Title:
**Thickness and Clearance Analysis of 3D Object Using Maximum Inscribed Cubes**

Authors:
Masatomo Inui, masatomo.inui.az@vc.ibaraki.ac.jp, Ibaraki University
Ryo Ohno, 19t1016r@vc.ibaraki.ac.jp, Ibaraki University
Nobuyuki Umezu, nobuyuki.umezu.cs@vc.ibaraki.ac.jp, Ibaraki University

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

DOI: 10.14733/cadconfP.2023.138-142

Introduction:
The part thickness is an important design parameter that affects the mass properties and robustness of mechanical products. Mechanical designers must work with an accurate understanding of the thickness distribution of parts. In international standards, part thickness is defined using conventional dimensions, 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 ray and sphere methods [1]. 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.

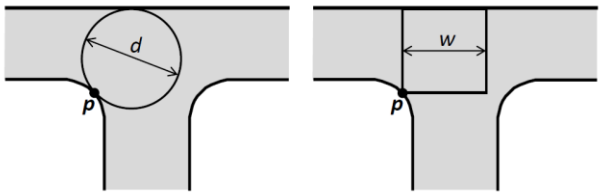


Fig. 1: Sphere method (a) and our current method (b).

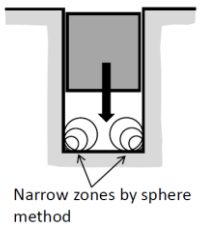


Fig. 2: Problem of sphere method for clearance evaluation.

The sphere method is used in many CAD systems to analyze the thickness of a 3D object. This method defines the thickness of point $\boldsymbol{p}$ on the surface of a 3D object based on the diameter of the maximum sphere inscribed on the object at $\boldsymbol{p}$ (Fig. 1(a)). For the plate-like parts, the thickness specified by the sphere method is generally consistent with the thickness defined by the dimensions in the standard. However, when this method is used to evaluate the clearance, it yields results that are difficult to understand. In the sphere method, the clearance at point $\boldsymbol{p}$ on the object surface is expressed as the diameter of the maximum circumscribed sphere tangential to $\boldsymbol{p}$. Consider the placement of a box-shaped part in space as shown in Fig. 2. Although the part can be installed dimensionally, the sphere method evaluates the clearance as narrow at the corners, and the part is judged to be difficult to install.

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 a point $\boldsymbol{p}$ on the surface of an object is defined as the width $w$ of the maximum cube inscribed at $\boldsymbol{p}$ (Fig. 1(b)). In the clearance

analysis, the width of the maximum circumscribed cube is used instead. The cubes are assumed to be aligned with respect to the X-, Y-, and Z-axes of the coordinate frame of the part. Machine designer often requires determining whether a box-like part can be placed in a specified position or whether a part can pass through a specified gap. This method is especially useful for assisting such tasks.

Main Idea:
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 is obtained through the following three steps:

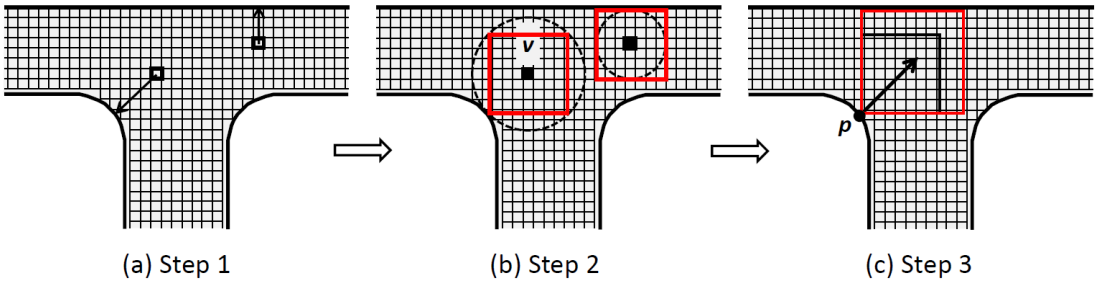


Fig. 3: Outline of the thickness computation process using the maximum inscribed cubes.

**Input:** Polyhedral model of a part shape.
**Step 1:** The input polyhedral model is converted to 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 are recorded for each voxel and a distance field [2] is constructed inside the polyhedral model.
**Step 2:** The distance value recorded in voxel ***v*** represents the radius of the sphere inscribed in the polyhedron and centered at ***v***. Using the information of each inscribed sphere, a cube centered at ***v***, aligned with respect to X-, Y-, and Z-axes, and inscribed in the polyhedron is calculated (Fig. 3(b)). The half value of the width of the cube is recorded as a new distance value for ***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 ***p*** on the surface of the model is determined by extending a ray from ***p*** to the interior of the cuboid distance field and examining the distance values recorded in the voxels on the line (Fig. 3(c)).
**Output:** Maximum cube inscribed at an arbitrary point on the surface, and aligned with respect to the coordinate axes.

For Step 1, we use our previously developed algorithm [3]. Details of Step 2 and 3 are described below.

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

**Initial values:** Axis-aligned cube ***c*** inscribed in ***S*** and another axis-aligned cube ***C*** circumscribed in ***S***.
**Step 2.1:** Obtain an axis-aligned cube ***M*** of midsize of ***c*** and ***C*** and centered at ***v***.
**Step 2.2:** If the difference between the widths of ***c*** and ***C*** is less than the given tolerance value $\varepsilon$, ***M*** is output as a cube whose center point is at ***v*** and inscribed in ***S***.

**Step 2.3:** If ***M*** intersects the polyhedron's surface, ***M*** is set to be new ***C***. If ***M*** does not intersect, ***M*** is set to be new ***c***. Return to Step 2.1.

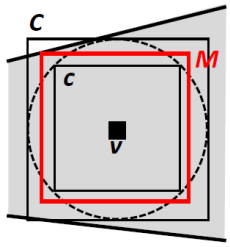


Fig. 4: Computation of ***M*** using bisection method.

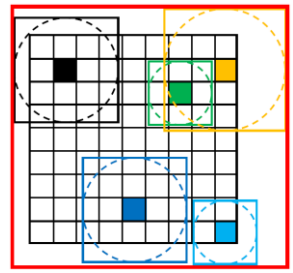

Fig. 5: 512 cubes circumscribed to the spheres and a box containing the cubes.

The half value of the width of the obtained inscribed cube was stored as the new distance value for voxel ***v***. This process is repeated for all voxels to obtain the cuboid distance field.

The most time-consuming process in the above algorithm is the intersection detection between a cube and polyhedron in step 2.3. Axis-aligned bounding boxes (AABB) tree is used to accelerate the calculation. An 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. 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. 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.

The construction of the cuboid distance field can be further accelerated by using the parallel processing capabilities of a GPU. Consider 512 voxels in an $8 \times 8 \times 8$ cuboidal lattice structure. For each voxel, an inscribed sphere centered on the voxel and a cube circumscribed to the sphere are defined, as well as a rectangular box containing all of these cubes (Fig. 5). Using this rectangular box and the AABB tree described above, we select the polygons that may be involved in the intersection detection in the inscribed cube calculation for the 512 voxels. The vertex data of the selected polygons are transferred to the GPU. A GPU thread is invoked for each of the 512 voxels, and the cube centered on each voxel and inscribed to the polyhedron is computed in parallel for 512 voxels using the transferred vertex data.

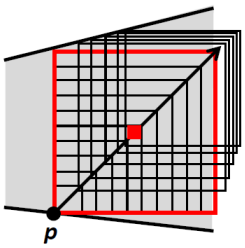


Fig. 6: Detection of the maximum inscribed cube by checking voxels along a ray.

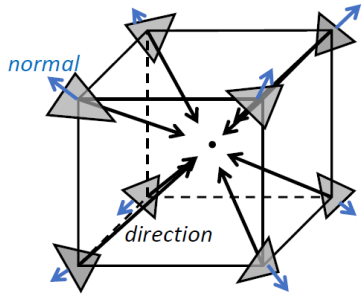


Fig. 7: 8 tracing directions for 8 polygons with different normal directions.

*Determination of Maximum Cube Inscribed to a Surface Point*

The determination of the maximum inscribed cube becomes a process of tracing and checking the distance values in the cuboid distance field along a certain straight line. Fig. 6 illustrates the tracing

process. As shown in the figure, the center points of the cubes inscribed at the surface point ***p*** are aligned on a straight line extending 45 degrees diagonally from ***p***. Therefore, we trace along this line in the cuboid distance field and examine 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 ***p***. The search direction can be classified into eight directions according to the normal direction of the polygon in which ***p*** exists (Fig. 7).

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 *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)

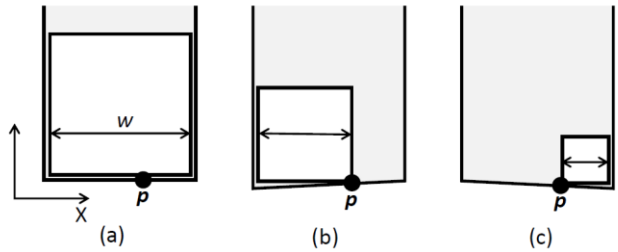


Fig. 8: Limitation of our method for determining the maximum inscribed cube for point ***p***.

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. 8 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 ***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. 8(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.

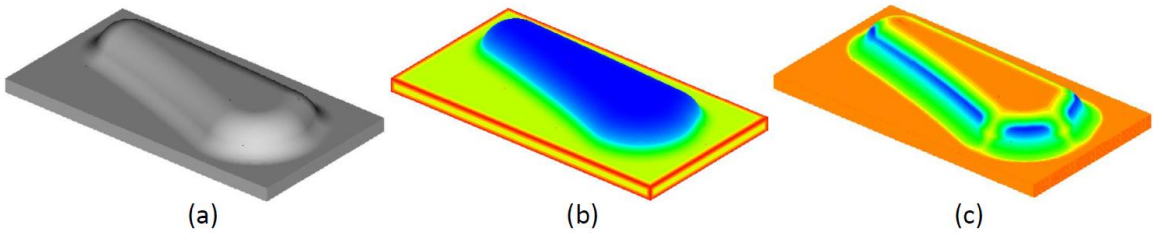


Fig. 9: Results for model A. Thickness evaluation using sphere method (b) and our current method (c).

*Computational Experiment*
Software was implemented to calculate the cuboid distance field using the developed algorithm, and the thickness analysis of the part using the maximum inscribed cubes were conducted. VisualStudio 2017 and CUDA 10.1 were used in the implementation. A notebook PC with a Core i7 CPU, GeForce RTX 3080Ti GPU, and 32GB memory was used for the calculations. Fig. 9 and 10 show the results of the thickness analysis using the software. In these figures, (b) is the result of analysis using the conventional sphere method and (c) shows the result of the newly developed method using the

maximum inscribed cubes. In the calculation, a voxel model with approximately 50 million voxels was used. Necessary computation time for the thickness visualization was 48.66 s for model A and 111.26 s for model B. In these examples, the computation time was about five times longer than with the conventional sphere method.

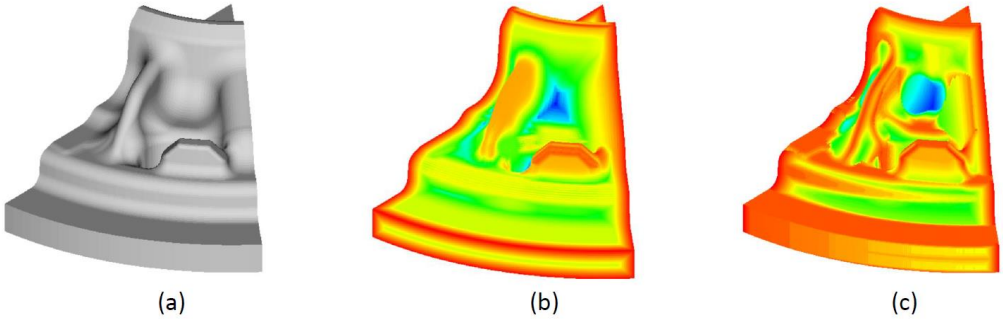


Fig. 10: Results for model B. Thickness evaluation using sphere method (b) and our current method (c).

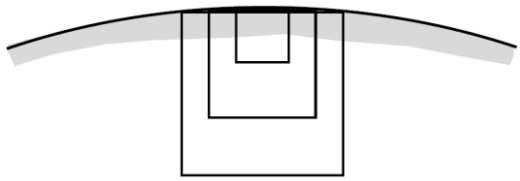

Fig. 11: Maximum inscribed cubes placed on the upper curved surface.

In the analysis, a large number of points ware densely scattered on the surface of the part. Then the thickness is analyzed for each point using the sphere method for (b) and using our current method with the cuboid distance field for (c). In both cases, the maximum thickness is colored blue and zero thickness is colored red. Since our method does not correctly calculate the maximum inscribed cube for surfaces parallel to the X-, Y-, or Z-axes, the coloring in (c) is not correct for those areas. Comparing the analysis results of (b) and (c), several areas with completely different thicknesses can be found, for example, the curved upper part of the model in Fig. 9. This is due to the smaller maximum inscribed cubes placed on the curved surface, as shown in Fig. 11.

Conclusions:

In this paper, we propose a new technique for evaluating the thickness and clearance of a 3D object using the maximum cubes inscribed in the object. We constructed a new distance field called the cuboid distance field and showed an efficient method for thickness analysis using it. We developed the software and conducted several computational experiments to evaluate its performance.

References:

[1] 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

[2] Jones, M. W.; Bærentzen, J. A.; Sr´amek, M.: 3D Distance Fields: A Survey of Techniques and Applications, IEEE Transaction on Visualization and Computer Graphics, 12 (4), 2006, 581–599. https://doi.org/10.1109/TVCG.2006.56

[3] 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