COMPUTER-AIDED DESIGN & APPLICATIONS, 2016
VOL. 13, NO. 2, 199–207
http://dx.doi.org/10.1080/16864360.2015.1084186

Shrinking sphere: A parallel algorithm for computing the thickness of 3D objects

Masatomo Inui, Nobuyuki Umezu and Ryohei Shimane

Ibaraki University, Japan

ABSTRACT
An interactive system is required to enable machine designers to precisely visualize the thickness of a machine part. The thickness of a 3D object at a surface point is given by the diameter of the maximum inscribed sphere (MIS) touching that point. In this paper, we propose a novel iterative algorithm, namely, the *shrinking sphere* algorithm, for computing the MIS at a specific surface point. The convergence speed of the proposed algorithm is very high, and several iterations are usually sufficient for obtaining the MIS. The parallel execution of the algorithm with a graphics processing unit (GPU) is presented for further improving the computation speed. On the basis of the proposed algorithm, an experimental thickness visualization system is implemented using Compute Unified Device Architecture (CUDA). This system can visualize the thickness of a complex object with nearly two million polygons in several minutes using a PC (Core i7 CPU, 32GB memory and GTX-980 GPU), which is sufficiently fast for practical purposes.

KEYWORDS
Thickness visualization; part design; AABB tree; GPU

1. Introduction

Thickness is a basic parameter in machine part design. The thickness of individual walls and ribs is important for calculating allowable stresses and strains of a machine part. In general, modern products are designed to be lightweight by reducing the wall thickness as much as possible. In additive manufacturing, wall thickness is a crucial factor affecting part production because it has a significant impact on the production time and cost. An interactive system is required to enable machine designers to precisely visualize the thickness of 3D objects.

In mechanical drawing, thickness is defined as the distance between points on two opposite parallel surfaces [1]. This definition is not suitable for determining the thickness specification of objects with complex curved shapes. There are several methods that have been developed to define the thickness of complex objects. In the sphere method, which is the most widely used definition of thickness for a 3D object, the thickness t at point $\boldsymbol{p}$ on the object surface is given by the diameter of the maximum inscribed sphere (MIS) touching $\boldsymbol{p}$, as shown in Fig. 1 [17]. Consider a 3D object, the surface of which is finely tessellated with small polygons of uniform size. The thickness of the object can be visualized by computing the MIS touching a point on each polygon, and by painting the polygon with a unique color corresponding to the diameter of the sphere.

In this visualization method, determination of the MIS touching a specific surface point is the most important step. In general, such a sphere is computed using an iterative root-finding method. In the simple bisection method, the sphere radius is repeatedly halved or magnified 1.5 times until the solution sphere is obtained. This method of finding the MIS often requires many iterations before reaching convergence. In this paper, we propose a new iterative algorithm, namely, the *shrinking sphere* algorithm, for computing the MIS touching a surface point of a polyhedral object. A sphere with a sufficiently large radius is given as an initial candidate for the solution. This sphere may have intersections with some surface polygons of the object. By using the geometric data of such polygons, a new candidate sphere with a smaller radius is obtained. This sphere-shrinking process is repeated until the radius change becomes sufficiently small. The convergence speed of our algorithm is very high, and in most cases, no more than five iterations are required for obtaining the MIS touching a surface point. We propose three techniques for further improving the performance of our algorithm: determination of a suitable initial sphere, use of hierarchical bounding volumes for efficiently selecting polygons intersecting the sphere, and use of the parallel-processing capability of a graphics processing unit (GPU) to accelerate the computation of the new sphere.

CONTACT Masatomo Inui masatomo.inui.az@vc.ibaraki.ac.jp

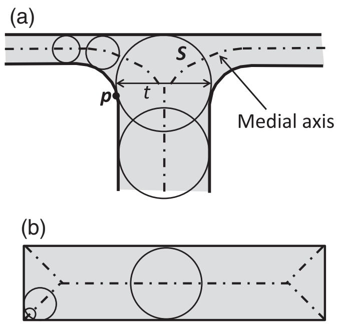

Figure 1. Thickness definition in the sphere method.

The remainder of this paper is organized as follows. Section 2 introduces some definitions of thickness for 3D objects and briefly reviews previous studies on thickness visualization and computation of the MIS. Section 3 describes the process flow of the proposed shrinking sphere algorithm. Section 4 discusses the use of parallel processing techniques with the GPU for improving the performance of the algorithm. Section 5 presents the thickness visualization results for sample objects. Finally, Section 6 summarizes our findings and concludes the paper.

2. Related studies

The two main methods for defining the thickness of a 3D object are the ray method and the sphere method [2,3,17]. In the ray method, the thickness at a point $\boldsymbol{p}$ on a surface is given by a ray shot from $\boldsymbol{p}$ in a direction opposite to the local outward normal. The Euclidean distance d between $\boldsymbol{p}$ and another point $\boldsymbol{q}$ corresponds to the thickness, where $\boldsymbol{q}$ is the intersection point of the ray with the immediately opposite surface of the object (see Fig. 2). This definition is ambiguous if the two surfaces containing $\boldsymbol{p}$ and $\boldsymbol{q}$ are not parallel, because the thickness values at $\boldsymbol{p}$ and $\boldsymbol{q}$ will be different.

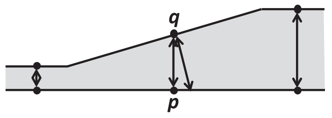

Figure 2. Thickness definition in the ray method.

The sphere method returns consistent results for any object. In this method, the thickness at a point $\boldsymbol{p}$ on a surface is given by the diameter of the MIS touching $\boldsymbol{p}$ (see Fig. 1(a)). Because the locus of the center of the MIS corresponds to the medial axis of the object [7], the thickness at a point corresponds to twice the distance between the point and the medial axis. In some cases, the thickness given by the sphere method is not consistent with the standard definition of thickness. For example, the thickness at a corner of a plate-like shape is smaller than that in the middle, as shown in Fig. 1(b). Subburaj et al. [19] proposed "exterior thickness," a modified version of the sphere method, which uses the skeleton of an object instead of the medial axis to define the thickness. However, this method is not suitable for evaluating the thickness of a thin wedge shape, where the thickness value becomes much greater than expected.

Some commercial CAD systems [16,18] provide thickness visualization functions. Most of these utilize the thickness evaluation software GeomCaliper [6] as an add-on function. GeomCaliper is a system specialized for the thickness visualization of polyhedral solid models; it supports both the ray method and the sphere method. According to the developer's report [17], GeomCaliper employs a uniform grid and k-d tree to achieve effective thickness computation; however, its technical details have not been published. It has been graphically shown in [17] that the thickness analysis time of GeomCaliper is proportional to the square of the number of polygons; therefore, a model with many polygons requires a long computation time.

At present, studies on thickness visualization are not being actively conducted by academic communities. Subburaj et al. proposed thickness analysis based on a voxel model [19]. In addition to "exterior thickness," they proposed two new metrics, namely, "radiographic thickness," a variant of the ray method, and "interior thickness," a type of distance field. A distance field is a voxel-based shape representation in which each voxel records the distance between its center and the object boundary [11]. Lu et al. proposed another distance-field-based thickness evaluation method for detecting thicker zones in 3D objects [12]. Further, we developed a thickness visualization system for a solid model in a previous study [9,10], whereby the thickness at a point on the object surface is determined using the sphere method and a distance field. Because the above-mentioned methods involve expensive distance field computation for thickness visualization, they are not useful for small-scale computers with limited resources.

Computation of the MIS of a 3D object is a challenging theoretical problem. Xie et al. proposed an algorithm for computing the MIS in high-dimensional polytopes, such as a convex polyhedron [20]. Computation of the MIS has also been discussed in the field of tolerance analysis. Sphericity tolerance controls the form error of a sphere-like object. Under such tolerance, the amount of shape deviation is evaluated according to the difference between the radii of two spheres. One bounds the shape internally, whereas the other bounds the shape externally; the MIS of the shape corresponds to the former [5]. In the medical image analysis field, the MIS is used to determine the centerline of the blood vessel models obtained from computerized tomography (CT) scans [15]. These works focused on the computation of the MIS in a general position, and they are not useful for evaluating the object thickness at a specific surface point.

3. Thickness visualization of polyhedral objects

Our method requires a tessellated CAD model, such as a model in the STL format, of an object as the input. Most commercial CAD systems provide a function to output a model of a solid with curved surfaces as a group of triangular polygons with a specified conversion accuracy and polygon size. For example, the user of a CATIA system can control the accuracy and polygon size of the output tessellated model using two parameters referred to as "sag" and "step." Our proposed method visualizes the object thickness by painting each surface polygon with a unique color corresponding to its thickness. The object surface should be finely tessellated with small polygons of uniform size. Because a single thickness value is assigned to each polygon, fine visualization cannot be achieved with a low-resolution polyhedral model.

3.1. Basic algorithm

Thickness visualization is achieved using the following three-step algorithm:

Step 1: For each polygon f, select a point $\boldsymbol{p}$ on the polygon where the thickness of the object is measured. In our current implementation, the center of gravity of the polygon is selected as $\boldsymbol{p}$.

Step 2: The thickness value at $\boldsymbol{p}$ corresponds to the diameter of the MIS touching $\boldsymbol{p}$. Our shrinking sphere algorithm is used in the computation.

Step 3: To complete the visualization, each polygon f is painted with a unique color corresponding to its thickness value.

The processing required for Steps 1 and 3 is straightforward. Therefore, only the shrinking sphere algorithm used in Step 2 is detailed here. In this step, the following processing is repeated for a point $\boldsymbol{p}$ on each surface polygon f of the object to obtain the MIS touching $\boldsymbol{p}$.

Step 2.1 Cast a ray from $\boldsymbol{p}$ along the opposite direction of the surface normal at $\boldsymbol{p}$. The center points of all spheres touching $\boldsymbol{p}$ must be located on the ray.

Step 2.2 As an initial candidate for the target sphere, a sphere with a sufficiently large radius touching $\boldsymbol{p}$ is defined. The MIS touching $\boldsymbol{p}$ is obtained by repeatedly shrinking the initial sphere in the following manner.

- The candidate sphere has some inclusions and/or intersections with the surface polygons of the object. In our algorithm, any case in which the sphere touches or contains a surface polygon is recognized as an intersection.
- For each intersecting polygon f_i except f, a point $\boldsymbol{p}_i$ on the polygon that is closest to the center point $\boldsymbol{c}$ of the sphere is computed. In the case shown in Fig. 3(a), the candidate sphere intersects six polygons. A polygon f_3 is recognized as an intersecting polygon because it touches the sphere surface. Six points, $\boldsymbol{p}_0$, $\boldsymbol{p}_1$, $\boldsymbol{p}_2$, $\boldsymbol{p}_3$, $\boldsymbol{p}_4$, and $\boldsymbol{p}_5$, are computed as the closest points on each of the intersecting polygons.
- Based on the computed points, the next candidate sphere touching $\boldsymbol{p}$ is obtained. For each $\boldsymbol{p}_i$, a new sphere whose center point is on the ray and whose surface passes through both $\boldsymbol{p}$ and $\boldsymbol{p}_i$ is computed (see Fig. 3(b)). From the computed spheres, a sphere with the smallest radius (bold sphere passing through $\boldsymbol{p}_5$ in the figure) is selected as the next candidate. By definition, the new sphere must have a smaller radius than the previous sphere.

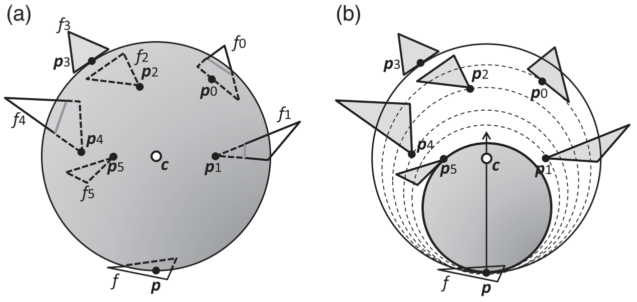

Figure 3. Sphere-shrinking process in Step 2.2.

The sphere-shrinking process described above is iterated until the diameter change between the previous sphere and the new sphere becomes sufficiently small. In our current implementation, the process terminates when the diameter change becomes smaller than $1/10^6$ of the model size. The obtained sphere corresponds to the MIS touching $\boldsymbol{p}$. The diameter of the sphere is considered to be the thickness of polygon f at $\boldsymbol{p}$. The convergence speed of this algorithm for obtaining the solution sphere is very high. In the case shown in Fig. 3(b), the selected sphere in the first iteration becomes the solution. Fig. 4 presents a flowchart summarizing the entire processing flow.

3.2. Initial candidate sphere

In our algorithm, the size of the initial candidate sphere has a significant impact on the computation result and

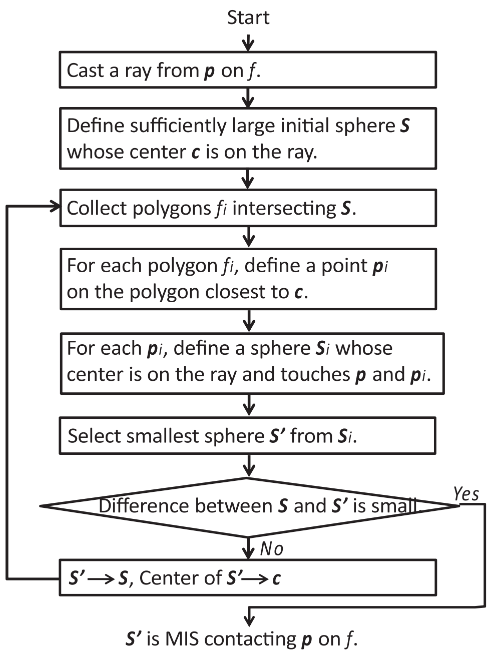
Start
Cast a ray from *p* on *f*.
Define sufficiently large initial sphere *S* whose center *c* is on the ray.
Collect polygons *fi* intersecting *S*.
For each polygon *fi*, define a point *pi* on the polygon closest to *c*.
For each *pi*, define a sphere *Si* whose center is on the ray and touches *p* and *pi*.
Select smallest sphere *S'* from *Si*.
Difference between *S* and *S'* is small
Yes
No
S'→*S*, Center of *S'*→*c*
S' is MIS contacting *p* on *f*.

Figure 4. Flowchart summarizing the shrinking sphere process.

cost. The initial sphere must have a sufficiently large radius; otherwise, it may not have any intersections with the surface polygons, and our algorithm will not work. On the other hand, if the radius of the initial sphere is too large, it will have too many intersections with the surface polygons, and Step 2.2 will therefore require a longer computation time to determine the next sphere.

In our implementation, the initial sphere for a surface point ***p*** is obtained using the ray method. As mentioned in Section 2, this method returns the distance between ***p*** and another point ***q***, where ***q*** is the intersection point of a ray shot from ***p*** with the immediately opposite surface of the object. This distance is used as the diameter of the initial sphere touching ***p***. Such a sphere touches the object surface at ***q*** when the two polygons containing ***p*** and ***q*** are parallel, as in Fig. 5(a). The sphere will penetrate the object surface near ***q*** when the two polygons are not parallel, as in Fig. 5(b). In all cases, an initial sphere can be defined that has some intersecting polygons or a polygon contacting at a location other than ***p***, and the

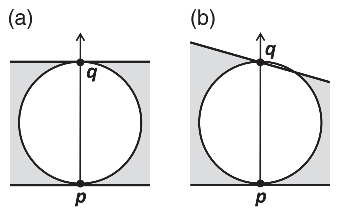

Figure 5. Initial candidate sphere determined using the ray method.

sphere-shrinking process can be executed properly. Most mechanical parts are designed to have constant thickness throughout the surface; therefore it is highly likely that this initial sphere is the MIS touching ***p*** (see Fig. 6), and fast convergence of the sphere-shrinking process can be expected.

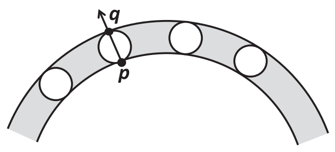

Figure 6. Initial candidate spheres of a part with constant thickness.

Because the STL model does not guarantee watertightness, the ray method may fail to obtain an intersection point on the opposite surface when the ray passes through the gap between polygons. We consider a bounding box that tightly encloses the entire object. Any inscribed sphere of the object must be contained within this box; thus, a sphere whose diameter is equal to smallest length of the box is used as the initial sphere when the ray method fails. Because such a sphere has a very large radius and would probably contain many intersecting polygons, much more time would be required in Step 2.2 to select a sphere with the smallest radius. However, the possibility of the ray passing through a gap is usually very small and the additional computation time caused by the initial large sphere can be ignored in most cases. Based on our experiments, the percentage of rays passing through a gap is less than 0.01% for CAD models of mechanical parts. This percentage increases to 1% for models constructed based on laser-scanned data from real objects because their lack of uniformity inevitably produces errors in the measurements of their shapes.

3.3. Use of hierarchical bounding box

The most time-consuming task in our algorithm is the detection of all surface polygons intersecting the sphere. The hierarchical structure of the axis-aligned bounding box (AABB) [13] is introduced for improving the efficiency of our algorithm. An AABB that tightly encloses the polygons is defined by measuring the coordinate ranges of the polygons in the x-, y-, and z-axis directions. We define a root AABB that encloses all the triangular polygons of the given model. Next, we project the center points of all the polygons in the AABB to a line parallel to its longest axis, and we sort the polygons according to the order of the projected points on the line (see Fig. 7). Then, two AABBs are formed, one by the first $n/2$ sorted polygons and the other by the remaining polygons.

These two AABBs are considered to be the children of the root AABB. The sorting and grouping operations of polygons and the process of defining child AABBs are iterated, and a binary AABB tree is obtained. The tree construction process is terminated when all leaf AABBs of the tree contain only *nmax* or fewer polygons, where *nmax* is the maximum number of polygons allowed for each leaf AABB. In our implementation, we set $nmax = 4$ based on numerical experiments. Each leaf AABB retains the number of polygons within as well as the geometric data (coordinates of vertices) of the polygons.

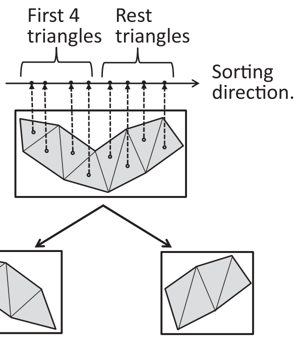

Figure 7. Hierarchical AABB tree construction process.

The AABB tree allows efficient detection of the surface polygons intersecting a sphere. The tree is traversed in the depth-first manner. At each tree node, the distance d between the center point $\boldsymbol{c}$ of the sphere and the AABB corresponding to the tree node is measured using an algorithm proposed in [4]. If d is larger than the sphere radius, no polygon within the box intersects the sphere, and the traversal of its child nodes can be canceled. Otherwise, its children are checked in a recursive manner. In the case shown in Fig. 8, only child AABBs of BBox_0 are traversed because the distance between BBox_0 and the sphere center is smaller than the sphere radius. After the culling operation during the AABB tree traversal, some AABBs at the leaf nodes of the tree are obtained. These AABBs hold polygons sufficiently close to the center point, and they may have intersections with the sphere. Because the oriented bounding box (OBB) can hold the polygons within, a hierarchical OBB tree is generally known to be more efficient in the culling operation [8]. According to our experiments, in most cases, the implementation using the AABB shows 10% better performance in terms of the required computation time than the implementation using the OBB.

The AABB tree is also valuable for efficiently selecting polygons intersecting the ray used in the initial sphere determination process. The tree is traversed in the depth-first manner in this selection. At each tree node, the intersection between the ray and an AABB corresponding to the node is checked. If the ray does not cross the box, no polygons within the box intersect the ray, and the traversal of its child nodes can be canceled. If the ray does cross the box, its child nodes are traversed in a recursive manner to detect a possible crossing between the ray and their holding polygons. After the tree traversal, some leaf AABBs are obtained. Since the polygons within these AABBs are the only ones that intersect with the ray, these are applied in the initial sphere determination process using the ray method.

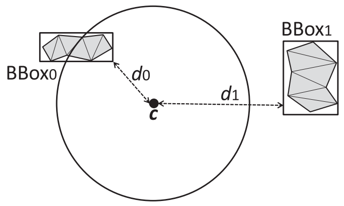

Figure 8. Culling operation with bounding boxes.

4. Parallel computations with GPU

In order to select the polygons intersecting the sphere, point-polygon distance computation is executed between the center point of the sphere and the polygons within the leaf AABBs obtained after the culling operation. For each intersecting polygon f_i except polygon f where the initial given point $\boldsymbol{p}$ is located, a point $\boldsymbol{p}_i$ on the polygon closest to the center point is computed. A new smaller sphere whose center point is on the ray shot from the given point $\boldsymbol{p}$ and whose surface passes through both $\boldsymbol{p}$ and $\boldsymbol{p}_i$ is obtained. These computations for each intersecting polygon are mutually independent; therefore, they can be performed using the GPU in a parallel manner.

The GPU is designed to have hundreds of small streaming processors (SPs) on a chip. These SPs can execute the same instructions with different data in parallel. In the implementation of the parallel distance computation software, Compute Unified Device Architecture (CUDA) is used [14]. CUDA enables programmers to utilize a GPU as a general-purpose single instruction/multiple data (SIMD)-type parallel processor. The main factor responsible for the acceleration gained by the GPU is the replacement of iterative execution of a function in a loop with parallel execution of its equivalent threads on the SPs. CUDA provides a parallel execution framework of threads in a C program. Using CUDA, programmers can describe a code to execute at most $65535 \times 65535 \times 512$ threads simultaneously.

In our current implementation, each thread is assigned to compute the radius of a new sphere based on the polygons in each leaf AABB. More precisely, the following computations are executed by each thread.

1. The thread computes the distance between the center point of the current sphere and each polygon in a leaf AABB corresponding to the thread.
2. If the distance is greater than the sphere radius, then the polygon data is simply discarded because the polygon does not intersect the sphere. If the polygon corresponds to polygon *f* where the initial given point $\boldsymbol{p}$ is located, then this polygon is also omitted from the following computations. Otherwise, a point on the polygon closest to the center point of the sphere is computed for each intersecting polygon.
3. A new sphere is derived by using the coordinates of $\boldsymbol{p}$ and the coordinates of the closest point on the intersecting polygon. The thread repeats the sphere-defining process for all the polygons in its corresponding AABB, and it returns the smallest sphere for the AABB as its result.

Fig. 9 shows four threads invoked for four leaf AABBs. Each thread computes the smallest sphere for its corresponding AABB as shown in the figure.

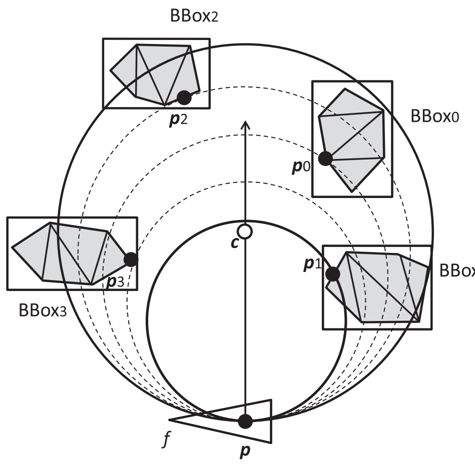

Figure 9. Each thread computes the smallest sphere for its corresponding AABB.

After the execution of all the threads, the smallest sphere among the computed spheres is selected as the next candidate for the MIS touching $\boldsymbol{p}$. In the case shown in Fig. 9, a sphere passing through point $\boldsymbol{p}_1$ is selected from among the four spheres. Because the computations of the MIS for a surface point $\boldsymbol{p}$ and for another point $\boldsymbol{q}$ are mutually independent, the sphere-shrinking operation can be executed for multiple surface points in a parallel manner. In our current implementation, the program simultaneously executes the threads mentioned above for 2048 surface points. This number of parallel processing points was determined by computational experiments in our current computing environment. This number can be increased in the future with a GPU containing more streaming processors.

This implementation is not optimized for fully utilizing the computation ability of the GPU. In the GPU computation, the geometric data of the polygons is stored in the global memory of the graphics board. Because the data access speed of the global memory is much lower than the computation speed of the GPU, the arithmetic unit of the GPU has to suspend processing until the required geometric data is completely transferred from the global memory to the registers of the arithmetic unit. In order to overcome this problem, a new parallel algorithm that uses the fast shared memory mechanism has been proposed; however, it has not been implemented thus far.

5. Computational experiments

Using the shrinking sphere algorithm, we implemented a thickness visualization system based on Microsoft Visual C++ and CUDA 6.5. In our experiments, we employed a Windows 8.1 64-bit PC (Intel Core-i7 4.0 GHz CPU, 32GB main memory, nVIDIA GTX-980 GPU). Fig. 10 shows the results obtained by applying our system to five sample models. In these images, red is assigned to zero thickness and blue is assigned to the maximum thickness of the model. In the visualization result for a cup (Sample A in Fig. 10), the blue zone (thicker zone) is extracted in the inner wall of the cup where the handle is connected outside the cup. As shown in Fig. 1(a), such a connecting part can contain a larger sphere than its adjacent wall part. Yellow lines (thicker zones) are visible in the curved surface in the top part of Sample D (see also Fig. 11(a)). There are many ribs in the other side of the part, and the yellow lines correspond to the connecting part of the ribs. As in the case of the cup, the connecting part of the rib can contain a larger sphere. Fig. 11 shows close-up views of Samples D and E shown in Fig. 10 (they correspond to the dotted squares in Fig. 10). Fine and precise thickness visualization is achieved for surfaces with small polygons.

Table 1 lists the time required to compute the thickness of all the surface polygons of the five sample models. Samples A, B, C, D, and E correspond to the sample models illustrated in Fig. 10. The shrinking sphere process is terminated when the diameter change becomes smaller than a predefined small epsilon value, which was $1/10^6$ of the approximate size of the model for the results presented here. From the table, it is clear that our algorithm achieves very fast convergence. Not more than five iterations are usually sufficient to obtain the MIS. We implemented another MIS computation program using the bisection method. This method used the same initial sphere given by the ray method and the same termination condition of

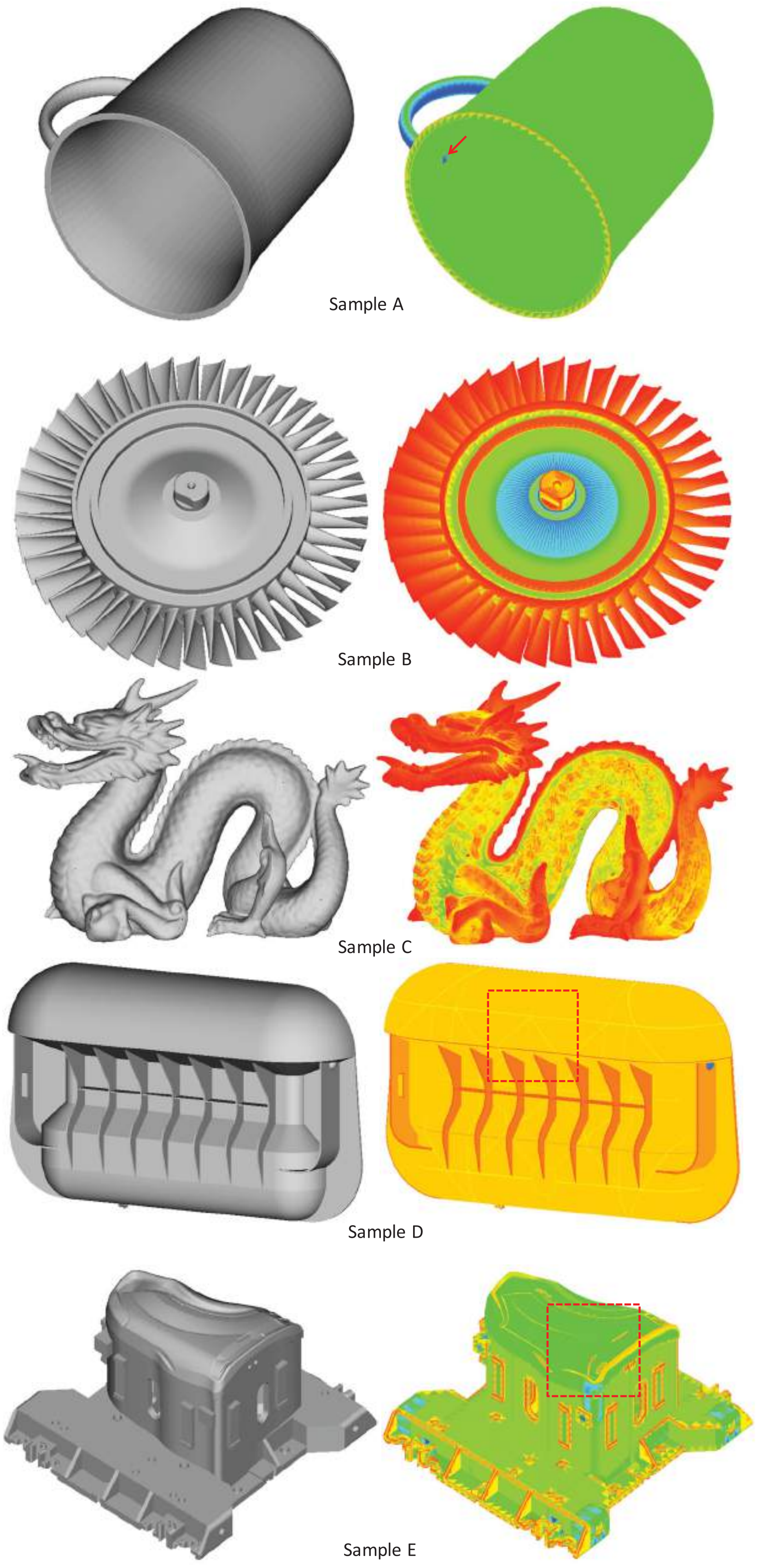

Figure 10. Thickness visualization results for sample parts.

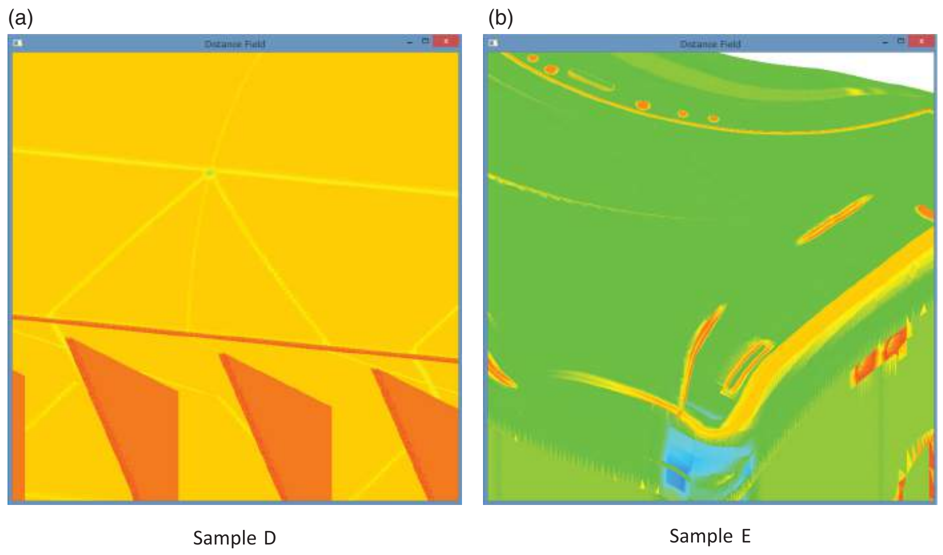

Figure 11. Close-up views of Samples D and E.

Table 1. : Time required for thickness visualization.

Model	Number of polygons	Model size	Average iterations for convergence	CPU comp. t_0(s)	GPU comp. t_1(s)	t_0/t_1
A	17,970	225.5 × 204.8 × 181.0	4.66	1.57	0.93	1.69
B	197,450	969.7 × 967.7 × 305.0	4.90	79.05	25.77	3.07
C	202,520	212.8 × 152.9 × 99.7	4.42	148.76	30.28	4.91
D	1,708,000	72.5 × 344.5 × 210.3	2.78	355.21	159.60	2.23
E	1,972,196	153.2 × 160.2 × 105.1	3.88	1542.29	339.04	4.55

the iteration; however, in general, this program required four or five times more iterations to obtain the MIS for the sample models given in Fig. 10. For example, the average number of iterations for Sample E was 16.40 using the bisection method, which is 4.2 times more than the necessary number of iterations using the shrinking sphere method.

Computation using the GPU was two to five times faster than computation using the CPU, especially for complex objects with numerous polygons. The thickness of an object with nearly two million polygons (Sample E) was computed in several minutes. Fig. 12 shows the performance of our system according to the increase in the number of polygons. The surface polygons of Sample A (uniformly thin shape) and B (thick and complex shape) were subdivided to obtain several new models of the same shape, but with a different number of polygons. These polygons were used in the experiments that obtained the

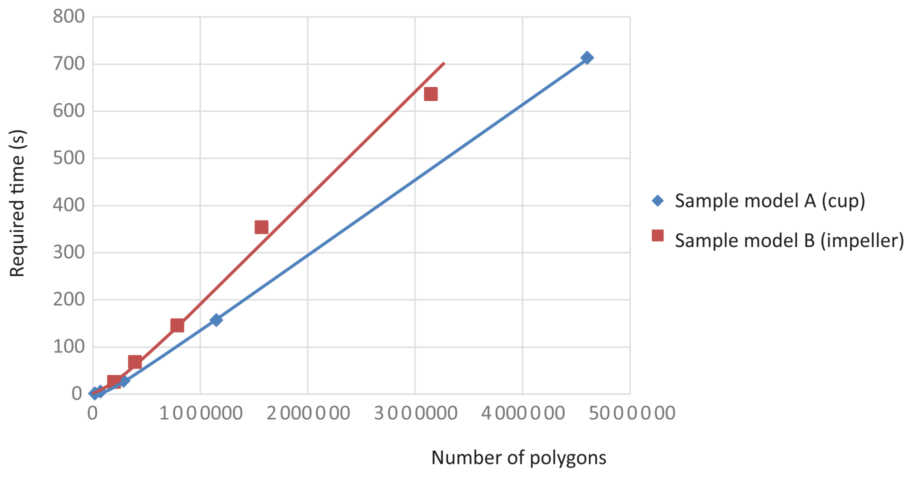

Figure 12. Time required for thickness visualization.

results in Fig. 12. The two curves visible in the graph are not straight lines; however, owing to the slow rate at which their slopes increase, they appear to approximate straight lines. A similar tendency appeared in the computation results for other models.

6. Conclusions

The shrinking sphere algorithm was proposed in this paper as a novel algorithm for calculating the thickness of tessellated models based on the MIS, and the proposed algorithm was applied to the thickness visualization of different samples. In this algorithm, a sphere with a sufficiently large radius touching a surface point is given as the initial candidate. Such a sphere has intersections with surface polygons of the object. Based on the intersecting polygons, a new sphere with a smaller radius is obtained. This sphere-shrinking process is repeated until the radius change between the previous sphere and the new sphere becomes sufficiently small.

The shrinking sphere algorithm has a high convergence speed, and no more than five iterations are usually sufficient for obtaining the MIS. Three additional methods were presented to improve the performance of the proposed algorithm:

- Determination of a suitable initial candidate sphere using the ray method,
- Use of hierarchical bounding volumes for efficiently selecting polygons intersecting the sphere,
- Use of the parallel-processing capability of the GPU to accelerate the computation.

Based on the proposed algorithm, an experimental thickness visualization system was implemented. This system could visualize the thickness of a complex object with nearly two million polygons in several minutes, which is sufficiently fast for practical purposes.

Acknowledgement

This work was supported by JSPS KAKENHI Grant Number 25420081.

ORCID

Masatomo Inui http://orcid.org/0000-0002-1496-7680
Nobuyuki Umezu http://orcid.org/0000-0002-7873-7833

References

[1] ASME: Dimensioning and Tolerancing Y14.5:2009, The American Society of Mechanical Engineers, 2009.
[2] Beiter K. A.; K. Ishii K.: Geometry-Based Index for Predicting Sink Mark in Plastic Parts, Technical Report, ERC/NSM-P-91-61, Engineering Research Center for Net Shape Manufacturing, The Ohio State University, 1991.
[3] Cocks D.: Die Cast Components: Aid to Efficient Design, Zinc Development Association, London, 1983.
[4] Ericson C.: Real-Time Collision Detection, Morgan-Kaufmann, 2004.
[5] Fanwu M.; Chunguang X. U.; Haiming L. I.; Juan H.: Sphericity Evaluation Using Maximum Inscribed Sphere Method, Procedia Engineering, 24, 2011, 737–742. http://dx.doi.org/10.1016/j.proeng.2011.11.2728
[6] GeomCaliper, http://geomcaliper.geometricglobal.com, Geometric.
[7] Giblin P. J.; Kimia B. B.: A Formal Classification of 3D Medial Axis Points and Their Local Geometry, IEEE Transaction on Pattern Analysis and Machine Intelligence, 26(2), 2004, 238–251. http://dx.doi.org/10.1109/TPAMI.2004.1262192
[8] Gottschalk S.; Lin M. C.; Manocha D.: OBBTree: A Hierarchical Structure for Rapid Interference Detection, Proc. of ACM SIGGRAPH, 1996. http://dx.doi.org/171-180.10.1145/237170.237244
[9] Inui M.; Umezu N.; Kobayashi K.: Parallel Distance Field Computation with GPU and Its Application for Evaluating Part Thickness, Proc. of ISCIE/ASME International Symposium on Flexible Automation, 2014, ISFA2014-40.
[10] 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. http://dx.doi.org/10.1016/j.jcde.2015.04.001
[11] Jones M. W.; Bærentzen J. A.; Sŕamek M.: 3D Distance Fields: A Survey of Techniques and Applications, IEEE Transaction on Visualization and Computer Graphics, 12(4), 2006, 881–599. http://dx.doi.org/10.1109/TVCG.2006.56
[12] 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. http://dx.doi.org/10.1016/S0010-4485(97)00015-8
[13] Moller T.; Haines E.: Real-Time Rendering, A K Peters, 1999.
[14] NVIDIA: CUDA Compute Unified Device Architecture, Programming Guide, 2007.
[15] Piccinelli M.; Veneziami A.; Steinman D. A.; Remuzzi A.; Antiga L.; A Framework for Geometric Analysis of Vascular Structures: Application to Cerebral Aneurysms, IEEE Transactions on Medical Imaging, 28(8), 2009, 1141–1155. http://dx.doi.org/10.1109/TMI.2009.2021652
[16] 3DS, http://www.3ds.com, Dassault Systemes, France.
[17] Sinha B.: Efficient Wall Thickness Analysis Methods for Optimal Design of Casting Parts, Engineering Design, 2007, http://geomcaliper.geometricglobal.com/images/file/EfficientWallThicknessAnalysisGeomCaliper.pdf
[18] SOLIDWORKS, http://www.solidworks.com, Waltham, MA.
[19] Subburaj K.; Patil S.; Ravi B.: Voxel-Based Thickness Analysis of Intricate Objects, International Journal of CAD/CAM, 6(1), 2006.
[20] Xie Y.; Snoeyink J.; Xu J.: Efficient Algorithm for Approximating Maximum Inscribed Sphere in High Dimensional Polytope, Proc. of SCG Annual Symposium on Computational Geometry, 2006. http://dx.doi.org/10.1145/1137856.1137861