Fast projection ray algorithm for 3D solid space segmentation

Qiankun Wu1*, Jiarun Wang1, Yongliang Zhang1 and Yonghong Li2
1North China Institute of Computing Technology, Beijing, 100083, China
2Academy of Military Sciences, Beijing, 100091, China
*Corresponding author’s e-mail: kyzx15@qq.com

Abstract: Spatial volume element segmentation of 3D solids has a wide range of applications, but its computation is generally time-consuming. Therefore, a fast projection ray algorithm for spatial segmentation of 3D solids is proposed, which is based on a 3D to 2D projection strategy and uses a ray intersection with the 3D body surface for acceleration. The experimental results show that the algorithm can accelerate up to 20,000 times faster than the traditional octree, and can complete the segmentation of millions of spatial volume elements within seconds.

1. Introduction

The 3D solid model can completely express all the shape information of an object and can determine without ambiguity whether a point is outside, inside or on the surface of an object. This model can further meet the requirements of physical calculations, finite element analysis and other kinds of practical applications. With the development of high-precision tilt photography and true 3D display technology, the accuracy of solid models displayed in 3D GIS is getting higher and higher, and with it comes the urgent need for rapid analysis and calculation of 3D solid models in geospatial space. Although high-performance parallel computing can be used for performance enhancement, many complex calculations cannot be task-parallel divided, making it impossible to use parallel techniques. The representation and analysis of 3D entities needs to break through the barrier of high time consumption in vector data computation, and new technical implementations need to be explored.

In 2D vector data, rasterization is often used to approximate 2D vector targets to speed up spatial analysis calculations. Compared to vector data rasterization has the following advantages: (1) some spatial analyses (e.g. superposition analysis, buffer analysis), which are very complex to implement with vector methods but very simple to implement with raster methods, are converted from vector data to raster data and then the results are transferred back to vector data; (2) the raster data is neatly arranged, which has the inherent advantage of providing a simplified method for complex calculations in vector data. This inherent advantage provides a simplified approach to complex calculations in vector data. Network analysis, which is traditionally considered to be the specialty of vector methods, can also be converted to raster methods and then converted to vector types after performing calculations using "water injection models" and "simulated annealing models,"[1-2] (3) in terms of plotting, raster data is more suitable for (4) statistics are usually processed in regional zones, and many spatial analyses require the distribution of these statistics to a geographic grid; (5) raster data have the advantage of zero initialization.[3] If rectangular pixels in two-dimensional space are replaced by cubic pixels in three-dimensional space, and then the body targets in three-dimensional are simulated and approximated, a rasterized representation of the three-dimensional object model can be achieved. With the development of GIS, scholars have conducted a lot of research on the rasterisation of 2D vector
data, such as DDA method, Bresenham algorithm, internal point diffusion method, ray method, scanning method, boundary algebra method. but there is no relatively complete and effective method for the rasterisation of solid models in 3D space.

In response to the above problems, a new fast projection ray \[^{[16]}\] algorithm (FPR) for 3D entity spatial segmentation is proposed, following the 2D vector data rasterisation method and borrowing the idea of grid dissection from Digital Earth \[^{[4,5]}\], which can perform fast body element approximation of 3D entities and realize 3D rasterized discrete representation of entity models in 3D space. This 3D rasterization technique can greatly support the effectiveness of spatial analysis in the field of GIS.

2. Overview of spatial octree partitioning
Spatial octree partitioning \[^{[6,7,8,13]}\] is a traditional method of spatial non-uniform meshing which partitions a spatial cube containing an entire three-dimensional entity into eight sub-regions of the Cartesian coordinate system into eight sub-cube meshes, which are organized into an octree. If the grid size of a sub-cube is not reached, further partitions are made for that sub-cube until the grid size is less than a given threshold. This algorithm takes advantage of spatial coherence, with the segmentation resulting in child nodes that are continuous and gap-free in the parent spatial domain.

2.1 Plain octree node space partitioning methods
In this paper, the plain octree node space partitioning method has been selected, the specific practice of this method is to bifurcate the octree nodes again in 3D space and then in the direction of each coordinate axis directly, its practice is in line with the common octree partitioning algorithm, with the following advantages.

1) Fast location: the recursive nature of cubic lattice entity construction allows for fast location of the cubic lattice entity where the target is located.
2) Fast subdivision: recursive construction of cubic lattice entities is used to specify cubic lattice entities for fast subdivision.
3) Fast merging: due to the recursive nature of cubic lattice entity construction, specified cubic lattice entities are quickly merged for further comprehensive analysis.
4) Uniform coding: uniform rules for coding cubic entities in a spatial context, with the possibility of defining a set of convenient and rational algebraic operations.
5) pruning optimization: pruning can be used to simulate arbitrary 3D shaped regions in the 3D octree space, providing the ability to handle 3D rasterization of 3D entities.
6) Scale adaptation: the cubic lattice entities can be subdivided arbitrarily depending on the level and are extremely adaptable to the scale of the actual analysis space.

2.2 Traditional Octagonal Grid Space Partitioning Algorithm
The traditional octree spatial partitioning method makes use of the recursive octree nature of spatial partitioning. First, according to the AABB enveloping box of the three-dimensional body \[^{[14]}\] the smallest spatially partitioned body region that can contain this range is constructed, and then on the basis of this spatial region, recursive spatial partitioning is carried out to determine whether the node's cube intersects with the three-dimensional body's enveloping box (the intersection relationship includes the inclusion relationship), and if it does, it is retained, and if not, it is discarded. The set of cubes is then traversed, and the 8 vertices of each cube are judged to be located inside or outside the 3D body as the basis for retaining or discarding the cubes. If the vertices of the cube are located inside the 3D body, the cube should be considered as a spatially partitioned node of the 3D body, and if the vertices are all outside the 3D body, the cube is discarded as a spatially partitioned node of the 3D body. According to this method in the octree space range of three-dimensional body partitioning, also through the spatial partition pruning to speed up the efficiency of the partition, that is, the last level of granularity if the body lattice to meet the discard conditions, then its subsequent sub-body lattice all discard no longer applied to solve the three-dimensional body partition node.
2.3 Design of octonion coding rules for split spaces

To facilitate the description of the algorithm in this paper, the different granularities of spatial segmentation are set into levels in this paper, for example, the segmentation of a given cube or rectangle is carried out on a block, with the initial state called level 0, and the level added by 1 for every octant subdivision, and so on, to set the level. The partitioned small grid of dissected blocks is called a volume element or volume lattice [15].

The following model will be used in the subsequent description of the algorithm, with the entire range of the space in which the 3D body is located, set from the origin \( O(0,0,0) \) to \( D(x_{\text{Max}}, y_{\text{Max}}, z_{\text{Max}}) \). The segmentation level is described by \( n(n \geq 0) \). Similarly, in order to effectively describe the volume lattice in the partition space, the volume lattice is encoded in this paper. In this paper, the following encoding method is used: the partition space is divided into three directions, which are labelled as \( X, Y, Z \) axes, and the side length of the cube is taken as one unit, i.e. the total side length of the partitioned space \( 1/2^n \), and the cube encoding is labelled as \( \text{code}(x, y, z)(0 \leq x, y, z < 2^n) \), and in particular, in order to make the spatial area occupied by a lattice under the level \( n \) unique and without duplication, the corresponding spatial domain of the lattice is called, i.e. each of its axes satisfies In particular, in order to make the spatial region occupied by a lattice of cubes under level \( n \) unique and without duplication, the lattice \( \text{code}(x, y, z) \) corresponds to a spatial domain called \( \text{cube}(x, y, z) \), which occupies a spatial region that is closed under and open above, i.e. each of its axes satisfies \( \text{code} \times n_{\text{div}} \leq P < (\text{code} + 1) \times n_{\text{div}} \), where \( n_{\text{div}} \) is the length of the side of the lattice along that axis (follows similarly), \( \text{code} \) the value is the coded value along the axis, and \( P \) is the component of the spatial coordinate value of any point of the lattice in that axis.

3. Fast projection ray algorithm for 3D solid space segmentation

In computers, there are various ways to store 3D solid models, the most popular of which is to store the information of triangular facets on its surface, so the input object of the algorithm FPR in this paper is the set of triangular facets on the surface of the 3D body. The algorithm in this paper is divided into three stages: 1) to find the set of spatially-segmented cubes intersecting the surface of the 3D body; 2) to find the set of spatially-segmented cubes intersecting the interior of the 3D body; 3) to fuse and de-duplicate the set of cubes from these two stages to obtain the final set of cubes without any omission. The spatially segmented cube set for the intersection of the surface of the 3D body is calculated separately for each triangular surface piece, and the set of cubes intersecting each triangular surface piece is calculated separately to form the set of cubes intersecting the surface of the 3D body. In order to find the spatial partition of the intersection of the three-dimensional body, we first find the three-dimensional body enclosing box, project the box to the coordinate plane, and design a ray set, and seek the intersection of this ray set and the three-dimensional body surface, through the intersection information to calculate and three-dimensional body internal intersection of the body element collection. Finally, the fusion and de-duplication process is carried out to obtain the last result.

3.1 Method of implementing a mesh segmentation of 3D body surface elements

In order to accurately solve for the set of all cubes intersecting the 3D body surface in the partition space at a given partition level \( n \), the method adopted in this paper is to first traverse the entire set of triangular facets of the 3D body surface, first calculate the set of cubes intersecting a single triangular facet, and then merge the set of all intersecting cubes of the triangular facet set into one set, sort and de-weight (since a single cube can intersect multiple triangular facets) to obtain the set of spatially partitioned cubes intersecting the 3D body surface. The set of spatially-segmented cubes intersecting the surface of a three-dimensional body is obtained. All triangular slices are similar, so one triangular slice is chosen to illustrate the process. Let the triangulated slice to be processed be \( S_\Delta R_1 R_2 R_3 \), then...
the processing of this triangulated slice is carried out in two steps. The first step is to find the set of cubes that intersect the sides of the triangular surface piece, using the sides $P_1P_2$ as an example, and calculating the remaining two sides in the same way. The first step is to calculate the code of the lattice where the two endpoints are located by $\text{code}(P) = P / \text{div}_n$, dividing by, rounding down if there is a decimal. If the two endpoints of the line are equal, the endpoint code is returned, otherwise the second step is performed. The second step, shown in Figure 1, is the projection of the calculated line segment on a coordinate plane, where it is $P_1$ located in the top right-hand corner of Figure 1, with the projection block of the lattice marked in light yellow. As can be seen from the projection diagram, there is a straight line between the two projection endpoints formed by a column or row of lattice projection edges, and in three dimensions there is a similar lattice surface stitched together in a plane between the two endpoints. The following is an example of how to calculate a lattice code for the intersection of a line segment with an $X$ axis. First, if there are integers that $i$ satisfy: $1 \leq i \leq n$ or $2 \leq i \leq n$ (i is an integer, $P(x)$ is the coordinate component of point $P$ on the $X$ axis), then calculate the intersection of the plane $y = j \ast \text{div}_n$ with the line, let the value of the intersection $Z$ be $(i \ast \text{div}_n, c_y, c_z)$, and get the body metric code through which the line passes.

$\text{code} \left( i, \frac{c_y}{\text{div}_n}, \frac{c_z}{\text{div}_n} \right), \text{code} \left( i-1, \frac{c_y}{\text{div}_n}, \frac{c_z}{\text{div}_n} \right)$.

(The calculation only requires one of the line segments to be positioned in order to avoid double counting). As the line segment enters and exits a new lattice, it must intersect the surface of the lattice, so the method ensures that all lattices intersected by the line segment are solved for. In Figure 1, the small quadrilateral is the projection of the lattice, where the orange color is the lattice where one of the endpoints is located, the indigo color is the result of the intersection of the line segment with the horizontal line (projection of the $xoy$ plane $y = j \ast \text{div}_n$ in the plane of the calculated lattice), and the light purple color is the result of the intersection of the line segment with the vertical line (projection of the plane $x = i \ast \text{div}_n$ in the plane $xoy$).

![Figure 1. Coordinate plane projection of a line segment.](image1)

![Figure 2. Diagram of the set of rays selected for intersection with a triangle.](image2)

The second step is to calculate the metric lattice intersecting the interior of the triangle. This method requires the projection of the triangle onto each of the three axis planes, and since the projection to each axis is solved in a similar way, this paper will only use $xoy$ the plane as an example. First, all three vertices of the triangle are projected onto $xoy$ the plane set as $P_{x1}$, $P_{x2}$, $P_{x3}$, where those points would be satisfied with $P_{x1}(x) \leq P_{x2}(x) \leq P_{x3}(x)$, what’s more if the relationship couldn’t be satisfied, it needs to be solved by ordering those vertices’ $x$ coordinate value. First, if there exists a
integer $i$ between $P_{o1}$ and $P_{o2}$ to meet $P_{o2}(x) < i^\ast \hat{d}v_n \leq P_{o2}(x)$, then calculate the $y(i, j)$ values of the intersection of the line $x = i^\ast \hat{d}v_n$ with the $\ell_{P_{o1}P_{o3}}$ and $\ell_{P_{o1}P_{o2}}$ at $y_1, y_2$, respectively, in the plane of projection; if the conditions ($y_1 < y_2$) are not met, exchange $y$ values, and if there are integers $j$ which satisfies $(i, j) y_1 < j^\ast \hat{d}v_n < y_2$, a ray which parallels to Z axis from $(i^\ast \hat{d}v_n, j^\ast \hat{d}v_n, 0)$ can cross the triangle at a point, as shown in Figure 2. For ease of description, this ray is called the ray $(i, j)$, and is denoted by $rad(i, j)$, which forms the set of rays of the triangle on the projection plane. Once the set of rays has been obtained, the intersection of each ray with the triangular surface is found by setting the value of the Z coordinate component of the intersection $y$ with the triangular surface to $z$, set $k = z/\hat{d}v_n$, then the triangular surface must intersect the body element $code(i-1, j-1, k)$, $code(i, j-1, k)$, $code(i-1, j, k)$ and $code(i, j, k)$ (because $rad(i, j)$ the triangle intersects the four body elements because it passes through the common prism of the four body elements and the intersection point is inside the triangle). The method of finding the intersection between and $P_{o3}P_{o2}$ has same procedures to solve. After traversing the set of projection rays in the three coordinate planes in this way, the set of all lattices that intersect only the interior of the triangle can be found.

The principle of this method is explained as follows: if a lattice does not intersect the sides of a triangle, but only its interior, the lattice is divided into two parts by the triangle, then the interior of the triangle intersects the cube of the lattice, and the projection of this cube to its parallel axis is a point located inside the projection of this triangle. All these possible projection points of the cube’s edges of the lattice have been identified from the three coordinate plane projections earlier in this paper, so all lattices that intersect only the interior of the triangle can be calculated in this way. In Figure 2, the set of light blue perpendicular lines is an example of a ray set which, can be seen in the figure, emits rays that intersect the triangle in its interior.

However, as there are duplicate lattices in the calculation result, it is necessary to sort and remove these duplicate lattices in the subsequent processing to obtain an ordered set of lattices that intersects the triangular surface, for example, the surface of the 3D body. Since this method has considered both the intersection with the triangular edges and the intersection with the interior of the triangular surface sheet, it is guaranteed that the set of lattices is the set of all lattices that intersect the surface of the 3D body. The main design idea of this stage is to ensure that no body elements are missed in the 3D body space segmentation and to theoretically guarantee the integrity of the 3D space segmentation results.

3.2 Method of implementing 3D internal body element mesh segmentation

In the previous subsection, the set of all lattices that intersect the surface of a 3D body has been solved for by traversing the information on the surface of the 3D body, but there are still lattices that intersect only the interior of the 3D body that have not been solved for. If the lattice only intersects the interior of the 3D body, then all points of the lattice will locate in the interior of the 3D body. Therefore, in order to find these lattices that only intersect the interior of the 3D body, this paper sets a characteristic point for the lattice, for example, the centroid of the lattice, and only needs to find all lattices whose centroids lie inside the 3D body, and the set of lattices that intersect the interior of the 3D body must be its subset. Following the example of the previous paper on the calculation of the intersection of the triangular surface sheet with the lattice, it is also possible to provide ideas for the subsequent solution of the set of lattices intersecting the interior of the three-dimensional body. In other words, to solve for the set of intersecting lattices with the interior of a three-dimensional body, it is possible to construct a set of intersecting rays with a three-dimensional body by following the way of constructing a ray set, which only needs to pass through the center point of the lattice, as follows.

The first step is to build the ray set. Let the set of triangular slices on the surface of the input spatial 3D body be $S$, and according to $S$, find its $AABB$ enclosing box as $box$. First, select the projection
plane. In this paper, the \( xoy \) plane is used as the projection plane, so the enclosing box \( box \) is projected as a rectangle set to \( Q_b \), as shown by the yellow rectangle in Figure 3.

It is obviously that the rectangular block contains the coordinate plane projection points \( Y \) (shown as red dots in the diagram) of the center of the body element.

**Figure 3.** Set of ray selections determined by the projection of the 3D body enclosing the box. **Figure 4.** Subset of rays selected from the set of rays intersecting the triangle

In the second step, traverse all the triangular face pieces on the surface of the 3D body and set the projection \( S_{aR_1R_2} \) in the plane \( xoy \) to \( S_{aR_0R_2R_3} \), as shown in the blue triangular projection on the left-hand side of Figure 4. The interior of the triangular projection may contain a subset of the above ray set. With the same tactic as previous subsection, all ray starting points inside the triangular projection are counted here in a similar way, as shown in Figure 4, where the interior of the triangular projection contains ray starting points in red. The next step is to calculate the intersection of the triangular face piece with the corresponding ray and store it in the set of intersections where this ray intersects the surface of the 3D body. Let \( \text{rad}(i, j) \) be a ray starting point located inside the triangle, then its index in the set of intersection points created based on the ray set is calculated as

\[
\text{index}(i, j) = \text{code}(i_x) - \text{code}(	ext{radsetMinx}) + \text{radsetXlen} \ast \left( \text{code}(j_y) - \text{code}(	ext{radsetMiny}) \right)
\]

where \( \text{index}(i, j) \) is the index value of the order of the ray within the ray set, \( \text{rad}(i, j) \)

\( \text{code}(i_x), \text{code}(j_y) \) represents the \( X \), \( Y \) axial component of the projection of the ray, and is the minimum value of the projection of the ray set on the axis \( X \), \( Y \), \( \text{code}(	ext{radsetMinx}) \) and \( \text{code}(	ext{radsetMiny}) \)

\( \text{radsetXlen} \) is the number of rays in the axial row of the ray set \( X \). By traversing all the triangles on the surface of the three-dimensional body, the intersection points of the rays in the ray set contained above are found and stored in the corresponding ray index locations. The set of intersection points corresponding to each ray is then sorted individually. Since the actual equation used in the calculation is a straight line, the true starting point of each ray can be seen as the axis \( Z \) at minus infinity, shooting towards the \( Z \) axis at positive infinity, then it is known from the knowledge of computational geometry that it intersects the three-dimensional surface an even number of times (special cases are not considered here), so the number of elements of each set of intersection points is even, and the sorted intersection points are paired two by two. Let \( \text{rad}(i, j) \) the pair of intersections in the corresponding \( X \), \( Y \) axis \( \text{code}(i, j) \), \( Z \) be of values \( z_{\text{min}}, z_{\text{max}} \), an \( z_{\text{min}} < z_{\text{max}} \), respectively, with

\[
z_{\text{min}} < (k+0.5) \ast \text{div} < z_{\text{max}}, \text{ and All the k values satisfying the above conditions } k \text{ are found and added to the corresponding } \ X \text{, } Y \text{ axial codes (i, j) determined by the rays ,all the cube lattices intersecting the interior of the 3D body code(i, j, k) can be calculated. Combined with the previous algorithm}


for the implementation of the three-dimensional body surface cube segmentation, the set of cubes intersecting the surface of the three-dimensional body is found, and all the cubes intersecting the three-dimensional body can be obtained by combining and removing the duplicate values.

This stage involves the computation with huge number of cubes related to the interior of the 3D body, so performance is extremely critical. The FPR algorithm uses a method of constructing rays that intersect the 3D body, which also uses a fast and efficient Hash retrieval technique. In the entire solution process, only one traversal of the set of triangular facets on the surface of the 3D body is required, whereas conventional algorithms require a large number of iterations of the set of triangular facets on the surface of the 3D body to determine the position of the cube in relation to the 3D body, with each calculation being extremely complex and involving magnitude of floating point operations. In theory, the FPR algorithm is more efficient.

4. Experiment
Experimental environment: graphics card: NVIDIA GTX1060, processor: Intel Core i5-8300H, operating system: Windows 10, memory: 8GB. 3D drawing engine: Open Scene Graph (OSG) 3.0.1 [10], development environment: Visual Studio 2010.

4.1 Display of the aggregation effect of the three-dimensional volume space segmentation cube grid collection
experimental content It mainly includes two parts: algorithm effect and performance analysis and 3D body space division effect.

![Figure 5. Segmentation level, time, and number of cubes for the cow model.](image)

4.1.1 Different levels of spatial segmentation effects for the same model. Firstly, the granularity level of the spatial segmentation has a very important relationship with the time spent on the spatial segmentation of the 3D body and the space occupied by the computer, as shown in Figure 5, which is derived from the model Figure 6, the number of triangles on the surface of the model is 5804, from the figure we can see that the algorithm time and the number of spatial segmentation cubes have an exponential growth trend with the increase in granularity level. The reason for this is that the number of cubes generated at this time is smaller than the number of triangles on the surface of the three-dimensional body, and the consumption of algorithm time is not obvious. This process can be regarded as an indispensable fixed time overhead in the operation of the algorithm. As can be seen in the figure,
this overhead tends to be stable at all levels of segmentation granularity and has little impact on the time performance of the high-level spatial segmentation. On the right-hand side of the line $L$, the number of cubes generated by the spatial segmentation of the 3D body exceeds the number of triangular slices on the surface of the 3D body, and the time taken to generate the cubes dominates the overall algorithm time, becoming a bottleneck in the performance of the algorithm.

In Figure 6, there is a significant difference in the segmentation of the 3D solid model using different segmentation granularity levels for the same model. Clearly, the higher the level of granularity, the better the spatial segmentation effect and the closer the resulting set of cubes to the original 3D solid model. This shows that when using spatial segmentation to do some spatial analysis of a 3D model as an alternative calculation, it is necessary to properly consider the actual engineering needs and choose the correct level of spatial segmentation, balancing the effect of spatial segmentation with the computational time and storage costs.

![Figure 6. Different levels of space segmentation effect display diagram of cow model.](image)

4.1.2 Different model segmentation effects at the same level. Figure 7 shows the spatial segmentation effects of three different 3D models, namely the Dumptruck model, the Glider model and the Cessna model, with the number of surface triangles of 23344, 640 and 7446 respectively, at a segmentation granularity level of 10. As can be seen from the figure, the spatially segmented set of cubes basically matches the original model in detail, and the spatially segmented set of cubes differs little from the original model in shape, indicating that the spatially segmented set is suitable for alternative calculations for the spatial analysis of 3D bodies. The figure has good adaptability to different models, what’s more, it demonstrates that the algorithm in this paper has good ability to spatially partition complex-shaped models, and the effect is relatively satisfactory.

![Figure 7. Display of the effect of different models at the same level](image)
4.2 Comparison with the octonion space partitioning algorithm

| Level | OS algorithm | FPR algorithm | Acceleration ratio |
|-------|--------------|---------------|--------------------|
|       | Time/s  | Count | Time/s  | Count |                        |
| 0     | 0.013   | 0     | 0.009   | 1     | 1.44                   |
| 1     | 0.026   | 0     | 0.008   | 4     | 3.25                   |
| 2     | 0.042   | 0     | 0.007   | 6     | 6.00                   |
| 3     | 0.131   | 30    | 0.008   | 30    | 16.38                  |
| 4     | 0.438   | 123   | 0.009   | 123   | 48.67                  |
| 5     | 2.678   | 639   | 0.009   | 639   | 297.56                 |
| 6     | 21.047  | 4199  | 0.012   | 4200  | 1753.92                |
| 7     | 163.110 | 29556 | 0.020   | 29557 | 8155.50                |

Firstly, a comparison of the spatial segmentation times at different levels of granularity for the same radar model using the new algorithm and the octonion segmentation algorithm is presented in Table 1. Column 1 of the table indicates the spatial segmentation level, columns 2 and 4 are the running times of the new algorithm and the octonion spatial segmentation algorithm at the corresponding levels, columns 3 and 5 are the number of spatial segmentation elements at their corresponding levels of granularity, and column 6 is the speed-up ratio achieved by the new algorithm to the octonion spatial segmentation algorithm. As can be seen from the table, with the increase of the segmentation granularity level, the time acceleration ratio of the algorithm in this paper shows a trend of gradually increasing, and at the segmentation granularity level of 8, its acceleration ratio can reach tens of thousands of times, which is far more time efficient than the existing octonion spatial segmentation algorithm.

![Figure 8. Comparison of the segmentation results of the old and new algorithms for the radar model](image-url)

Secondly, it can be seen from Table 1 that the number of sets generated by the FPR algorithm and the octonion spatial partitioning algorithm for spatial partitioning is very close in number when the model partitioning level reaches 8, 218889 and 218885 respectively, which indicates that the FPR algorithm is as effective as the traditional algorithm.

Finally, there is a small difference in the number of cubes at certain levels of granularity, the reason for this difference is shown in Figure 8, in the octonion spatial partitioning algorithm, the discarding of cubes generated by the spatial partitioning comes from the determination of whether the vertices of the cubes intersect the 3D body, as shown in the red grid on the right in Figure 8, although these red grids intersect the 3D body, their eight vertices are outside the 3D and do not intersect the 3D. At this point, the octonion segmentation algorithm will not be able to count these body elements into the set of 3D body space segmentation body elements. In this paper, the intersecting cubes on the surface...
and inside of the three-dimensional body are treated separately, so that the cubes can be counted together, which shows that the FPR algorithm can fill in the omissions of existing algorithms in the above situation, and is more complete than existing algorithms for three-dimensional body space segmentation.

5. Conclusion
In this paper, a novel fast projection ray algorithm (FPR) for spatial segmentation of 3D solids is proposed. Practical tests show that, in contrast to the traditional octree segmentation method, the segmentation calculation speed of the 3D solid model has been significantly improved, the segmentation effect is more satisfactory and can be basically close to the original solid shape, while the volume element approximation of the 3D solid can be used to assist in the analysis of the optimal deployment of radar stations, communication base stations, etc.

However, due to the complexity of 3D entities, the spatial segmentation algorithm FPR needs to be further extensively tested for more complex 3D entities. Meanwhile, parallel programming techniques such as CUDA can be further adopted in the future to continuously improve the performance of the algorithm and expand the application areas of the algorithm FPR based on the computing power of crowds of cores provided by GPU hardware\(^\text{[11,12]}\).

Reference.
[1] WANG H J. Research on network analysis method based on water injection model[J]. Journal of Wuhan University Information Science Edition, 2007, 32(5): 23-24.
[2] AIBERT Finding minimal cost paths in raster geographic information system map representations, genetic algorithms, simulated annealing and tabu search. Kent State University, USA, 2004.
[3] Hu P, Yang C Y. Basic theoretical problems of GIS-the spatial view of map algebra[J]. Journal of Wuhan University Information Science Edition, 2002, 27(6): 616-621.
[4] SN YDER J P. An equal-area map projection for polyhedral globes[J]. Geographic Information and Geovisualization, 1992, 29(1): 10-21.
[5] GOODCHILD M F, YANG S. A hierarchical data structure for global geographic information system [J]. Computer Graphics Vision and Image Processing, 1992, 54(1): 31-44.
[6] DUTTON G. Encoding and handling geospatial data with hierarchical triangular meshes [A]. Proceeding of 7th International Symposium on Spatial Data Handling, 1996, 34-43.
[7] WICKMAN F E, ELVERS E. A system of domains for global sampling problems [J]. Geografiska Analer, 1974, 56(3/4): 201-212.
[8] FEKETE G. Rendering and managing spherical data with sphere quadtrees [A]. Proceedings of the Visualization’90 IEEE Computer Society, 1990, 176-186.
[9] WHITE D, KIMMERLING J, OVERTON W S. Cartographic and geometric components of a global sampling design for environment monitoring [J]. Cartography & Geographical Information Systems, 1992, 19(1): 5-22.
[10] Akria K. Dissection of a sphere and Yin-Yang grids[J]. Earth Simulator, 2005, 3: 20-28.
[11] Kevin Sahr, Denis White, and A. Jon Kimerling. Geodesic Discrete Global Grid Systems [J]. Cartography and Geographic Information Science, 2003, 30(2): 121-134.
[12] Dong X.F., Zhang W. Pang, M.Y.. Analyzing efficiency of space-subdivision-based searching data structures[J]. Computer Engineering and Applications, 2016, 52(15): 73-78.
[13] Mark de Berg, et al. J. Deng. Computational geometry: Algorithms and Applications[M]. Beijing: Tsinghua University Press, 2009.
[14] Cao T, Beijing. Quick Encoding Compression Algorithm of Octree for Modeling Three Dimensional GIS[J]. Journal of image and graphics, 2002, 7(1): 50-54.
[15] Open Scene Graph[N/OL]. [2021-07-04]. Http://www.openscenegraph.org.
[16] Nickolls J R, Buck I, Garland M, et al. Scalable parallel programming with CUDA[J]. Queue, 2008.
[17] Li J., Zhang D.W., Jiang X.M.. Review on Parallelized Flood Inundation Models[J]. Computer Engineering and Applications. 2021,57 (13): 1-7

[18] Zhou S, Yan L I, Zhang Y, et al. Research of a Large-Scale 3D Model Visualization Method based on Octree and PagedLOD[J]. Journal of Dalian Jiaotong University, 2020.