Article

Efficient Construction of Voxel Models for Ore Bodies Using an Improved Winding Number Algorithm and CUDA Parallel Computing

Lei Liu ¹, Yong Sun ¹,* 1, Min Ji ², Huimeng Wang ¹ and Jiantao Liu ¹

¹ School of Surveying and Geo-Informatics, Shandong Jianzhu University, Ji’nan 250101, China; 2021165122@stu.sdjzu.edu.cn (L.L.); wanghm@lreis.ac.cn (H.W.); liujiantao18@sdjzu.edu.cn (J.L.)

² College of Geodesy and Geomatics, Shandong University of Science and Technology, Qingdao 266590, China; jimin@sdust.edu.cn

* Correspondence: sunyong21@sdjzu.edu.cn

Abstract: The three-dimensional (3D) geological voxel model is essential for numerical simulation and resource calculation. However, it can be challenging due to the point in polygon test in 3D voxel modeling. The commonly used Winding number algorithm requires the manual setting of observation points and uses their relative positions to restrict the positive and negative solid angles. Therefore, we proposed the Winding number with triangle network coding (WNTC) algorithm and applied it to automatically construct a 3D voxel model of the ore body. The proposed WNTC algorithm encodes the stratum model by using the Delaunay triangulation network to constrain the index order of each vertex of the triangular plane unit. GPU parallel computing was used to optimize its computational speed. Our results demonstrated that the WNTC algorithm can greatly improve the efficiency and automation of 3D ore body modeling. Compared to the Ray casting method, it can compensate for a voxel loss of about 0.7%. We found the GPU to be 99.96% faster than the CPU, significantly improving voxel model construction speed. Additionally, this method is less affected by the complexity of the stratum model. Our study has substantial potential for similar work in 3D geological modeling and other relevant fields.

Keywords: 3D model; voxelization; ore body; winding number; triangulation network; CUDA

1. Introduction

Computer vision, computer graphics, and related fields have opened up new possibilities for analyzing data using geographic information systems (GIS). Three-dimensional modeling technology has revolutionized traditional 2D GIS into 3D GIS modeling [1]. Three-dimensional GIS modeling is an innovative technique that entails the construction of a model with geospatial data in a virtual 3D space using 3D software or development tools [2,3]. The application of 3D GIS in geological research is essential for geotechnical engineering and exploration work [4,5]. Geologists can use 3D visual models to display geological structures and visualize and analyze geological data effectively [6–8]. Complex ore bodies pose a challenge for 3D modeling, making these tasks a common difficulty in practical work [9–11].

When creating 3D models of ore bodies, two methods are commonly used: triangulation irregular network (TIN) and generalized tri-prism (GTP). These methods are crucial for modeling ore bodies and have been improved to address issues such as building complex models of ore bodies [12,13]. TIN has proven to be effective in creating surface models of strata [14,15]. On the other hand, GTP is great for quickly and accurately fitting complex geological bodies like faults and lenses [15]. However, GTP requires that the strata of the same geological layer be divided into multiple GTP units, which means that the GTP model has more data components than the TIN model. The Delaunay triangulation network is
a classic algorithm used in TIN, which is widely applied in various research fields. For example, it is applied to data analysis of meteorological stations in 2D GIS [16], the field of 3D printing, reinforcement of printed objects [17], and 2D data conversion to generate data keys for data encryption [18].

The consecutive and vector models have limitations in describing mining entities, and their application range is narrow. This makes it difficult to estimate mine reserves and simulate actual mining effects. To overcome these limitations, the geological block method is crucial for building an ore body model and analyzing geology data [19]. The essence of this method lies in the construction of voxel models, which are similar to pixels in three-dimensional space and comprise spatial voxels [20].

The 3D voxel model has good geometric characteristics, with adjacent voxels and flexible particle sizes. Masoumi et al. used the voxel model to simulate the mine in blocks to describe the mineral distribution with complex geometric boundaries [21]. The voxel unit can contain attribute information suitable for data analysis in 3D space. Navarro et al. used the model combined with the strength-grade factor in an overall rock factor to automatically assess rock structure, strength, and waste/ore identification [22]. The voxel model is conducive to providing an intuitive description of ore blocks planned to be caved over the life of the mine [23], which is beneficial to the application of the model in mining engineering. On the other hand, voxel model is used in many 3D modeling fields. Lei et al. [24] realized the construction of Chicago’s urban three-dimensional voxel model through elastic grid-scale voxel units. In addition, the application of voxels also includes medical workpiece processing and other fields [25,26].

Despite its poorer edge-fitting effect in comparison to the GTP model, each voxel unit of the 3D voxel model can record ore body data, such as mineral phase and content, making it convenient for data analysis. Therefore, constructing the voxel model is integral to the current mine application model construction [27]. For example, Chang et al. and Jia et al. used voxels to analyze and predict the distribution of ore-forming points of ore bodies in combination with mathematical models [28,29]. They constructed an information-based three-dimensional ore body model to realize mineralization information mining. In addition, Wang et al. compared the interpolation effects of different mineral contents through the 3D voxel ore body model [30]. It can be seen that the voxel model is beneficial to constructing and analyzing the three-dimensional model of the ore body. An essential problem to be solved in the voxelization of a 3D model is the point in polygon test.

The point in polygon has its roots in the early days of geometric graphics [31]. Today, as computer graphics have become essential in many fields, many scholars continue to discuss this problem [32,33]. With the application of three-dimensional geometry, the problem has expanded to include discussions of methods, improvements, and efficiency. It is also a research hotspot for applying three-dimensional point clouds [34]. Among the solutions to this problem, scholars often study the Ray casting method, also known as the odd-even method [35,36]. This method can make quick and intuitive judgments but requires discussing many specific issues. It can be challenging to summarize various exceptional cases of rays passing through three-dimensional geometry in the expansion of the three-dimensional space algorithm. Other methods, such as the Voronoi tessellations, can solve the point in polygon test in a convex polygon [37]. These methods can better handle the edges of graphics and exhibit better operational efficiency. However, these methods have limited effectiveness with non-convex polygons. Furthermore, three-dimensional geometry is more complex than two-dimensional plane graphics, and handling arbitrary polygons in three-dimensional space expansion is challenging.

The Winding number algorithm is a valuable geometric technique for solving the point in polygon test in 2D GIS. By calculating the winding number of a point around a polygon, it can determine whether the point is inside or outside the polygon. While scholars have explored the discrimination of 3D non-convex polygon models with self-intersections, holes, or degeneracies [38,39], the Winding number remains a mathematical concept that is still being studied in complex analysis. In this field, it counts the number of complex roots.
of a polynomial [40]. Kodama et al. [41,42] extended the Winding number algorithm into 3D space by summing solid angles to determine whether a point is inside or outside a non-convex polyhedron. This method can implement shape classification in 3D space by support vector machine (SVM). However, as the algorithm lacks a calculation method to determine the positive and negative of the solid angle based on the relative position of the observation point, it requires an artificial setup. While the Winding number algorithm is practical and easy to implement in 2D space, 3D space presents unique challenges due to the triangle face direction that cannot be unified by clockwise or counterclockwise rearrangement in 2D. Therefore, it is necessary to overcome these challenges when extending the algorithm to 3D space.

To address the drawbacks of the traditional Ray casting method and Winding number algorithm, our research introduces a novel method for network coding, Tri-Coding, which utilizes a triangulation model. Therefore, we proposed an improved Winding number with a Tri-Coding (WNTC) algorithm using the Delaunay triangulation model for voxelization model construction. The WNTC algorithm solves the problem by constraining the positive and negative values calculated by the internal and external directions of the triangular normal vector. The proposed WNTC algorithm eliminates the need for manual intervention in the modeling process, which makes it more conducive to 3D modeling program automation performance.

The proposed WNTC algorithm in this study uses many inverse trigonometric functions to model a three-dimensional ore body model. As a result, it is a computationally intensive process. Fortunately, advancements in GPU technology [43] have made it much more efficient than traditional CPU computing. To improve the process’s efficiency, we used GPU instead of CPU to execute some of the operations. In parallel programming, subroutines can perform functions independently, and appropriate thread allocation can improve CPU utilization and operation speed [44]. NVIDIA’s Compute Unified Device Architecture (CUDA) is a powerful computing platform that allows programs to transfer computing processes to GPU, reducing the time spent on tedious and repetitive computing [45,46]. This architecture can significantly boost the computing efficiency of large data volumes when other factors, such as CPU control time, occupy less computing time of the program.

In summary, our study offers an effective solution to the point in polygon test in 3D modeling. We proposed the WNTC algorithm and applied it to create practical voxel models of three-dimensional ore body models. Furthermore, we utilized the CUDA programming method to boost the computing speed of the WNTC algorithm, thereby improving the modeling efficiency of the model.

2. Materials and Methods

2.1. Data and Experimental Environment

The borehole data used in this study consist of 258 locations, and the boreholes are collected according to 9 nearly parallel planes and a total of 30 layers of data is collected. The boreholes are widely distributed, with a range of 1455.34 m in the x-direction, 1561.61 m in the y-direction, and a height difference of 263.67 m.

Once the surface model of each layer is constructed using Delaunay triangulation, it contains the most significant number of triangular faces in the layer of 1032 triangles. Assuming that the particle size of a voxel is set to $12 \times 12 \times 4$ m, approximately 1.02 million discrete points need to be processed during discretization. These points must be evaluated as point in polygon with different layer models.

2.2. Construction and Voxelization of Ore Body Surface Model

The ore body surface model was constructed by Delaunay triangulation. Based on the Delaunay model, the voxel model can be constructed by discretizing the model’s bounding box, obtaining the undetermined points, and judging the position relationship between
the binding points and the model body. The steps of the voxel modeling are as follows, as shown in Figure 1.

Calculate the position relationship between each point and the surface models. That is, take the point in polygon test which needs to be able to handle non-convex geometry. The standard methods are the Ray casting method and the Winding number algorithm.

The Ray casting method is challenging in three-dimensional space. When the ray passes through the intersection lines or the vertex of triangular surfaces, the number of faces the ray passes through will be an uncertain result and the odd-even result will become unreliable.

2.3. Improved Winding Number Algorithm with Tri-Coding

2.3.1. Winding Number Algorithm

The Winding number algorithm as shown in Figure 2 is a method to judge whether a point exists inside the polygon in two-dimensional space by calculating the sum of the angles between one polygon and one point. This method makes it easier to generalize the case mentioned in the above ray method, which is suitable for convex and non-convex polygons.

As shown in Figure 3, the extending Winding number algorithm in 3D can introduce the concept of a solid angle instead of a directed angle in the 2D plane to realize the expansion of the algorithm in 3D space.

The solid angle is formed between the point \( P \) and the triangular faces of the polyhedron to be measured in the Winding number algorithm in 3D, and the sum of the values of these solid angles gives the position relation of the point inside or outside the geometry. One solid angle \( S_i \) can be obtained by Formula (1), where the \( \theta \) is one dihedral angle constituting the \( S_i \).

\[
S_i = (\theta_{3i-2} + \theta_{3i-1} + \theta_{3i} - \pi) \times r^2 \tag{1}
\]

It is known that the surface area of a sphere with radius 1 \((r = 1)\) is \(4\pi\). Therefore, the cumulative value \( S \) of \( S_i \) is calculated, as shown in Figure 3. When \( P \) is inside the polygon (Figure 3a), \( S \) is precisely the surface area of the unit sphere, then \( S = 4\pi \). When \( P \)
is outside the geometry (Figure 3b), according to the above principle of the sum of solid angles, it is easy to know that the relation between the relative positions of the face $BCD$ and the other three faces is opposite. Its projected area on the unit sphere $P$ is equal to the sum of the projected areas of the other three faces, then $S = 0$. This method is not limited to tetrahedrons but also applies to any polyhedrons in 3D space. Therefore, the diagnosis formula can be given by the Formula (2):

$$S = \sum_{i=1}^{n} S_i = \sum_{i=1}^{3n} \theta_i - n\pi = \begin{cases} 4\pi, \text{ inside} \\ 0, \text{ outside} \end{cases}$$

(2)

$n$ in Formula (2) represents the number of triangular faces of the test polyhedron. Then the time complexity of the method is $O(n)$.

![Figure 2. Winding number in 2D. $P$ is the target point. $V$ is the vertex of the polygon. $\alpha$ is the directed angle of the directed edge, and the $a_2$ is clockwise, others are counterclockwise.](image)

![Figure 3. Winding number in 3D. Target point $P$ is inside (a) or outside (b) the tetrahedron $ABCD$ formed by the triangle faces. The shadow represents the projection of the corresponding solid angle.](image)

### 2.3.2. Tri-Coding

This algorithm is sensitive to edge order and requires vertices to be encoded in an order (clockwise or counterclockwise) in 2D. Unlike the two-dimensional polygon, it is challenging to generate the “same direction” of the spatial triangles through vertex ordering for three-dimensional geometry. This study uses the Delaunay triangulation network method to build the stratum surface model, and it is expected that this positive and negative relationship can be given through the regularity shown in Figure 4. Thus, our study proposes a method named Tri-Coding, which codes each triangular face to realize the constraint on the direction of normal vectors.
Algorithm 1, where the vertex sequences of the triangles in the figure are arranged counterclockwise, as shown by the red right-hand rule.

Figure 4. When the normal vector $\vec{v}$ of the space triangle $ABC$ points away from the point $P$ (a), $S_1$ is positive; otherwise, when $\vec{v}$ points to $P$ (b), $S_1$ is negative. The direction of $\vec{v}$ follows the right-hand rule.

In Figure 5, we can quickly know that for any two triangles with common sides, when their vertices are arranged in a counterclockwise or clockwise rotation, the direction of their common sides is reversed in the two triangles, and their normal vectors are pointing to one side. Therefore, we can encode an obj model data composed of triangle faces through the common edge relationship of adjacent triangles to obtain model data that can satisfy the calculation, that is the Tri-Coding algorithm. The algorithm’s pseudo-code is shown in Algorithm 1, where $\Delta^I$ is a data set of triangles read from the OBJ model. $\Delta^E$ is the encoded set from $\Delta^I$ by Tri-Coding. $\delta$ is one triangle from $\Delta^I$ and $\xi$ is its vertex id. $e$ is one edge of one triangle to judge the $\xi$ swaps order or not and $\zeta$ is its vertex id.

Figure 5. The relationship of the directed edges in a triangular pyramid expansion diagram (The vertex sequences of the triangles in the figure are arranged counterclockwise, as shown by the red and blue arrows in the figure, where the vertex sequences at the common sides of the triangles are in reverse order in their respective triangles. For instance, the edge $FD$ is $\vec{FD}$ in triangle $AFD$ and is $\vec{DF}$ in triangle $DFG$).

Regard the first triangle as the reference triangle, and the algorithm will recursively search the other triangle with common edge $e$. If the target triangle $\delta$ is found, the vertices of its common edge $e$ are judged to assess whether the vertex sequence of $\delta$ is the same as that of the reference triangle. If so, the two vertices id of the same common edge $e$ in the target triangular surface $\delta$ are exchanged. If the reverse is true, the swap is not performed.
Algorithm 1 Tri-Coding

1: procedure TRI CODING RECURSIVE (Δ₁, Δ₀, ε)
2: 
3: if there exists a triangle δ = (ξ₁, ξ₂, ξ₃) ∈ Δᵢ that contains edge e then
4:   Extract δ from Δᵢ
5:   if ξ₁ = ξ₂ and ξ₂ = ξ₃ then
6:     Δ₀ ← Δ₀ ∪ {δ}
7:     Call TRI CODING RECURSIVE on (Δᵢ, Δ₀, (ξ₁, ξ₃))
8:   end if
9:   else if ξ₂ = ξ₃ and ξ₁ = ξ₃ then
10:      Δ₀ ← Δ₀ ∪ {δ}
11:     Call TRI CODING RECURSIVE on (Δᵢ, Δ₀, (ξ₂, ξ₁))
12:     Call TRI CODING RECURSIVE on (Δᵢ, Δ₀, (ξ₁, ξ₃))
13:   end if
14:   if ξ₁ = ξ₃ and ξ₂ = ξ₂ then
15:       Update δ ← (ξ₂, ξ₁, ξ₃) and Δ₀ ← Δ₀ ∪ {δ}
16:       Call TRI CODING RECURSIVE on (Δᵢ, Δ₀, (ξ₁, ξ₃))
17:     Call TRI CODING RECURSIVE on (Δᵢ, Δ₀, (ξ₂, ξ₁))
18:   end if
19:   else if ξ₂ = ξ₁ and ξ₃ = ξ₂ then
20:       Update δ ← (ξ₁, ξ₂, ξ₃) and Δ₀ ← Δ₀ ∪ {δ}
21:       Call TRI CODING RECURSIVE on (Δᵢ, Δ₀, (ξ₁, ξ₂))
22:     Call TRI CODING RECURSIVE on (Δᵢ, Δ₀, (ξ₂, ξ₃))
23:   end if
24:   else if ξ₃ = ξ₂ and ξ₁ = ξ₂ then
25:       Update δ ← (ξ₂, ξ₁, ξ₃) and Δ₀ ← Δ₀ ∪ {δ}
26:       Call TRI CODING RECURSIVE on (Δᵢ, Δ₀, (ξ₂, ξ₁))
27:     Call TRI CODING RECURSIVE on (Δᵢ, Δ₀, (ξ₁, ξ₃))
28:   end if
29: end if
30: end procedure
31: procedure TRI CODING (Δᵢ)
32: 
33: input: Triangle set Δᵢ
34: initialize Δ₀ = ∅ and e as an arbitrary edge of any triangle in Δᵢ
35: call TRI CODING RECURSIVE on (Δᵢ, Δ₀, e)
36: output: Δ₀
37: end procedure

For example, in the part of the triangulation network in Figure 6, when Tri-Coding algorithm takes triangle V₁V₂V₃ and triangle V₁V₂V₆ as the reference triangle the two coding results of the vertex sequence and the normal vector direction obtained are indicated as shown in the columns of Figure 6b,c.
vector from a point to any point of the spatial triangle and the normal vector of the spatial vectors change as shown in the (a). When Tri-Coding algorithm takes triangle $V_1V_2V_6$ as the reference triangle, the normal vectors change as shown in the (b). And when the reference triangle is $V_1V_7V_8$, the normal vectors change as shown in the (c). Blue vectors are original, and the red vectors are encoded.

It can be seen that if the normal vector of the first triangle detected by the program is pointing outside the model. After coding, the other normal vectors point outside the model. Similarly, when the normal vector of the first triangle points inside, the other normal vectors will all point inside.

2.3.3. WNTC: Winding Number with Tri-Coding

Based on the Tri-Coding algorithm proposed in this study, the Winding number algorithm in 3D is improved, and named the Winding Number with Tri-Coding (WNTC) algorithm. In this algorithm, the normal vector of the spatial triangle forming the model has a normal vector with the same direction. At this time, the angle relation between the vector from a point to any point of the spatial triangle and the normal vector of the spatial triangle can be adopted. The signed value $S_i$ is calculated as shown in Figure 7, when the angle between $V_0$ and $\vec{V}$ is acute or obtuse.

**Figure 6.** Part of the triangulation network and normal vectors. the vertex order of the triangulation network is the subsequence of the vertex ordering given sequence $V_1 \rightarrow V_2 \rightarrow V_3 \rightarrow V_4 \rightarrow V_5 \rightarrow V_6 \rightarrow V_7 \rightarrow V_8 \rightarrow V_9$, the normal vector of each triangulation plane is with different direction (a). When Tri-Coding algorithm takes triangle $V_1V_2V_6$ as the reference triangle, the normal vectors change as shown in the (b). And when the reference triangle is $V_1V_7V_8$, the normal vectors change as shown in the (c). Blue vectors are original, and the red vectors are encoded.

**Figure 7.** The direction of $\vec{V}_0$ and $\vec{V}$. The red vector $\vec{V}_0$ is the vector formed by point $P$ pointing to any point on the space triangle $ABC$, and the blue vector $\vec{V}$ is the normal vector of the triangle. When the direction of $\vec{V}_0$ and $\vec{V}$ is close to the same, the angle is acute (a), and when the direction of $\vec{V}_0$ and $\vec{V}$ is close to the opposite, the angle is obtuse (b).

Introduce variable $s$, to mark the symbol of $S_i$. When $s = 1$, $S_i > 0$, and when $s = -1$, $S_i < 0$. The value of $s$ is as follows Formula (3).

$$s = \begin{cases} 
1, & 0 < \arccos\left(\frac{\vec{V}_0 \cdot \vec{V}}{|\vec{V}_0| |\vec{V}|}\right) < \frac{\pi}{2} \\
-1, & \frac{\pi}{2} < \arccos\left(\frac{\vec{V}_0 \cdot \vec{V}}{|\vec{V}_0| |\vec{V}|}\right) < \pi 
\end{cases}$$

(3)
Then combine the formula of solid angle $S_i$ (Formula (1)) with Formula (3) and let $r = 1$, so the value of $S_i$ is as follows:

$$S_i = s \times (\theta_{3i-2} + \theta_{3i-1} + \theta_{3i} - \pi)$$

(4)

As mentioned in Section 2.3.2, the Tri-Coding algorithm may cause that $S_i$ is all less than 0 or the cumulative value $S = -4\pi$ due to the direction of the spatial triangle used as the coding basis. In order to simplify the diagnosis, $S$ can be processed with absolute value in Formula (3).

$$S = \sum_{i=1}^{n} S_i = \begin{cases} 4\pi, & \text{inside} \\ 0, & \text{outside} \end{cases}$$

(5)

2.4. Improved Algorithm Efficiency Based on CUDA

Based on the proposed WNCTC algorithm, the CUDA parallel program is designed, and the overall process of the program is shown in Figure 8. In the experimental section of this study, a part of the model is tested. The number of discrete points in the bounding box is about 430,000, calculated with the stratum model data. According to the WNCTC algorithm, CUDA parallel program is designed.

Figure 8. CUDA program flow chart.

The input data can be expressed as two matrices $M_1$ and $M_2$ in Formulas (6) and (7), where $M_1$ is the matrix of target points, the size of which is $n \times 3$, $n$ is the number of the points, and each row of $M_p$ is the coordinate of each point $(x, y, z)$; $M_t$ represents the information of each triangular face of the three-dimensional geometry, the size is $m \times 9$, $m$ is the number of triangular faces of the geometry, and each row of $M_2$ represents the coordinates of the three vertices of each triangular face $[(x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3)]$.

$$M_p = \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ \vdots & \vdots & \vdots \\ x_n & y_n & z_n \end{bmatrix}$$

(6)

$$M_t = \begin{bmatrix} x_{11} & y_{11} & z_{11} & x_{12} & y_{12} & z_{12} & x_{13} & y_{13} & z_{13} \\ x_{21} & y_{21} & z_{21} & x_{22} & y_{22} & z_{22} & x_{23} & y_{23} & z_{23} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{m1} & y_{m1} & z_{m1} & x_{m2} & y_{m2} & z_{m2} & x_{m3} & y_{m3} & z_{m3} \end{bmatrix}$$

(7)

The Winding number algorithm can then be expressed as:

$$W(M_p, M_t) = M_S$$

(8)

In Formula (8), function $W$ represents the WNCTC algorithm for calculating the dihedral angle solid angles of points and triangular faces, and $M_S$ is a matrix of $n \times m$ size,
representing the result after calculating solid angles between \( n \) points and \( m \) triangles, respectively, each row represents a point, and each column corresponds to a triangular face.

\[
M_S = \begin{bmatrix}
S_{11} & S_{12} & \cdots & S_{1m} \\
S_{21} & S_{22} & \cdots & S_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
S_{n1} & S_{n2} & \cdots & S_{nm}
\end{bmatrix}
\]  \hspace{1cm} (9)

Then each row of \( M_S \) is summed, and the discriminant matrix is set to \( M_D \) in Formula (10) which derived from Formula (5).

\[
M_D^T = \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}_{1 \times n} \times M_S^T
\]  \hspace{1cm} (10)

During the above operations, \( S_{ij} \) operations are independent of each other, therefore, in the design of CUDA, the operation of \( S_{ij} \) is taken as the kernel function, and the thread grid and thread block are divided according to two-dimensional logic, so \( n \times m \) threads can be created, and each thread calculates a \( S_{ij} \).

The correlation operation of one \( S_{ij} \) is taken as an atomic operation, and its time complexity is set to \( O(1) \). Then, the time complexity to complete the voxel modeling of geometry is as follows:

\[
T = m \times n \times O(1) = O(m \cdot n)
\]  \hspace{1cm} (11)

The calculation time is obviously affected by the number of points and the complexity of the stratum model. Through the CUDA parallel computing design, our study takes advantage of its excellent performance of simultaneous computing with a lot of threads to improve modeling efficiency. Assuming that the number of thread grid and thread blocks used in CUDA programs are \( g \) and \( b \), respectively, \( g \times b \) threads will perform computation at the same time, and its time complexity is as follows:

\[
T = \frac{O(m \cdot n)}{g \cdot b}
\]  \hspace{1cm} (12)

When \( (m \cdot n) = (g \cdot b) \), the running time of this algorithm will be almost unaffected by the number of points to be measured and the complexity of the model, but relatively speaking, the space complexity will increase.

3. Results
3.1. Comparison of Methods

Figure 9a shows that a simple non-convex geometry was used to verify the improved effect of the proposed Winding Number with Tri-Coding (WNTC) algorithm. The information for the obj model of this geometry is presented in Figure 9b.

![Figure 9](image_url)

**Figure 9.** One single non-convex geometry \( V_1 V_2 V_3 V_4 V_5 \) (a) with the data (b). Points \( P_1 (1, 1, 0.3) \) and \( P_2 (0, 1, 0.3) \) are test points inside and outside the model, respectively.
As can be seen from Table 1, when the positive and negative solid angles are not distinguished, the $S_i$ values are all positive. It is impossible to distinguish the relative positive and negative relationships between $S_i$ corresponding to face 6 ($V_1V_2V_3$) located in the non-convex position of the geometry and other triangular faces, and the obtained summing results cannot correctly judge whether the point is inside or outside the geometry. And it indicates that through Tri-Coding, each $S_i$ will obtain the correct relative positive and negative relationship, and the cumulative value $S$ can be correctly calculated.

Table 1. The Calculation result $S_i$ and $S$ by Winding number algorithm with or without Tri-Coding in Figure 9.

<table>
<thead>
<tr>
<th>Triangle</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>$S=\sum S_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>without Tri-Coding</td>
<td>$S_i (P_1)$</td>
<td>4.758</td>
<td>3.646</td>
<td>3.646</td>
<td>0.471</td>
<td>0.471</td>
<td>0.426</td>
</tr>
<tr>
<td>$S_i (P_2)$</td>
<td>3.237</td>
<td>1.618</td>
<td>1.618</td>
<td>1.618</td>
<td>1.618</td>
<td>3.237</td>
<td>12.946 $\neq 0$</td>
</tr>
<tr>
<td>with Tri-Coding</td>
<td>$S_i (P_1)$</td>
<td>-4.758</td>
<td>-3.646</td>
<td>-3.646</td>
<td>-0.471</td>
<td>-0.471</td>
<td>0.426</td>
</tr>
<tr>
<td>$S_i (P_2)$</td>
<td>3.237</td>
<td>-1.618</td>
<td>-1.618</td>
<td>-1.618</td>
<td>-1.618</td>
<td>3.237</td>
<td>0</td>
</tr>
</tbody>
</table>

3.2. The Voxel Models Effect

As shown in Figure 10, compared with the proposed WNTC algorithm in this study (Figure 10c), the voxel model generated by the Ray casting method has certain defects (the red circle in Figure 10b). The specific differences are described in detail in Figure 11. And the JC03, KC03-52, and KC03-6 indicate a number to distinguish different formations in Figure 11.

Figure 10. A part of the model with different modeling methods showing the effect diagrams of the explicit model of some stratum (different colors indicate different stratum) (a), the voxel models constructed by Ray casting method (b) and the proposed WNTC algorithm in this study (c). The missing voxels in the models constructed by Ray casting method have been marked in red circles.
Figure 11. The model effect of some single strata, the effect diagrams of the explicit model and the voxel models constructed by the Ray casting method and the WNTC algorithm in this study. The missing voxels in the models constructed by Ray casting method have been marked in red circles.

Figure 11a–c show the explicit model effects of the JC01, KC03-52, and KC03-6 strata, respectively. Figure 11d–f illustrate the voxel model created using the Ray casting method. The number of voxels included are 18,674, 12,376, and 13,604, respectively, and the voxel size is $12 \times 12 \times 4$ m. It can be seen from the model effect that some voxels are missing.

On the other hand, Figure 11g–i display the voxel models generated by the WNTC algorithm proposed in this study using the above stratum models. The number of voxels included are 18,681, 12,411 and 13,711, respectively. It is evident from the model effect that the voxel model generated by the proposed WNTC algorithm is more accurate.

Compared with the model developed by the WNTC algorithm, the model built by the Ray casting method requires an additional 107 voxels. This means that the model developed by the Ray casting method has some missing voxels. In contrast, the WNTC algorithm proposed in this study can compensate for a voxel loss of about 0.7%, effectively improving the accuracy of the voxel model.

3.3. Comparison of Efficiency

To effectively compare the efficiency improvement effect of CUDA parallel computing, the experiment conducted measurements on about 430,000 points under both CPU and GPU, as shown in Table 2. For different parts of the stratum, the proposed WNTC algorithm was used for each group of experiments. The time taken for sequential operations in the
CPU and parallel processes in the GPU were calculated. The results in Table 2 are sorted in ascending order by the number of model vertices and model triangle faces. And Figure 12 shows the running time of CUDA application design on GPU and CPU.

Table 2. Running time of voxel modeling in different stratum models by the CUDA.

<table>
<thead>
<tr>
<th>Group</th>
<th>Code of Stratum</th>
<th>Number of Points</th>
<th>Number of Triangles</th>
<th>Running Time in CPU (s)</th>
<th>Running Time in GPU (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>KC04-13</td>
<td>228</td>
<td>312</td>
<td>32.52</td>
<td>0.003</td>
</tr>
<tr>
<td>2</td>
<td>JC04</td>
<td>242</td>
<td>320</td>
<td>33.126</td>
<td>0.003</td>
</tr>
<tr>
<td>3</td>
<td>JC05-1</td>
<td>238</td>
<td>348</td>
<td>36.134</td>
<td>0.003</td>
</tr>
<tr>
<td>4</td>
<td>KC03-3</td>
<td>248</td>
<td>348</td>
<td>36.086</td>
<td>0.002</td>
</tr>
<tr>
<td>5</td>
<td>KC03-51</td>
<td>266</td>
<td>356</td>
<td>36.861</td>
<td>0.002</td>
</tr>
<tr>
<td>6</td>
<td>KC03-12</td>
<td>262</td>
<td>372</td>
<td>38.564</td>
<td>0.003</td>
</tr>
<tr>
<td>7</td>
<td>KC02-2</td>
<td>256</td>
<td>376</td>
<td>38.896</td>
<td>0.003</td>
</tr>
<tr>
<td>8</td>
<td>KC03-53</td>
<td>278</td>
<td>380</td>
<td>39.406</td>
<td>0.003</td>
</tr>
<tr>
<td>9</td>
<td>JC02</td>
<td>270</td>
<td>396</td>
<td>41.012</td>
<td>0.003</td>
</tr>
<tr>
<td>10</td>
<td>KC03-13</td>
<td>296</td>
<td>432</td>
<td>44.849</td>
<td>0.002</td>
</tr>
<tr>
<td>11</td>
<td>KC02-1</td>
<td>292</td>
<td>460</td>
<td>47.64</td>
<td>0.002</td>
</tr>
<tr>
<td>12</td>
<td>JC01</td>
<td>314</td>
<td>460</td>
<td>47.544</td>
<td>0.003</td>
</tr>
<tr>
<td>13</td>
<td>KC03-1</td>
<td>300</td>
<td>476</td>
<td>49.418</td>
<td>0.002</td>
</tr>
<tr>
<td>14</td>
<td>KC03-52</td>
<td>464</td>
<td>740</td>
<td>76.895</td>
<td>0.002</td>
</tr>
<tr>
<td>15</td>
<td>KC04-2</td>
<td>484</td>
<td>752</td>
<td>78.447</td>
<td>0.003</td>
</tr>
<tr>
<td>16</td>
<td>KC03-6</td>
<td>576</td>
<td>924</td>
<td>95.833</td>
<td>0.003</td>
</tr>
<tr>
<td>17</td>
<td>JC03</td>
<td>588</td>
<td>944</td>
<td>97.605</td>
<td>0.002</td>
</tr>
<tr>
<td>18</td>
<td>KC03-11</td>
<td>616</td>
<td>988</td>
<td>102.428</td>
<td>0.003</td>
</tr>
<tr>
<td>19</td>
<td>DB</td>
<td>636</td>
<td>1032</td>
<td>107.361</td>
<td>0.7</td>
</tr>
</tbody>
</table>

Figure 12. The running time of CUDA application design on GPU and CPU.

According to the data presented in Figure 12 and Table 2, it is evident that the CUDA application runs much faster on GPU than on CPU. The results of the tests show that, on average, the GPU takes up only 0.04% of the time taken by the CPU to complete the computation. Moreover, the highest ratio of GPU running time to CPU running time is a mere 0.65%, which tremendously improves the overall efficiency and is not significantly affected by the number of triangle faces.

4. Discussion

When modeling ore bodies, there are two common methods: implicit construction and explicit modeling. With implicit construction, an implicit function is used to represent
the surface of the stratum, which covers the surface points of the ore body model. Explicit modeling, on the other hand, involves determining vertex coordinates and establishing connections between them, such as fitting triangular faces to the geological stratum surface. This results in a regular and easily manageable model. When building the voxel model, it is necessary to judge the point in polygon test. In the geometric method, it is easy to calculate the position relationship between the point and the space triangles (implicit function). The main issue discussed in this study is the construction method of the voxel model of ore bodies, so explicit modeling methods are used for research and validation.

Compared with the existing 3D ore body voxel modeling research, this paper presents an automated modeling scheme using the Winding number algorithm. Through the design and application of the Tri-Coding algorithm, the modeling process no longer needs artificial control.

The limitations of the methods proposed in this paper could be argued as follows:

1. The Winding number with the Tri-Coding algorithm proposed in this paper only discusses the structure of the model, which is composed of triangular faces. When constructing a computer visualization model, polygons outside of triangles are allowed. The parameter structure must be adjusted accordingly if the algorithm is applied to a model constructed by polygons with more than three edges. This may involve increasing the number of calculations for the solid and dihedral angle. The Winding number algorithm in 3D provides a more accessible alternative than the traditional ray casting method. However, the WNTC algorithm, which relies on inverse trigonometric functions, demands higher operation accuracy.

2. Our study’s GPU thread grid and block are relatively direct. The number of triangular faces of the stratum model used in this experiment is up to about 1000, which does not exceed the hardware environment limit of this experiment, so the number of triangles is directly used as the number of GPU thread grids to verify the effect of CUDA parallel computing design on the improvement in program running speed.

Given the above limitations, the following possible effective solutions are proposed:

1. When the model body is made up of polygons other than triangles, additional steps can be taken to adapt the algorithm using the abovementioned method. Alternatively, the polygons can be divided into triangles to meet the algorithm’s parameters outlined in this paper. The latter method is more straightforward. However, it may increase the amount of data in the model depending on the number and structure of the non-triangular polygons. Handling calculation accuracy with care during program design is crucial to avoid any mistakes in the final judgment.

2. When a model consists of a large number of triangular surface elements that exceed the capability of the GPU’s hardware environment to handle thread meshes, the required threads can be grouped using software control. The CUDA parallel computing process is then performed, with part of the host memory allocated to set registers, save pre-calculated results, and complete the operation via CPU coordinated control.

5. Conclusions

In this study, we propose the Winding number with Tri-Coding (WNTC) algorithm, which combines Tri-Coding and the Winding number algorithm in 3D. To efficiently voxel model 3D explicit models, our study also combined CUDA program design and optimized the implementation process of the proposed WNTC algorithm. Through theoretical analysis and experimental verification, we found that the proposed WNTC algorithm can better constrain the positive and negative winding number during voxel modeling. It is better and faster than the traditional Ray casting method in the voxelization of the model.

The innovation offered in this paper lies in the design of a triangulation network coding (Tri-Coding) for the Winding number algorithm based on the following rules: Through the sequence relationship of the vertices of two triangles with common edges in the storage structure of the computer program, it is found that when their vertices are surrounded in a clockwise or counterclockwise direction, the common edges are arranged
in the opposite order in their respective vertex sequence lists. The algorithm realizes the constraint of the normal vector of the triangular surface of the model so that it points to the internal and external of the model to improve the Winding number algorithm in 3D.

This is conducive to more adequately describing the spatial relationship between points and triangles in the Winding number algorithm. The Winding number in the 3D algorithm no longer needs to deal with the positive and negative association of the solid angle in the operation process. It solves the problem of the computer’s weak perception of the relative position relationship between points and polyhedrons in 3D space. In addition, it makes the model’s surface in 3D space directional, which is conducive to the expansion of certain 2D geometric processing algorithms in 3D space.

The proposed WNTC algorithm suits any 3D model with an explicit surface model structure. Additionally, our proposed method is not limited to voxel modeling of the ore body model and has strong applicability for other application fields with general explicit model voxel requirements.

In future research, we will discuss the data processing method of drilling data under specific conditions. If there are complex geological conditions, such as faults and lenses, in the region, more geological exploration data should be introduced. We will also carry out operations on the drilling data to improve the model shape.

Author Contributions: Conceptualization, Yong Sun; Data curation, Yong Sun and Min Ji; Formal analysis, Huimeng Wang; Investigation, Yong Sun and Lei Liu; Methodology, Yong Sun and Lei Liu; Project administration, Yong Sun and Jiantao Liu; Resources, Min Ji; Supervision, Huimeng Wang; Writing—original draft, Yong Sun and Lei Liu; Writing—review and editing, Huimeng Wang. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the No.1 Institute of Geology and Mineral Resources of Shandong Province (No. 2022DW02), the National Natural Science Foundation of China (No. 42171113), and the Shandong Natural Science Foundation Youth Program (No. ZR2021QD113).

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available due to the confidentiality requirements of the project source.

Conflicts of Interest: The authors declare no conflict of interest.

References
2. Wu, A.; Che, T.; Li, X.; Zhu, X. A ship navigation information service system for the Arctic Northeast Passage using 3D GIS based on big Earth data. Big Earth Data 2022, 6, 453–479. [CrossRef]
5. Pirot, G.; Joshi, R.; Gruaud, J.; Lindsay, M.D.; Jessell, M.W. LoopUI-0.1: Indicators to support needs and practices in 3D geological modelling uncertainty quantification. Geosci. Model. Dev. 2022, 15, 4689–4708. [CrossRef]


17. Qiao, Y.; Lv, N.; Ouyang, X. Variable density filling algorithm based on delaunay triangulation. *Micromachines* 2022, 13, 1262. [CrossRef]


23. Cetin, M.C.; Klein, B.; Li, G.; Futcher, W. Tracking grade heterogeneity in a panel cave mine: A reconciliation study investigating the impact of mixing from an ore sorting perspective. *Minerals* 2023, 13, 1333. [CrossRef]


29. Jia, F.; Su, Z.; Nian, H.; Yan, Y.; Yang, G.; Yang, J.; Shi, X.; Li, S.; Li, L.; Sun, F.; et al. 3d quantitative metallogenic prediction of indium-rich ore bodies in the dulong Sn-Zn polymetallic deposit, Yunnan Province, SW China. *Minerals* 2022, 12, 1591. [CrossRef]


32. Hao, J.; Sun, J.; Chen, Y.; Cai, Q.; Tan, L. Optimal Reliable Point-in-Polygon Test and Differentiating Coding Boolean Operations on Polygons. *Symmetry* 2018, 10, 477. [CrossRef]


39. Kumar, G.N.; Bangi, M. An Extension to Winding Number and Point-in-Polygon Algorithm. *IFAC-Pap.* 2018, 51, 548–553. [CrossRef]


42. Kodama, S. Shape classification based on solid angles by a support vector machine. *Intell. Data Anal.* 2022, 26, 933–957. [CrossRef]
43. Bagies, T.; Le, W.; Sheaffer, J.; Jannesari, A. Reducing branch divergence to speed up parallel execution of unit testing on GPUs. *J. Supercomput.* 2023, 79, 18340–18374. [CrossRef]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.