






## Thickness and Clearance Analysis of 3D Object Using Maximum Inscribed Cubes

Masatomo Inui<sup>1</sup> , Ryo Ohno<sup>2</sup>  and Nobuyuki Umezu<sup>3</sup> 

<sup>1</sup>Ibaraki University, [masatomo.inui.az@vc.ibaraki.ac.jp](mailto:masatomo.inui.az@vc.ibaraki.ac.jp)

<sup>2</sup>Ibaraki University, [19t1016r@vc.ibaraki.ac.jp](mailto:19t1016r@vc.ibaraki.ac.jp)

<sup>3</sup>Ibaraki University, [nobuyuki.umezu.cs@vc.ibaraki.ac.jp](mailto:nobuyuki.umezu.cs@vc.ibaraki.ac.jp)

Corresponding author: Masatomo Inui, [masatomo.inui.az@vc.ibaraki.ac.jp](mailto:masatomo.inui.az@vc.ibaraki.ac.jp)

**Abstract.** Machine designers must have an accurate understanding of the thickness distributions of parts. In this study, a novel thickness evaluation method for polyhedral objects is proposed in which the thickness at a point on the object's surface is determined by the size of the maximum cube inscribed at that point. Considering a three-dimensional object with the inside and outside inverted, the thickness definition can also be used to evaluate the amount of clearance on the part surface or between parts. The interior of a polyhedral object is converted to an equivalent voxel model, and a "cuboid distance field" consisting of inscribed cubes centered at each voxel is defined. The maximum inscribed cube tangent to any point on the surface of the object was obtained by analyzing the cuboid distance field along a half-line extending from the point to the interior of the object. We devised a fast algorithm to compute the cuboid distance field using a hierarchical bounding box structure of surface polygons in combination with the parallel processing function of the graphics processing unit. The effectiveness of the algorithm was verified through computational experiments.

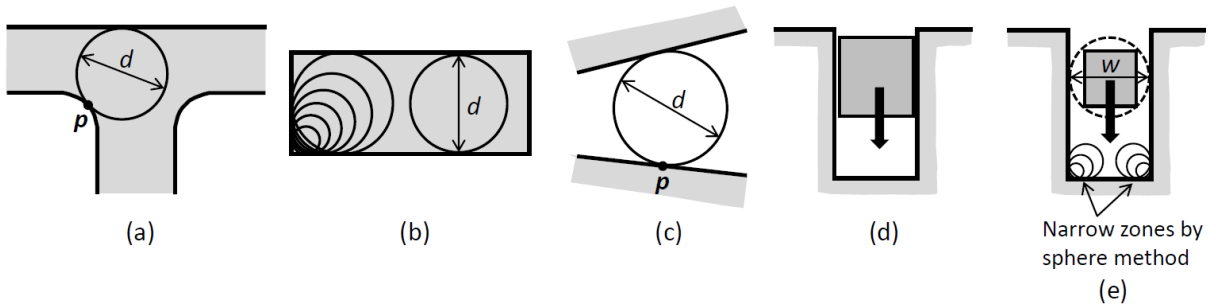
**Keywords:** Distance Field, Thickness, Clearance, Graphics Processing Unit, Geometric Modeling

**DOI:** <https://doi.org/10.14733/cadaps.2024.646-658>

### 1 INTRODUCTION

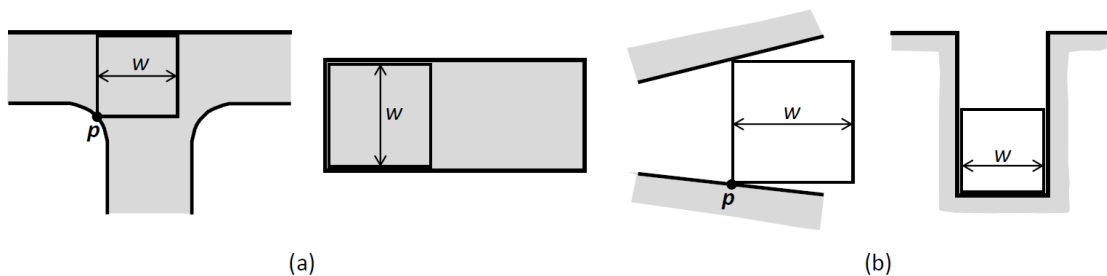
The part thickness is an important design parameter that affects the mass properties and robustness of mechanical products. Machine designers must work with an accurate understanding of the thickness distributions of parts. In international standards, part thickness is defined using conventional dimensions [1] unsuitable for use in three-dimensional (3D) CAD models with complex shapes. For this reason, many CAD systems use thickness evaluation for 3D objects based on nonstandard definitions, such as the ray and sphere methods [15]. Considering a 3D shape with the inside and outside inverted, the definition of thickness can also be used to evaluate the amount of clearance on the part surface or between parts.

The sphere method is used in many CAD systems to analyze the thickness of a 3D object. This method defines the thickness at point  $p$  on the surface of a 3D object based on the diameter  $d$  of the maximum sphere inscribed on the object at  $p$  (Fig. 1(a)). For the plate-like parts, the thickness specified by the sphere method was generally consistent with the thickness defined by the dimensions in the standard. Because the maximum inscribed sphere becomes smaller near a convex shape, the sphere method displays all corners of the part as thin (Fig. 1(b)). Thus, it is difficult to understand the thickness distribution of a part correctly using the sphere method. To avoid this problem, the corners of parts are often ignored when visualizing the thickness distribution using the sphere method.



**Figure 1:** Thickness and clearance analysis using the sphere method.

This problem has even more troublesome consequences when the sphere method is used for the clearance analysis. In the sphere method, the clearance at point  $p$  on the object surface is expressed as the diameter of the maximum circumscribed sphere tangential to  $p$  (Fig. 1(c)). Consider the placement of a box-shaped part into a slot as shown in Fig. 1(d). Although the part can be installed dimensionally, the following two problems arise when the amount of the clearance is evaluated by the sphere method (Fig. 1(e)). The first problem is that even if the clearance is evaluated as  $w$  by the sphere method, it does not mean that a box-shaped part with a width of  $w$  can be placed. Since the sphere method guarantees the placement of a sphere of diameter  $w$ , placement can only be guaranteed for smaller parts that fit into this sphere. Another problem is that the sphere method evaluates the clearance as narrow at the corners, and the part is judged to be difficult to install. The shapes of several mechanical parts were often approximated using rectangular boxes. The sphere method has difficulty in evaluating clearances that consider the characteristics of the placement of such parts.



**Figure 2:** Thickness analysis using the maximum inscribed cube (a) and clearance analysis using the maximum circumscribed cube (b).

In this paper, we propose a novel method for evaluating the thickness and clearance of a 3D object that is less likely to cause such problems. In this method, the thickness at point  $p$  on the surface of an object is defined as the width  $w$  of the maximum cube inscribed at  $p$  (Fig. 2(a)). In the clearance analysis, the width of the maximum circumscribed cube is used instead (Fig. 2(b)). The cubes are

assumed to be aligned with respect to the X-, Y-, and Z-axes of the coordinate frame of the part. By using this method, it is possible to solve the problem in which the corners of a plate-like part are judged to be thin, or difficulty in correctly evaluating the possibility of placing box-shaped parts. By properly stretching the part shape vertically or horizontally before performing the analysis, it is possible to evaluate the thickness/clearance using a more general rectangular object instead of a cube.

In the next section, we present prior studies on thickness and clearance analysis. Section 3 outlines the computation method for the proposed thickness and clearance analysis using the maximum inscribed/circumscribed cubes. Section 4 describes the details of our computational method, focusing on parallel processing using a graphics processing unit (GPU). Section 5 presents the results of calculations using our prototype software. Finally, Section 6 presents our conclusions.

## 2 PRIOR STUDIES

In addition to the sphere method, the ray method [15] is a commonly used thickness evaluation method in commercial CAD systems. In this method, the thickness at an arbitrary point  $\mathbf{p}$  on the object's surface is defined using a ray that starts from  $\mathbf{p}$  and extends to the inward normal direction at  $\mathbf{p}$ . The intersection point  $\mathbf{q}$  between the ray and surface immediately opposite  $\mathbf{p}$  is calculated, and the Euclidean distance between  $\mathbf{p}$  and  $\mathbf{q}$  is used as the thickness value at  $\mathbf{p}$ . However, the ray method has a problem in that the thickness of  $\mathbf{p}$  differs from that of  $\mathbf{q}$  when the surfaces containing points  $\mathbf{p}$  and  $\mathbf{q}$  are not parallel.

GeomCaliper is software specialized for visualizing the thickness of 3D objects using ray and sphere methods. GeomCaliper employs a uniform spatial grid and a k-d tree to achieve an efficient thickness calculation performance [15]. The authors developed software to analyze the thickness of arbitrary points on a polyhedral object using the sphere method [7]. The software achieves a high calculation speed using the parallel processing capability of the GPU. This software has also been used to realize a technique for simulate the sink marks that appear on the surfaces of plastic parts during injection molding [9].

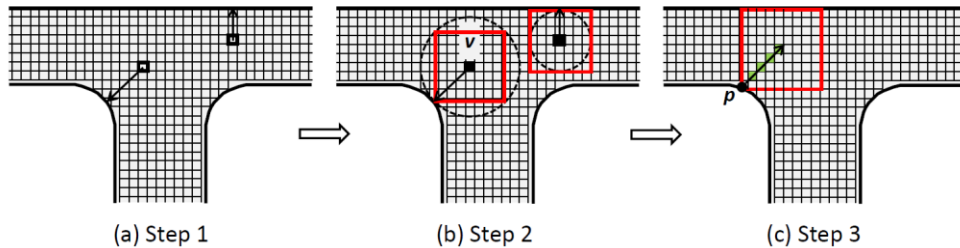
Patil and Ravi proposed three thickness analysis methods using voxel models [14]. They are called (i) "radiographic thickness," which is based on a transformation of the ray method, (ii) "exterior thickness," which is a modification of the sphere method, and (iii) "interior thickness," which is a type of distance transformation. The first two methods can be regarded as modifications of the ray and sphere methods. Therefore, the problems associated with the ray and sphere methods have not been solved. Lu *et al.* proposed a thickness analysis method based on distance transformations to detect thick regions of a 3D object [11]. They used this method to automatically detect shapes prone to blowholes during casting.

The distance field is a method used to analyze the thickness distribution inside a 3D object [2, 3, 4, 5, 6, 8, 10, 12, 16, 17]. The interior of a 3D object is decomposed into a uniform grid of cells (voxels), where the distance from the center of the voxel to the surface of the object is calculated and recorded in each voxel. Using the distance field, the thickness distribution of a 3D object can be efficiently visualized using the sphere method [8]. In this study, the distance field is also used in some processes to speed up the computation. When using high-resolution voxel models, the computation of the distance field is a time-consuming process. In [6, 8, 16], methods to manage an object's surface polygons using a hierarchical tree structure of bounding boxes were proposed to reduce the computation cost. Some of them realized a further reduction in cost using the parallel processing capability of the GPU [8, 16].

## 3 ALGORITHM OUTLINE

In this section, we present an outline of our thickness analysis method using inscribed cubes. Using the inverted shape of the objects, the same method can be used to evaluate their clearance. The

maximum cube inscribed at an arbitrary point on the surface of a 3D object was obtained through the following three steps:



**Figure 3:** Outline of our algorithm. (a) calculation of the distance field, (b) calculation of the cuboid distance field, and (c) evaluation of the thickness using the cuboid distance field.

**Input:** Polyhedral model of a machine part.

**Step 1:** The input polyhedral model is converted into an equivalent voxel model (Fig. 3(a)), and the distance from the center point of each voxel to the nearest point on the surface of the polyhedron is calculated. The obtained distance values were recorded for each voxel and a distance field was constructed inside the polyhedral model.

**Step 2:** The distance value recorded in voxel  $\mathbf{v}$  represents the radius of the sphere inscribed in the polyhedron and centered at  $\mathbf{v}$ . Using the information of each inscribed sphere, a cube centered at  $\mathbf{v}$ , aligned with respect to the X-, Y-, and Z-axes, and inscribed in the polyhedron is calculated (Fig. 3(b)). The half value of the width of the cube obtained is recorded as a new distance value for  $\mathbf{v}$ . New distance field obtained by using this method is called "cuboid distance field" in this study.

**Step 3:** The maximum inscribed cube in contact with point  $\mathbf{p}$  on the surface of the model is determined by extending a ray from  $\mathbf{p}$  to the interior of the cuboid distance field and examining the distance values recorded in the voxels on the line (green voxels in Fig. 3(c)).

**Output:** Maximum cube inscribed at an arbitrary point on the surface, and aligned with respect to the coordinate axes.

The following two technologies were newly developed in this study.

- In Step 2, we computed the inscribed cube centered on each voxel to obtain the cuboid distance field. An efficient calculation of the cuboid distance field was realized using GPU parallel processing technology, bounding volume hierarchy, and GPU shared memory. The conventional distance field stores the radius of the inscribed sphere centered at each point in a solid. Our cuboid distance field, on the other hand, records half the width of the inscribed cube centered at a point inside the object. This is the first study to define such a distance field and develop an efficient method for its calculation.
- In Step 3, we developed a technique for computing the maximum inscribed cube tangent to an arbitrary point on the surface of a polyhedron using the cuboid distance field. Our method assumes that the orientation of the surface polygon is not parallel to the coordinate axes.

## 4 DETAILS OF ALGORITHM

This chapter describes the details of the calculations in each step.

### 4.1 Step 1 Construction of Distance Field

As mentioned in the previous section, many studies have been conducted on the voxelization of 3D models and the construction of the distance field in the voxel model.

In this study, we employed the original distance field construction algorithm [8].

The basic procedure for distance field construction is an iterative calculation of the distance between the center point  $\mathbf{p}$  of each voxel and the surface polygon of the 3D object. Our algorithm [8] suppresses unnecessary distance calculations using hierarchical bounding boxes to make this process efficient. An axis-aligned bounding box (AABB) that holds all surface polygons of the object within is defined. The polygons were classified into two groups consisting of the same number of polygons, and two smaller AABBs enclosing each group of polygons were defined. These processes are repeated to organize all surface polygons of a given object into a binary tree structure of AABBs.

In the distance calculation, the AABB tree was traversed with point  $\mathbf{p}$  from the root node to the leaf nodes in a breadth-first manner. Consider  $2^n$  AABBs ( $BB_0, BB_1, \dots, BB_{2^n-1}$ ) in the  $n$ -th layer of the tree. For each AABB  $BB_i$ , compute the distance  $d_i$  representing the shortest distance between  $\mathbf{p}$  and a point on  $BB_i$ . Another distance  $D_i$  is computed which represents the maximum distance between  $\mathbf{p}$  and a point on  $BB_i$ . Select box  $BB_{min}$  whose value  $D_{min}$  is the smallest at the maximum distance  $D_i$ . If  $d_j$  is larger than  $D_{min}$  then the distance between  $\mathbf{p}$  and any polygon within  $BB_j$  must be larger than the distance between  $\mathbf{p}$  and a polygon within  $BB_{min}$  therefore, all polygons within  $BB_j$  can be ignored in the distance calculations. Child nodes of  $BB_j$  can be excluded from the following process in the breadth-first traversal.

After AABB tree traversal, polygons in the AABB of the reached leaf nodes were collected. The distance between the collected polygons and  $\mathbf{p}$  is calculated in parallel using a GPU to determine the distance between  $\mathbf{p}$  and the surface of the object.

## 4.2 Step 2 Construction of Cuboid Distance Field

The distance field obtained in Step 1 contains, for each voxel  $\mathbf{v}$ , the distance from the voxel center to the nearest point on the polyhedron surface. This value corresponds to the radius of the sphere  $\mathbf{S}$  centered at  $\mathbf{v}$  and is inscribed in the polyhedron. Consider a cube  $\mathbf{c}$  inscribed in  $\mathbf{S}$  and another cube  $\mathbf{C}$  circumscribed in  $\mathbf{S}$ . The orientations of  $\mathbf{c}$  and  $\mathbf{C}$  are aligned to the X-, Y-, and Z-axes. The size of the axis-aligned cube  $\mathbf{M}$  centered at  $\mathbf{v}$  and inscribed in the polyhedron must be equal to or larger than  $\mathbf{c}$  and equal to or smaller than  $\mathbf{C}$  (Fig. 4). Therefore, using  $\mathbf{c}$  and  $\mathbf{C}$  as the initial values, the size of  $\mathbf{M}$  was determined using the bisection method. Specifically, the following process is repeated for all voxels:

**Initial values:** Axis-aligned cube  $\mathbf{c}$  inscribed in  $\mathbf{S}$  and axis-aligned cube  $\mathbf{C}$  circumscribed in  $\mathbf{S}$ .

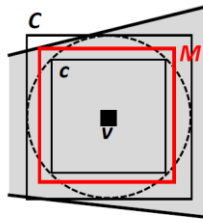
**Step 2.1:** Obtain an axis-aligned cube  $\mathbf{M}$  of midsize  $\mathbf{c}$  and  $\mathbf{C}$  centered at  $\mathbf{v}$ .

**Step 2.2:** If the difference between the widths of  $\mathbf{c}$  and  $\mathbf{C}$  is less than the given tolerance value,  $\mathbf{M}$  is output as a cube whose center point is at  $\mathbf{v}$  and inscribed in  $\mathbf{S}$ . In our current implementation, 0.01 mm is set to the tolerance value.

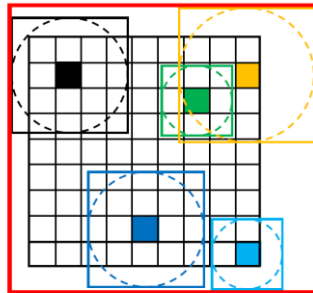
**Step 2.3:** If  $\mathbf{M}$  intersects the polyhedron's surface,  $\mathbf{M}$  is set as a new  $\mathbf{C}$ . If  $\mathbf{M}$  does not intersect,  $\mathbf{M}$  is set to a new  $\mathbf{c}$ . Return to Step 2.1.

The half value of the width of the obtained inscribed cube was stored as the new distance value for voxel  $\mathbf{v}$ . After computing all voxels, a cuboid distance field was constructed.

The most time-consuming process in the above algorithm is the intersection detection between a cube and polyhedron in Step 2.3. The hierarchical AABB tree constructed in Step 1 can be used to accelerate the calculation. To determine the intersection of a cube and a polyhedron, the AABB tree was traversed in a depth-first manner, and at each node, the intersection of the corresponding AABB and cube was checked. When an AABB that does not intersect the cube is found, the traversal of AABB tree after the corresponding node is canceled, to narrow down the polygons that may intersect the cube efficiently. When the leaf node of the AABB tree was reached, an intersection check was performed between the cube and polygons in the AABB corresponding to the leaf node.



**Figure 4:** Relationships between  $C$ ,  $c$  and  $M$  for voxel  $v$ .

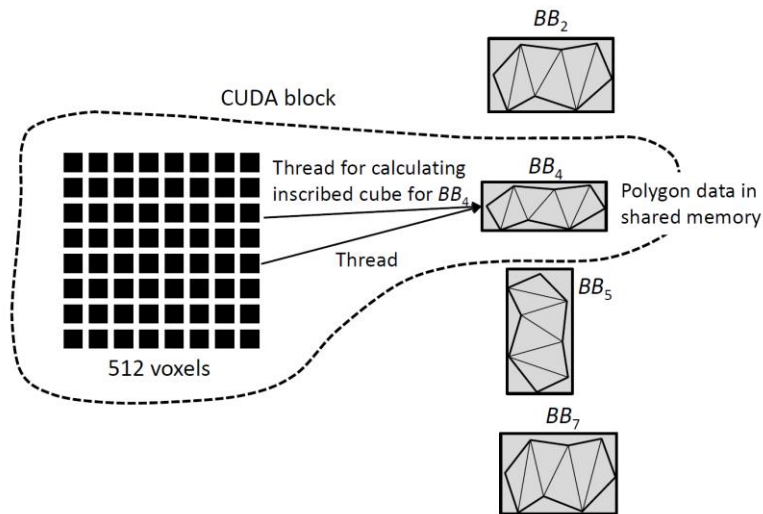


**Figure 5:** A large rectangular box (red) that encompasses small cubes associated with the voxels in the sublattice.

The construction of the cuboid distance field can be accelerated using the parallel-processing capabilities of a GPU. The lattice structure of the voxel model was divided into  $8 \times 8 \times 8$  cuboidal sublattices with 512 voxels each. The calculation of a cube inscribed into the polyhedron with a voxel as its center was performed for these 512 voxels in parallel. For each voxel in the sublattice, an inscribed sphere centered at the voxel and an axis-aligned cube  $C$  circumscribing the sphere were defined. In Fig. 5, a sphere and cube circumscribing the sphere with the same color are illustrated for each voxel with a unique color. When calculating an inscribed cube centered at a voxel using the bisection method, polygons that intersect the cube always intersect circumscribing cube  $C$ . Therefore, cube  $C$  can be used to narrow down the candidate polygons that may intersect.

Consider the calculation of the inscribed cubes for two neighboring voxels  $v_0$  and  $v_1$  in the sublattice. Because they are very close, the set of polygons referred to in the intersection calculation to obtain an inscribed cube for  $v_0$  and another set of polygons referred to in the calculation for  $v_1$  are expected to be almost the same. Therefore, instead of selecting candidate polygons for each of the 512 voxels in the sublattice, we selected candidate polygons for all 512 voxels. A larger rectangular box encompassing 512 cubes associated with the voxels was defined (large red box in Fig. 5). Using this box and the AABB tree, polygons can be involved in intersection detection in the inscribed cube calculation for 512 voxels. The vertex data of the selected polygons are transferred to a GPU. A GPU thread was invoked for each of the 512 voxels, and the cube centered on each voxel and inscribed to the polyhedron was computed in parallel for the 512 voxels using the transferred data.

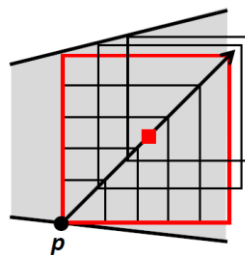
Fig. 6 illustrates the calculation method for the inscribed cube centered on each voxel using polygons stored in the leaf AABBs obtained as a result of the AABB tree traversal. AABBs  $BB_2$ ,  $BB_4$ ,  $BB_5$  and  $BB_7$  in the figure represent the leaf boxes resulting from the AABB tree traversal.



**Figure 6:** Thread and block definition for parallel intersection calculation.

The polygons in these AABBs are referenced in subsequent intersection calculations to obtain the inscribed cube. In the GPU processing, a set of threads called blocks is defined [13]. For each box  $BB_i$  with candidate polygons, we define a block consisting of 512 threads. A thread in each block computes, for the corresponding voxel in a sublattice, the smallest cube centered on that voxel, and inscribes it to the polygon in the box. After processing each of the 512 voxels, a candidate inscribed cube was obtained for each box. In the example shown in Fig. 6, four candidates for the inscribed cube are obtained for each of the 512 voxels, corresponding to  $BB_2$ ,  $BB_4$ ,  $BB_5$  and  $BB_7$ . The smallest cube was selected as the inscribed cube centered on that voxel.

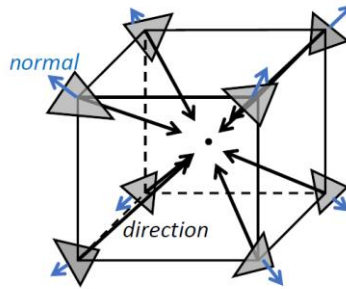
The GPU has fast shared memory that can be used by threads within the same block. In our software, the geometric information of the polygons in the leaf AABB corresponding to the block is stored in shared memory and used in the parallel inscribed cube calculation of the 512 voxels. By storing shareable data in a high-speed memory and using it with multiple threads, the time required for storing and accessing data can be reduced, enabling further acceleration of the process.



**Figure 7:** Determination of the maximum inscribed cube by tracing a half line extended from a surface point  $p$ .

### 4.3 Step 3 Maximum Inscribed Cube for Surface Point

The determination of the maximum cube inscribed to a surface point of a 3D object involves tracing and checking the distance values in the cuboid distance field along a straight line starting from the point. Fig. 7 illustrates the tracing process. As shown in the figure, the center points of the cubes inscribed to the surface point  $p$  are aligned on a straight line extending 45° diagonally from  $p$ .



**Figure 8:** Classification of the tracing direction according to the surface normal.

Therefore, we traced along this line in the cuboid distance field and examined the voxels on the line and their distance values. As the first peak of the distance value corresponds to half the width of the maximum inscribed cube, we can use twice the thickness value at  $\mathbf{p}$ . The search direction can be classified into eight directions according to the normal direction of the polygon in which  $\mathbf{p}$  exists (Fig. 8).

```

if (normal.x > 0.0 && normal.y > 0.0 && normal.z > 0.0) then direction = (-1.0, -1.0, -1.0)
if (normal.x > 0.0 && normal.y > 0.0 && normal.z < 0.0) then direction = (-1.0, -1.0, 1.0)
if (normal.x > 0.0 && normal.y < 0.0 && normal.z > 0.0) then direction = (-1.0, 1.0, -1.0)
if (normal.x > 0.0 && normal.y < 0.0 && normal.z < 0.0) then direction = (-1.0, 1.0, 1.0)
If (normal.x < 0.0 && normal.y > 0.0 && normal.z > 0.0),
    then the direction = (1.0, -1.0, -1.0).
if (normal.x < 0.0 && normal.y > 0.0 && normal.z < 0.0) then direction = (1.0, -1.0, 1.0)
& (normal.y < 0.0 && normal.z < 0.0 && normal.z > 0.0);
    then, the direction = (1.0, 1.0, -1.0)
If (normal.x < 0.0 && normal.y < 0.0 && normal.z < 0.0), the direction is = (1.0, 1.0, 1.0)

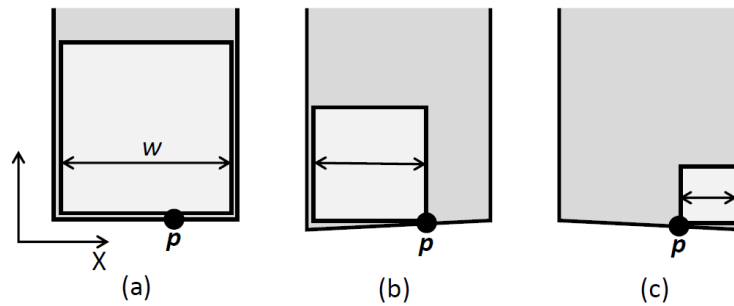
```

For machine parts, the normal direction of surface polygons is often parallel to the X-, Y-, or Z-axes. In such cases, some of the X, Y, or Z components in the normal direction were zero, and the above classification did not apply. One possible solution to this problem is to rotate the polygon slightly such that all components in the normal direction of the polygon are nonzero; however, this method does not solve this problem. Fig. 9 illustrates the problem with this method in two dimensions. If the polygon (edge) is parallel to the x-axis, the size of the largest cube inscribed at point  $\mathbf{p}$  is  $w$ . If the orientation of this polygon is slightly tilted, the maximum inscribed cube changes depending on the tilting direction, as shown in Fig. 9(b) and (c). In both cases, the size of the cube becomes smaller than  $w$ . We are developing a new method to solve this problem but have yet to design an efficient algorithm.

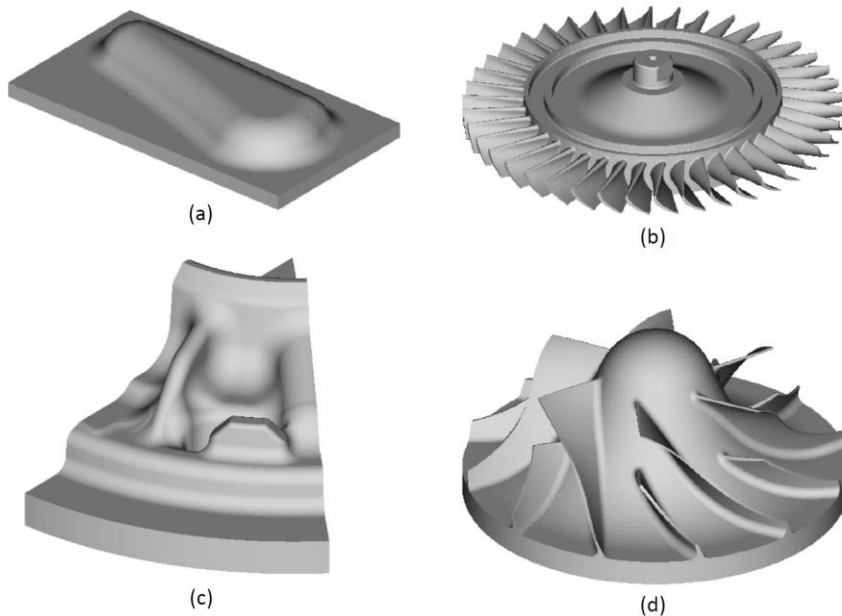
## 5 COMPUTATIONAL EXPERIMENTS

The software was used to calculate the cuboid distance field by using the developed algorithm. Computational experiments were conducted to analyze the thickness of the part using the maximum inscribed cubes. Visual Studio 2017 and CUDA 10.1 [13] were used for implementation. A notebook PC with a Core i7 CPU, GeForce RTX 3080Ti GPU, and 32GB memory was used for the calculations.





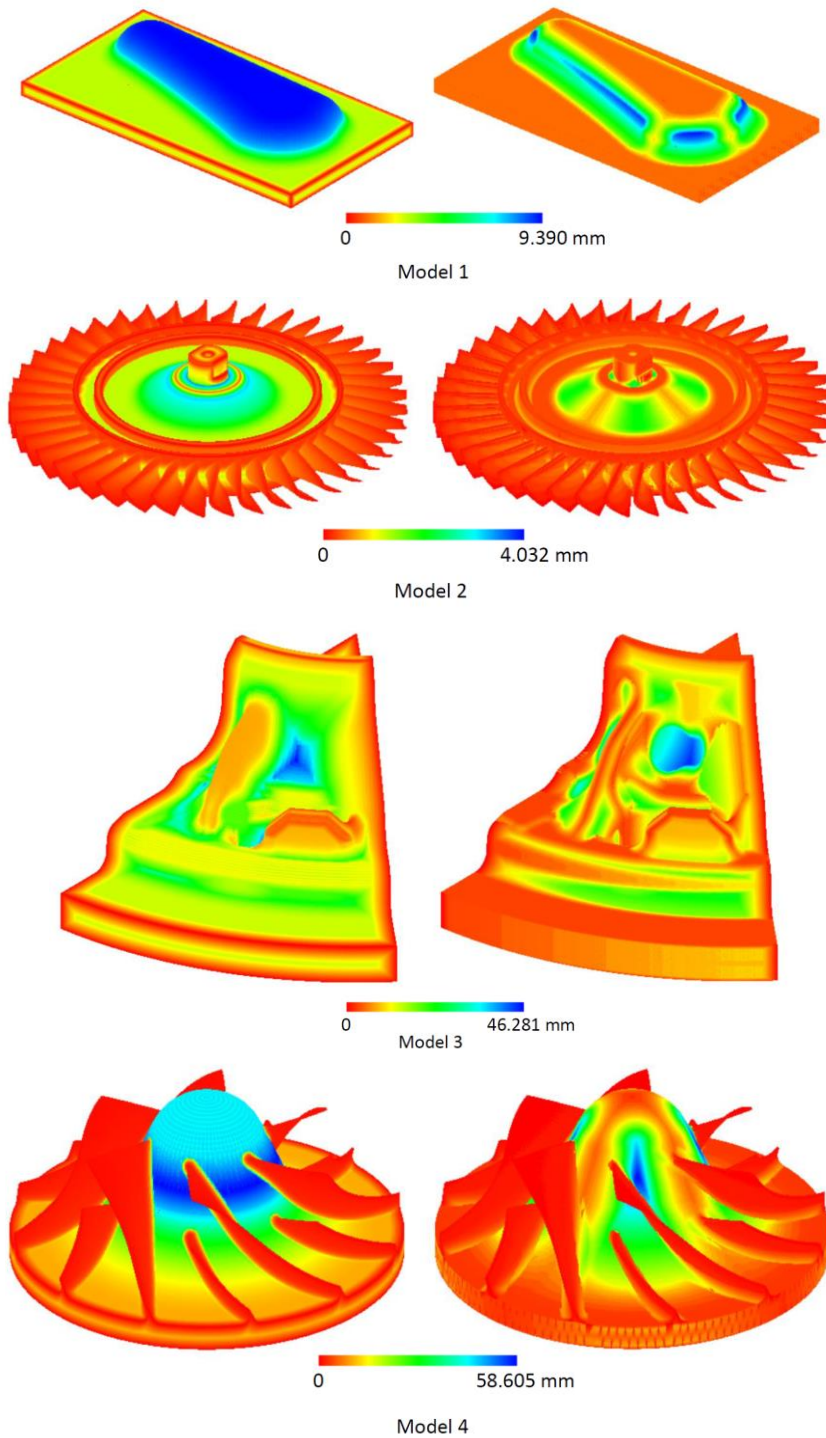
**Figure 9:** Limitation of the proposed method.



**Figure 10:** Sample models used in our experiments. (a) Model 1, (b) Model 2, (c) Model 3, and (d) Model 4.

Fig. 10 shows the sample polyhedral models used in the experiments. Fig.11 shows the thickness analysis results of the models. In this analysis, many points are densely scattered on the part's surface. Subsequently, the thickness was analyzed for each point using the sphere method for (a) and our current method with the maximum inscribed cubes for (b). In both cases, the maximum thickness is colored blue, and the zero thickness is colored red. As our method does not correctly calculate the maximum inscribed cube for surfaces parallel to the X-, Y-, or Z-axes, the coloring results for such surfaces may not be correct. In these models, the total computation time for thickness analysis with our current method was approximately five times longer than that of the sphere method.

By comparing the analysis results in Figs. 11(a) and (b), several areas with completely different thickness distributions can be observed. The analysis results using the sphere method showed that all the corners of the part were thin, whereas the plate-like shape was uniform in thickness using the current method. In our method, the upper part of the curved shape in Model 1 was determined to be thinner.



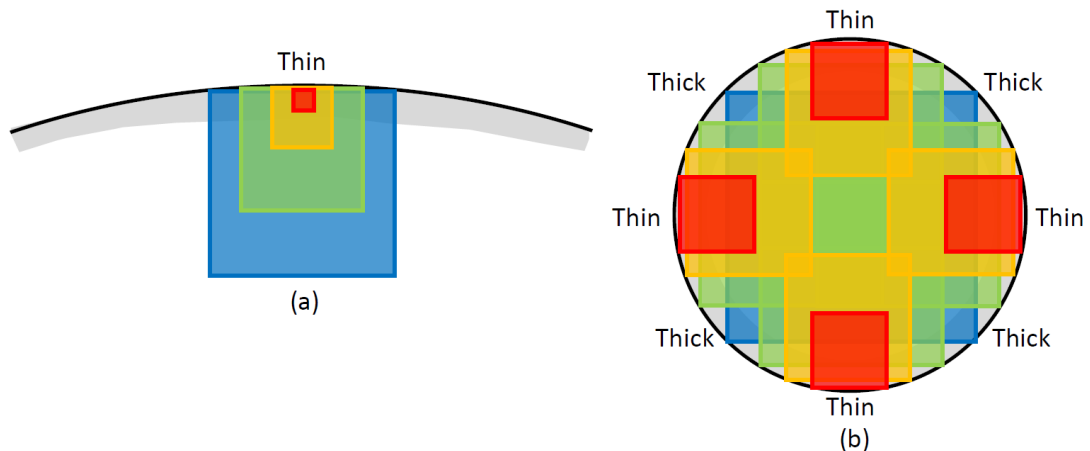
**Figure 11:** Thickness visualization results. (a) Sphere method. (b) Our method using maximum inscribed cubes.

Periodic changes in thickness can be observed in the lateral part of Model1, 2, and 4. These differences were due to the changes in the size of the cubes that could be placed inside the object. As shown in Fig. 12(a), the cubes that can be placed in the upper part of the curved shape become smaller; as a result, they are considered thinner. Inside the lateral part, as shown in Fig. 12(b), the size of the cubes that can be placed changes with a 90-degree cycle, resulting in the repeated appearance of thick and thin areas.

Table 1 lists the necessary computation time to analyze the thickness. In the table, from the left column to the right, the number of polygons of the given models, the number of voxels of the distance field, and the necessary time for constructing the distance field are listed (Step 1). The necessary time for constructing the cuboid distance field (Step 2), the number of points scattered on the model surface, and the time necessary for calculating the size of the maximum cubes inscribed to the points (Step 3) are also given in the table. It can be observed that the processing time required in Step 2 was the longest, and the processing time for Step 3 was almost negligible. In these models, the calculations were completed in about four minutes, and our software is considered sufficiently practical.

	Num. polys	Num. voxels	Time for Step 1 (s)	Time for Step 2 (s)	Num points	Time for Step 3 (s)
Model 1	17,842	102,290,282	19.525	73.521	48,218	0.641
Model 2	197,450	102,491,663	21.586	65.311	75,354	0.566
Model 3	15,910	100,871,641	33.771	184.922	121,498	0.901
Model 4	41,724	101,327,283	38.179	183.738	91,469	0.963

**Table 1:** Necessary computation time.



**Figure 12:** Variation in the size of cubes that can be placed inside an object. (a) Maximum inscribed cubes placed in the upper part of the curved shape. (b) Maximum inscribed cubes placed inside the lateral part of the object.

## 6 CONCLUSIONS

In this study, a novel thickness evaluation method for a polyhedral solid model was developed, in which the thickness of a point on the surface of the model was determined by the size of the maximum cube inscribed to the point. This method can solve the problem of the conventional sphere method, which evaluates all the corners of a solid as thin, and is considered effective in the evaluation of clearance on a 3D object. The interior of a polyhedral object is converted into an equivalent voxel model, and a cuboid distance field consisting of inscribed cubes centered at each voxel is defined.

We devised an algorithm to compute the cuboid distance field quickly using a hierarchical AABB tree of surface polygons of the object in combination with GPU parallel processing technology. The maximum inscribed cube tangent to any point on the object's surface was obtained by analyzing the cuboid distance field along a half-line extending from the point to the object's interior. The effectiveness of the algorithm was verified through computational experiments. Our algorithm for calculating the maximum inscribed cubes is limited in that it cannot correctly calculate the thicknesses of the surfaces parallel to the X-, Y-, and Z-axes. We intend to solve this problem and study applications of the developed thickness evaluation method in the future.

Masatomo Inui, <https://orcid.org/0000-0002-1496-7680>

Ryo Ohno, <https://orcid.org/0009-0007-1473-4602>

Nobuyuki Umezu, <https://orcid.org/0000-0002-7873-7833>

## REFERENCES

- [1] Dimensioning and Tolerancing Y14.5–2018, ASME, 2019.
- [2] Bastos, T.; Celes, W.: GPU-Accelerated Adaptively Sampled Distance Fields, Proc. of IEEE International Conference on Shape Modeling and Applications, 2008, 171–178. <https://doi.org/10.1109/SMI.2008.4547967>
- [3] Chang, B.; Cha, D.; Ihm, I.: Computing local signed distance fields for large polygonal models, Computer Graphics Forum, 27(3), 2008, 799–806. <https://doi.org/10.1111/j.1467-8659.2008.01210.x>
- [4] Danielsson, P.-E: Euclidean distance mapping, Computer Graphics and Image Processing, 14, 1980, 227–248. [https://doi.org/10.1016/0146-664X\(80\)90054-4](https://doi.org/10.1016/0146-664X(80)90054-4)
- [5] Frisken, S.F.; Perry, R.N.; Rockwood, A.P.; Jones, T.R.: Adaptively sampled distance fields: a general representation of shape for computer graphics, Proc. 27th annual conference on Computer graphics and interactive techniques, Siggraph '00, 2000, 249–254. <https://doi.org/10.1145/344779.344899>
- [6] Gu´eziec, A.: Meshsweeper: dynamic point-to-polygonal mesh distance and applications, IEEE Trans. Vis. Comput. Graph., 7(1), 2001, 47–61. <https://doi.org/10.1109/2945.910820>
- [7] Inui, M.; Umezu, N.; Shimane, R.: Shrinking sphere: A parallel algorithm for computing the thickness of 3D objects, Computer-Aided Design and Applications, 13(2), 2015, 199–207. <https://doi.org/10.1080/16864360.2015.1084186>
- [8] Inui, M.; Umezu, N.; Wakasaki, K.; Sato, S.: Thickness and clearance visualization based on distance field of 3D objects, Journal of Computational Design and Engineering, 2(3), 2015, 183–194. <https://doi.org/10.1016/j.jcde.2015.04.001>
- [9] Inui, M.; Onishi, S.; Umezu, N.: Visualization of potential sink marks using thickness analysis of finely tessellated solid model, Journal of Computational Design and Engineering, 5(4), 2018, 409–418. <https://doi.org/10.1016/j.jcde.2018.02.003>
- [10] Jones, M.W.; Bærentzen, J.A.; Sramek, M.: 3D distance fields: a survey of techniques and applications, IEEE Transactions on Visualization and Computer Graphics, 12(4), 2006, 881–899. <https://doi.org/10.1109/TVCG.2006.56>
- [11] Lu, S.C.; Rebello, A.B.; Miller, R.A.; Kinzel, G.L.; Yagel, R.: A simple visualization tool to support concurrent engineering design, Computer-Aided Design, 29(10), 1997, 727–735. [https://doi.org/10.1016/S0010-4485\(97\)00015-8](https://doi.org/10.1016/S0010-4485(97)00015-8)
- [12] Mullikin, J.C.: The vector distance transform in two and three dimensions, CVGIP: Graphical Models and Image Processing, 54(6), 1992, 526–535. [https://doi.org/10.1016/1049-9652\(92\)90072-6](https://doi.org/10.1016/1049-9652(92)90072-6)
- [13] CUDA C Programming Guide, NVIDIA, 2018.
- [14] Patil, S.; Ravi, B.: Voxel-based representation, display and thickness analysis of intricate shapes, Proc. Ninth International Conference on Computer Aided Design and Computer Graphics (CAD-CG'05), 2005. <https://doi.org/10.1109/CAD-CG.2005.86>

- [15] Sinha, B.: Efficient wall thickness analysis methods for optimal design of casting parts, Presented at Engineering Design, 2007. Available: [https://geomcaliper.geometricglobal.com/wp-content/blogs.dir/13/files/2009/09/EfficientWallThicknessAnalysis\\_GeomCaliper.pdf](https://geomcaliper.geometricglobal.com/wp-content/blogs.dir/13/files/2009/09/EfficientWallThicknessAnalysis_GeomCaliper.pdf)
- [16] Sud, A.; Manocha, D.: Fast distance field computation using graphics hardware, UNC Computer Science Technical Report TR03-206, 2003. <http://gamma.cs.unc.edu/DIFI/Sud-DiFi-TR026.pdf>
- [17] Sullivan, A.; Erdim, H.; Perry, R.N.; Frisken, S.F.: High accuracy NC milling simulation using composite adaptively sampled distance fields, Computer-Aided Design, 44(6), 2012, 522–536. <https://doi.org/10.1016/j.cad.2012.02.002>