptualization (equal); methodology (equal);formal analysis (equal); project administration (equal); supervision (equal); writing-review and editing (lead). Li Haixia: Writing-review and editing (supporting); data curation (supporting). Wang Yuanyuan: Investigation (supporting); supervision(supporting). Wang Haoyu: Writing-original draft(supporting); Writing-review and editing (supporting); supervision(supporting).

Financial disclosure

This research was supported by the National Natural Science Foundation of China,under the Grant No.52175313.The authors would like to thank for this financial support.

Conflict of interest

The authors declare no potential conflict of interests.

References

1. Yuan J, Chen G, Li H, Prautzsch H, Xiao K. Accurate and Computational: A review of color reproduction in Full-color 3D printing. *Materials & Design* 2021; 209: 109943. https://doi.org/10.1016/j.matdes.2021.109943, https://www.sciencedirect.com/science/article/pii/S0264127521004974.

2. Hofmann DC, Bordeenithikasem P, Tosi LP, Hendry M, Stolpe M. Towards additively manufacturing excavating tools for future robotic space exploration. *Engineering Reports* 2020; 2(2).

3. Yin J, Li M, Dai G, Zhou H, Ma L, Zheng Y. 3D Printed Multi-material Medical Phantoms for Needle-tissue Interaction Modelling of Heterogeneous Structures. *Journal of Bionic Engineering* 2021; 18(2): 346-360. doi: 10.1007/s42235-021-0031-1

4. Page M, Obein G, Boust C, Razet A. Adapted Modulation Transfer Function Method for Characterization and Improvement of 3D printed surfaces. *Electronic Imaging* 2017; 2017: 92-100. doi: 10.2352/ISSN.2470-1173.2017.8.MAAP-279

5. Lachmayer L, Ekanayaka V, Hürkamp A, Raatz A. Approach to an optimized printing path for additive manufacturing in construction utilizing FEM modeling. *Procedia CIRP* 2021; 104: 600-605. doi: https://doi.org/10.1016/j.procir.2021.11.101

6. Elkhuizen WS, Essers TTW, Song Y, et al. Gloss, Color, and Topography Scanning for Reproducing a Painting's Appearance Using 3D Printing. *Journal on Computing and Cultural Heritage (JOCCH)* 2020; 12: 1 - 22. doi: DOI:10.1145/3317949

7. Mahmoud D, Elbestawi MA. Lattice Structures and Functionally Graded Materials Applications in Additive Manufacturing of Orthopedic Implants: A Review. *Journal of Manufacturing and Materials Processing* 2017; 1(2). doi: 10.3390/jmmp1020013

8. 3MF Consortium. https://3mf.io/specification/; . [Online; accessed 2022].

9. Liu C, Jing S. Multi-attribute Voxelization Technology of AMF for 3D Printing. In: 2019 IEEE 6th International Conference on Industrial Engineering and Applications (ICIEA). IEEE. ; 2019; Tokyo, Japan: 81-85.

10. Xing J, Li F, Wang S, Wang Y, Zong G, Fan Y. Study on 3D Color Slicing Technology Based on OBJ Model. *Computer Science* 2020; 47(S2): 266-270+280. doi: 10.11896/jsjkx.20020002

11. Young G, Krishnamurthy A. GPU-accelerated generation and rendering of multi-level voxel representations of solid models. *Computers & Graphics* 2018; 75: 11-24. doi: https://doi.org/10.1016/j.cag.2018.07.003

12. Skylar-Scott MA, Mueller J, Visser CW, Lewis JA. Voxelated soft matter via multimaterial multinozzle 3D printing. *Nature* 2019; 575: 330-335. doi: https://doi.org/10.1038/s41586-019-1736-8

13. Huang SH, Zhang LC, Han M. An Effective Error-Tolerance Slicing Algorithm for STL Files. *Int J Adv Manuf Technol* 2002; 20(5): 363-367. doi: https://doi.org/10.1007/s001700200164

14. Huang R, Chae SI. Implementation of an OpenVG Rasterizer with Configurable Anti-Aliasing and Multi-Window Scissoring. In: The Sixth IEEE International Conference on Computer and Information Technology (CIT'06). IEEE. ; 2006; Seoul, Korea (South): 179-179.

15. Stratasys. http://www.stratasys.com/en; . [Online; accessed 2022].

16. Luu TH, Altenhofen C, Ewald T, Stork A, Fellner D. Efficient slicing of Catmull-Clark solids for 3D printed objects with functionally graded material. *Computers & Graphics* 2019; 82: 295-303. doi: https://doi.org/10.1016/j.cag.2019.05.023

17. Yao D, Yuan J, Tian J, Wang L, Chen G. Pigment Penetration Characterization of Colored Boundaries in Powder-Based Color 3D Printing. *Materials* 2022; 15(9): 3245. doi: 10.3390/ma15093245

18. Brunton A, Arikan CA, Urban P. Pushing the Limits of 3D Color Printing: Error Diffusion with Translucent Materials. *ACM Trans. Graph.* 2016; 35(1): 1-13. doi: 10.1145/2832905

19. Hu H, Zhang L, Zhang J, Wang Y, Shi Y. Fast Algorithm of Slicing Surface-color AMF Model. *Journal of Computer-Aided Design & Computer Graphics* 2017; 29(11): 2108-2116.

20. Xiao Y, Yang Y, Wang Y. The Algorithm and Application of Fast Surface Slice and Color Acquisition for Color 3D Printing. In: 2021 International Conference on Computer Engineering and Artificial Intelligence (ICCEAI). IEEE. ; 2021; Shanghai, China: 216-220.

21. Yang G, Liu W, Wang W, Qin L. Research on the Rapid Slicing Algorithm Based on STL Topology Construction. *Advanced Materials Research* 2010; 97-101: 3397 - 3402.

22. Gaol FL. Bresenham Algorithm: Implementation and Analysis in Raster Shape. *J. Comput.* 2013; 8(1): 69-78.

23. Liu H, Mo J, Chu A. An Algorithm of Creating Layered Bitmap in Rapid Prototyping of Three Dimensional Printing. *JOURNAL OF ENGINEERING GRAPHICS* 2009; 30(5): 63-68.

How to cite this article: Liu RY, Yang WD, Li HX, Wang YY, and Wang HY (2022), Color printing based on 3MF: color diffusion from the surface to the interior of voxel model, *Q.J.R. Meteorol. Soc.*, *2017;00:1–6*.