HOSTED BY

ELSEVIER

Available online at www.sciencedirect.com

ScienceDirect

Journal of Computational Design and Engineering 2 (2015) 183–194

www.elsevier.com/locate/jcde

Thickness and clearance visualization based on distance field of 3D objects

Masatomo Inui, Nobuyuki Umezu*, Kazuma Wakasaki, Shunsuke Sato

Ibaraki University, 4-12-1 Nakanarusawa, Hitachi, Ibaraki 316-8511, Japan

Received 28 January 2015; received in revised form 14 April 2015; accepted 15 April 2015
Available online 28 April 2015

Abstract

This paper proposes a novel method for visualizing the thickness and clearance of 3D objects in a polyhedral representation. The proposed method uses the distance field of the objects in the visualization. A parallel algorithm is developed for constructing the distance field of polyhedral objects using the GPU. The distance between a voxel and the surface polygons of the model is computed many times in the distance field construction. Similar sets of polygons are usually selected as close polygons for close voxels. By using this spatial coherence, a parallel algorithm is designed to compute the distances between a cluster of close voxels and the polygons selected by the culling operation so that the fast shared memory mechanism of the GPU can be fully utilized. The thickness/clearance of the objects is visualized by distributing points on the visible surfaces of the objects and painting them with a unique color corresponding to the thickness/clearance values at those points. A modified ray casting method is developed for computing the thickness/clearance using the distance field of the objects. A system based on these algorithms can compute the distance field of complex objects within a few minutes for most cases. After the distance field construction, thickness/clearance visualization at a near interactive rate is achieved.

Keywords: Inscribed spheres; Spatial coherence; Parallel computation; Axis-aligned bounding box (AABB); Modified ray casting; GPU

1. Introduction

Thickness and clearance are basic parameters in product design. The thickness of individual walls and ribs is important for calculating allowable stresses and strains in functional analysis. In general, modern products are designed to be lightweight by reducing the wall thickness by as much as the required structural strength will permit. Thickness evaluations are important in other design tasks, such as for the shape of the insulator that shields the noise of an automobile engine. The insulator shape must have a sufficient and constant thickness across its surface to reduce the volume of transmitted sound.

Part thickness is important from the viewpoint of manufacturability. In injection molding, hot melted plastic material is forced into a mold cavity so that it cools and hardens to take the shape of the required part. It is difficult to insert this plastic material into very thin wall shapes. If the wall thickness is large and not uniform, local depressions (sink marks) may appear because of the excessive shrinkage of thicker regions during the cooling process [1,2]. To assist the machine designer, some CAD systems provide part thickness visualization functions [3–5].

The thickness of the complementary shape of a part should correspond to the clearance around the part. Sufficient clearance between engine components is necessary for cooling their surfaces using air flow. Moreover, clearance affects the accessibility of cutting tools and fixtures to the part surface during the machining process. Clearance evaluation is an important process for automobile safety. The international regulations state that exterior surface parts that could be contacted by a sphere of radius 50 mm must have a roundness of greater than R2.5 [6,7]. Detecting the sphere contact shape is equivalent to identifying part surfaces with a clearance of greater than 100 mm.

In this paper, we propose a novel method for visualizing the thickness and clearance of three-dimensional (3D) objects in a polyhedral representation. The proposed method employs the distance field of an object for the visualization. Consider a solid object in a box-like space. The 3D distance field of the object is a uniform cell decomposition of the space where at each cubic cell

*Corresponding author. Tel. +81 294 38 5262.
E-mail address: umezu@mx.ibaraki.ac.jp (N. Umezu).
Peer review under responsibility of Society of CAD/CAM Engineers.

http://dx.doi.org/10.1016/j.jcde.2015.04.001

(voxel) in the space, the distance from the center of the voxel to the closest point on the object surface is recorded. In addition, other properties such as the identification number of the closest polygon may be recorded in the voxel. The usage of recorded distance values depends on whether the voxel is internal or external to the object. A distance field with internal voxels is used for thickness visualization, whereas clearance visualization is realized using a distance field with external voxels.

The distance field of a polyhedral object can be obtained via iterative computation of the distance between the center of each voxel and the surface polygons. Polygons on the object surface can be classified into groups according to their proximity. For each group, an axis-aligned bounding box (AABB) that tightly encloses the polygons is defined [8]. Using an AABB tree, i.e., a hierarchical structure of boxes, polygons that are sufficiently close to a given voxel can be selected. Similar sets of polygons are usually selected as close polygons for close voxels. On the basis of this spatial coherence, a novel parallel algorithm is designed in order to compute the distances between a cluster of close voxels and the polygons selected by the culling operation so that the fast shared memory mechanism of the graphics processing unit (GPU) is fully utilized.

Thickness t of a 3D object at point $\boldsymbol{p}$ on the surface is defined as the diameter of the maximum inscribed sphere $\boldsymbol{S}$ contacting the surface at $\boldsymbol{p}$ (see Fig. 1) [3]. Similarly, clearance c at $\boldsymbol{p}$ is defined as the diameter of the maximum circumscribed sphere $\boldsymbol{T}$ externally contacting the surface at $\boldsymbol{p}$. The thickness/clearance of objects can be visualized by distributing points on the visible surface of the objects and painting them with a unique color corresponding to their thickness/clearance values. A novel method, namely, modified ray casting, is developed for computing the thickness/clearance at each surface point. Ray casting is a typical method for visualizing 3D scalar fields. In regular ray casting, a line of sight (ray) through the object is assigned for each pixel. Pixel color is determined by accumulating values in the scalar field along the ray. In our modified method, each ray is cast in the same way as in the regular method until it reaches a point on the object surface. Then, the ray turns in a direction perpendicular to the surface and proceeds into the distance field to detect the first peak value in the field that corresponds to the radius of the maximum inscribed or circumscribed sphere contacting the surface at that point.

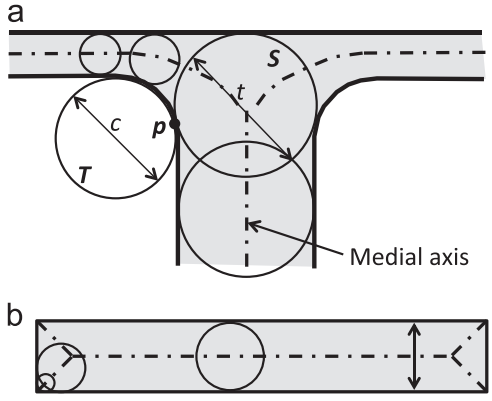

Fig. 1. Thickness/clearance definitions in the sphere method.

The remainder of this paper is organized as follows. Section 2 provides some definitions of thickness and clearance for 3D objects. In addition, it briefly reviews previous studies on distance field computation and thickness/clearance visualization. Section 3 summarizes the contributions of the present study. Section 4 describes a parallel distance field computation algorithm and its implementation using Compute Unified Device Architecture (CUDA) [9], an industry-standard GPU computation environment. Further, it discusses the use of shared memory on the basis of spatial coherence of the distance field. Section 5 describes the modified ray casting method for visualizing the thickness/clearance of objects. Some methods for improving the visualization performance are also discussed. Section 6 presents thickness/clearance visualization results for sample objects. Using the parallel processing capability of the GPU, a distance field with around 80 million voxels can be computed within a few minutes at a sufficiently high speed for practical use. After the distance field is obtained, the thickness/clearance of an object can be visualized at a near-interactive rate by using our modified ray casting algorithm. Finally, Section 7 summarizes our findings and concludes the paper.

2. Related studies

2.1. Thickness/clearance definitions and analysis

In mechanical drawing, thickness is defined as the distance between points on two opposite parallel surfaces. This definition is not suitable for objects with complex curved surfaces. The two major methods for defining the thickness of a 3D object are the ray method and the sphere method [1,3,10]. In the ray method, the thickness at a point $\boldsymbol{p}$ on a surface is given by using a ray originating 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 an intersection point between the ray and the surface immediately opposite to 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 $\boldsymbol{q}$ become different.

The sphere method always returns consistent results. In this method, the thickness at a point $\boldsymbol{p}$ on a surface is given by the diameter of the maximum inscribed sphere contacting the surface at $\boldsymbol{p}$ (see Fig. 1(a)). Since the locus of the center of the maximum inscribed sphere corresponds to the medial axis of the object (dashed lines in the figure) [11], the thickness at a surface point corresponds to twice the distance between the point and the medial axis. In general, the thickness given by the sphere method is consistent with the mechanical drawing definition of thickness for a plate-like shape, except at its corners, where the diameter of the

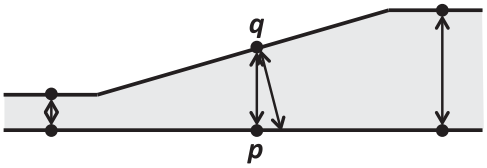

Fig. 2. Thickness definition in the ray method.

maximum inscribed sphere decreases, as shown in Fig. 1(b). Subburaj et al. defined "exterior thickness" by proposing a modification to the sphere method. In this method, the skeleton of an object is used instead of its medial axis to define thickness [10]. However, exterior thickness is not suitable for evaluating the thickness of thin wedge shapes, where the thickness value becomes much greater than expected.

Some CAD systems provide thickness visualization functions [4,5]. In general, their visualization quality is not adequate for precisely understanding variations in the thickness of an object. 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 a report [3], it employs a uniform grid and k-d tree for achieving good thickness computation performance; however, the technical details of this report have not been published. Furthermore, to the best of our knowledge, no commercial system is available for visualizing product clearance.

Thickness/clearance visualization has not been actively researched in academic communities. Subburaj et al. [10] proposed thickness analysis based on a voxel model. In addition to "exterior thickness", they proposed two new metrics: "radiographic thickness", which is based on a variant of the ray method, and "interior thickness", a type of distance transform that will be explained later. Lu et al. [12] proposed thickness analysis based on a distance transform for detecting thicker regions of 3D objects. Previously, we developed a simple thickness visualization system for a solid model [13], whereby the thickness of a polygon was determined by using the sphere method with a distance field.

Most clearance analysis methods developed thus far are specialized for specific manufacturing requirements, for example, the configuration space for robot motion planning [14] and the accessibility cone for machinability evaluation [15,16]. Similarly, in the medical field, some clearance visualization methods are known to assist in the navigation of implants in virtual environments [17,18]. These methods are too specialized for simply visualizing the clearance distance around objects.

2.2. Distance field computation and visualization

We use the distance field for thickness/clearance visualization. Fast algorithms for constructing the distance field have been actively investigated [19]. There are two basic approaches. The first approach is named distance transform, which is based on the propagation of the distance information [20,21]. In this method, voxels on the object boundary are detected and their exact distance values are calculated in the initialization step. The distances are then propagated to the remaining internal or external voxels. The new distance of a voxel is computed from the distances of its neighbors by updating the values according to the pre-defined template [22–24]. Zhao [25] proposed a fast sweeping method for solving the eikonal equation in a discrete manner. Chang et al. [26] followed Zhaofs method for computing a complete distance field. Distance fields obtained by propagation are basically approximations, and they are not suitable for some engineering applications where precise results are necessary.

The second approach is based on the exact computation of the distance between a voxel and the surface polygons. In distance field construction, some distance computations can be discarded on the basis of their spatial coherence. Payne and Toga [27] utilized the coherence by storing polygon data in a hierarchical bounding box. Gueziec [28] extended this method in his Meshsweeper algorithm, where a distance interval is computed for each bounding box. This interval gives the lower bound and upper bound of distances between a voxel and any polygons in the box. If the lower bound of a box is greater than the upper bounds of some other boxes, then all polygons in the box can be ignored in the distance computations.

Each feature (vertex, edge, or facet) of a triangle mesh can be converted into its corresponding characteristic polyhedron containing the points closest to the feature. The computation cost of the distance field can be reduced by classifying the voxels according to the characteristic polyhedrons [29]. Characteristic polyhedrons for all features of objects constitute a 3D Voronoi diagram, which is a partitioning of the 3D space into regions where each region consists of all points that are closer to one feature than to any other [30]. Hoff et al. [31] proposed a Voronoi diagram computation method accelerated by polygon rendering hardware. Sud et al. [32,33] improved their method and used it for the computation of the 3D distance field.

The adaptively sampled distance field (ADF) records the distance values adaptively according to local details, and it stores the data in a spatial hierarchy for efficient processing. In general, an ADF requires less memory than a regular distance field based on uniform cell decomposition. An ADF that stores distances at the cell vertices of an octree was proposed in [34–36]. The distance values in a voxel are derived by the trilinear interpolation of values at the vertices. The ADF is not effective for some engineering applications where additional distance-related properties, for example, identification number of the closest polygon, are required to be stored in each voxel.

Visualization of a 3D scalar field such as a distance field is a typical topic in volume rendering [37]. There are two basic approaches for rendering a scalar field: explicit extraction of the iso-surface in the field and generation of a semi-transparent picture by accumulating the scalar values in the field. Ray casting is used in the second approach [38,39]. In this method, a ray is cast into the voxels of the scalar field, and it samples the voxel values at certain intervals. The sampled values are accumulated along the ray for determining the opacity required to produce a semi-transparent picture.

In volume rendering, the distance field is usually not a visualization target, but it is used to facilitate visualization or computation of other information in the 3D space. For example, a distance field is used for computing an offset surface [40]. The distance field is also useful in ray casting. The interval size for sampling points along the ray is critical to the rendering performance in ray casting. By using the distance information recorded in the voxels in advance, an interval size that is guaranteed not to penetrate the object boundary can be determined [41]. To the best of our knowledge, no report on the visualization of object thickness/clearance based on the distance field exists in the literature.

3. Contributions of the present study

In this paper, we propose a fast and precise method for visualizing the thickness/clearance of 3D objects on the basis of their internal/external distance field. Thus, the contributions of our work are as follows.

Thickness/clearance visualization based on distance field: The major contribution of the present study is development of a novel method for precisely visualizing the thickness/clearance of 3D objects based on the distance field. The sphere method is used for measuring the thickness/clearance of the objects. In our previous system [13] and in commercial systems such as GeomCaliper [3–5], the thickness of objects is visualized by painting each polygon with a single color corresponding to its thickness value. In general, the visualization quality of this method is low if the object is roughly tessellated with large polygons or polygons of non-uniform sizes. Our method visualizes the thickness/clearance of objects on the basis of points that densely cover the visible surfaces of the objects. For each point, a maximum inscribed or circumscribed sphere contacting the surface at that point is computed by the modified ray casting method, and the point is painted with a unique color corresponding to the thickness/clearance value (i.e., the diameter of the contacting sphere).

Parallel computation of distance field: Most critical precondition of using our thickness/clearance visualization method in practice is fast construction of precise distance field. We use the concept of spatial coherence for reducing the computation cost of the distance field in the case of a uniform resolution. In contrast to previous methods proposed by Payne and Toga [27] and by Gueziec [28], our method uses a distance range between a cluster of voxels and the bounding box of polygons in the culling operation. The geometric data of the polygons that remains after the culling operation are stored in the fast shared memory mechanism of the GPU. They are efficiently used by threads, each of which computes the distance between a voxel in the cluster and the polygons. The parallel processing capabilities of GPUs are advancing at a rapid pace. Hence, we believe this approach is more promising than other methods using traditional depth buffer hardware (for example [32, 33]). We adopt the initial concept of the method described in [13]. The method presented in this paper uses axis-aligned boxes to enclose the voxel cluster and surface polygons in order to improve the culling performance in the distance field computation.

4. Parallel distance field computation

4.1. Preparations

Our method requires a tessellated CAD model of objects as the input. Most commercial CAD systems provide a function to output the model data as a group of triangular polygons, for example, in the STL file format. The model is decomposed into a set of small cubic cells (voxels) according to a uniform spatial grid. Each voxel is classified as an internal or external voxel according to the position of its center with respect to the object boundary; if the center of a voxel lies on the boundary, the voxel is classified as both internal and external.

Data conversion from the input polyhedral model to its equivalent voxel model is performed using the vertical ray method. Consider an axis-aligned box-like space that tightly encloses the given model. The space is subdivided using a uniform axis-aligned spatial grid with equal intervals. The grid is projected onto the xy-plane. From each projected grid point, an upward ray is extended along the z-axis direction, and the intersection points of the ray with the surfaces of the given polyhedral model are computed. The intersection points are sorted according to their z-coordinates, and a set of segments corresponding to the internal part of the object on the ray are obtained. Spatial grid points located on the segments are selected as the center points of the internal voxels.

The computation of a model with external voxels is performed in a similar manner. In this case, the box enclosing the polyhedral model is expanded by D (10% of the largest axis of the box) by shifting the six rectangular surfaces of the box outwards. As in the case of the model with internal voxels, vertical rays are generated, and the intersection points of the rays with the surface polygons are computed. Based on the obtained points, segments corresponding to the external part of the object are derived. The grid points located on the segments correspond to the center points of the external voxels.

The resolutions of the spatial grid are determined such that the total number of generated internal (external) voxels is close to a predetermined number m. In the current implementation, m is set to 80 million, near maximum number of voxels allowed in our computing environment. A parallel algorithm for converting a polyhedral model into its corresponding voxel model is implemented. A program based on this algorithm can convert a complex polyhedral model with 2 million polygons into its equivalent voxel model with 80 million cells in less than 5 s.

To prepare data that is suitable for the proposed method, the surface polygons of the input model are classified into small groups according to their proximity. This classification proceeds by using an AABB tree, i.e., a hierarchical structure of boxes [8]. Consider n triangular polygons forming the model surface. Define a root AABB that encloses all triangular polygons of the given model. Polygons in the AABB are sorted along a line parallel to its longest axis. Then, two child AABBs are formed by the first $n/2$ sorted polygons and the remaining polygons. The process of defining child AABBs is iterated, and a binary AABB tree is obtained. The tree construction process is terminated when all leaf AABBs of the tree contain only n_{max} or fewer polygons, where n_{max} is the maximum number of polygons allowed for each leaf AABB. In our implementation, we set $n_{max} = 4$ 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.

4.2. Culling of unnecessary distance computations

The basic process in the distance field construction is the computation of the distance between the center of a voxel and a polygon on the object surface. The computation cost can be

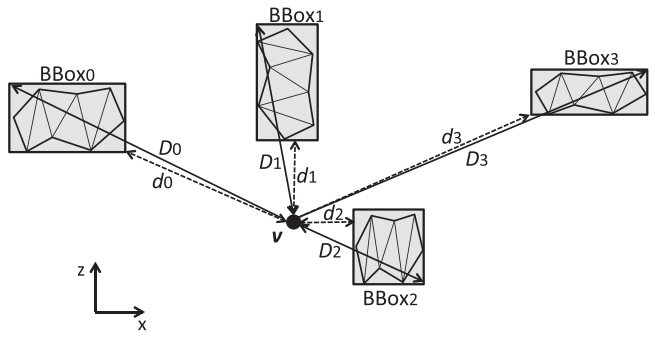

Fig. 3. Culling operation in a layer in an AABB tree.

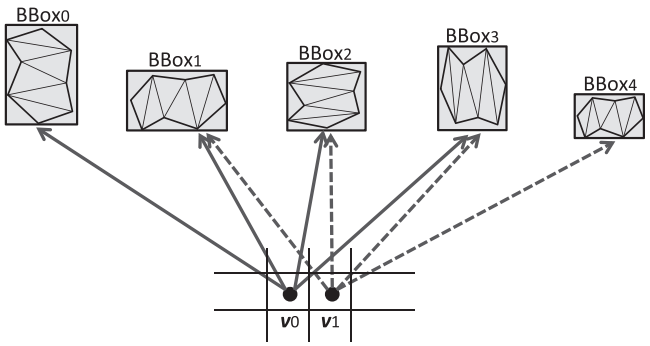

Fig. 4. Spatial coherence of polygons for close voxels.

reduced by using the hierarchical AABB tree. Traverse the tree from the root node to the leaf nodes in the breadth-first manner. The following culling operation is executed in each layer. Assume that n bounding boxes ($\mathrm{BBox}_0, \mathrm{BBox}_1, \ldots, \mathrm{BBox}_{n-1}$) are found in a layer as shown in Fig. 3. Here, d_i represents the shortest distance between a voxel center $\boldsymbol{v}$ and a point on the surface of BBox_i, whereas D_i represents the longest distance between $\boldsymbol{v}$ and a point on BBox_i. Since polygons within BBox_i are bounded by the box, the distance between $\boldsymbol{v}$ and any polygon in BBox_i must be greater than d_i and less than D_i.

Select a box BBox_{min} whose value D_{min} is the smallest among the D_i values. If d_i is greater than D_{min}, then the distance between $\boldsymbol{v}$ and any polygon within BBox_i must be greater than the distance between $\boldsymbol{v}$ and any polygon within BBox_{min}; therefore, polygons within BBox_i can be ignored in the distance field construction. Fig. 3 shows that four bounding boxes are found in a certain layer during breadth-first AABB tree traversal. Further, D_2 of BBox_2 is the smallest among D_0, D_1, D_2, and D_3. Since d_0 and d_3 are greater than D_2, polygons within BBox_0 and BBox_3 are not relevant to the shortest distance for $\boldsymbol{v}$, and they can be excluded from the following computation. In the next layer in the breadth-first tree traversal, child bounding boxes of only BBox_1 and BBox_2 are evaluated in the culling operation.

4.3. Parallel distance computations with GPU

After the culling operation during the AABB tree traversal, some AABBs are obtained at the leaf nodes of the tree. These AABBs enclose polygons sufficiently close to the given point $\boldsymbol{v}$. The point-polygon distance computation is finally applied to the polygons within these obtained AABBs and $\boldsymbol{v}$. The distance between $\boldsymbol{v}$ and the polygons is computed using the GPU in a parallel manner.

A GPU consists of hundreds of small streaming processors (SP) on a chip. The main factor underlying GPU acceleration is the replacement of the iterative execution of a function in a loop with the parallel execution of its equivalent threads on SPs. CUDA is designed to provide a parallel execution framework of threads in a C program [9]. In order to properly manage the threads, CUDA provides grid and block structures. A block is a 1D, 2D, or 3D array of threads. The maximum number of threads in a single block must be less than or equal to 512. A grid is a 1D or 2D array structure of blocks. The total number of blocks must be less than or equal to 65,535 for each dimension.

The data used by the GPU must be initialized in the main memory for the CPU and transferred to the global memory (device memory) for the GPU. The data stored in the global memory is "globally" accessible from any SP. After the computation, the result is written back to the global memory. Eight SPs constitute a streaming multi-processor or SM (Recent GPU architecture features more SPs in single SM). Each SM corresponds to a block of threads, and threads in the same block are executed by SPs in a single SM corresponding to the block. A SM includes shared memory, which is a special "global" memory accessible only by SPs in the same SM. Because of the correspondence between SPs in an SM and threads in a block, threads in the same block can use the shared memory as a global memory. Owing to the high data-access speed of the shared memory, effective use of the shared memory is crucial to realizing a high-performance CUDA program.

4.4. Use of spatial coherence

In the GPU computation, the geometric data of the polygons is stored in the global memory. The data access speed of the global memory is much slower than the computation speed of the GPU; therefore, the arithmetic unit of the SP has to suspend processing until the required geometric data is completely transferred from the global memory to the registers of the SP. This speed limitation can be overcome by using the fast shared memory mechanism.

In the distance field computation, polygons in the same leaf AABB are often selected for voxels in close proximity. Fig. 4 shows this characteristic. In this figure, $\boldsymbol{v}_0$ and $\boldsymbol{v}_1$ are the center points of two adjacent voxels. Owing to their closeness in the voxel model, polygons close to $\boldsymbol{v}_0$ are usually also close to $\boldsymbol{v}_1$. After the AABB tree traversal, BBox_0, BBox_1, BBox_2, and BBox_3 are obtained as the leaf AABBs sufficiently close to $\boldsymbol{v}_0$, and BBox_1, BBox_2, BBox_3, and BBox_4 are obtained as the leaf AABBs sufficiently close to $\boldsymbol{v}_1$. Since BBox_1, BBox_2, and BBox_3 are close to both $\boldsymbol{v}_0$ and $\boldsymbol{v}_1$, the polygon data in these 3 AABBs can be gsharedh by threads for computing the distance for $\boldsymbol{v}_0$ and other threads for computing the distance for $\boldsymbol{v}_1$. By transferring the data of these polygons to the shared memory from the global memory before the distance computation, and by using the data in the shared memory for the computation, a significant improvement in performance can be realized.

The actual implementation for using the shared memory is as follows. By using the grid structure of the voxel model, a cluster of voxels with $8 \times 8 \times 8$ resolution is obtained as the voxels in close proximity. Our parallel distance computation software derives the distances for these 512 ($=8 \times 8 \times 8$) voxels simultaneously. An axis-aligned cube tightly bounding these voxels is defined. Instead of a point, this cube is used for retrieving leaf AABBs that enclose the polygons sufficiently close to the voxels. In order to cull AABBs using the cube, it is necessary to compute the shortest distance and longest distance between the cube and an AABB. As shown in Fig. 5, the shortest distance d between a cube and an AABB is computed by using their gap distances in each axis direction. The largest distance D corresponds to the longest diagonal length of an axis-aligned box enclosing the cube and the AABB within. The same AABB culling rule explained in the previous section is applicable to the AABB tree traversal, except that it involves the use of a cube of the voxel cluster instead of a point. New shortest and longest distance metrics between a cube and an AABB are used for selecting the AABBs to be culled.

After the culling operations, a set of leaf AABBs enclosing polygons sufficiently close to the cluster of 512 voxels is obtained. A CUDA program is defined to compute the shortest distance for these 512 voxels with respect to the polygons in the obtained leaf AABBs. In our software, a block of threads is defined for each leaf AABB, and each thread in the block is defined to compute the shortest distance between the center of a voxel in the cluster and polygons in the leaf AABB corresponding to the block (see Fig. 6). For these block and thread definitions, all threads in the same block compute distances with respect to the polygons in the same AABB; therefore, they can share the polygon data during the processing operations. Such polygon data are copied from the global memory to the shared memory of the SM corresponding to the block for realizing fast data access in the distance computation.

5. Thickness/clearance visualization with distance field

In our method, the thickness/clearance of 3D objects is visualized by using the distance field. The visualization is achieved using the following three-step algorithm:

Step 1: Generate dense points completely covering visible surfaces of the object.

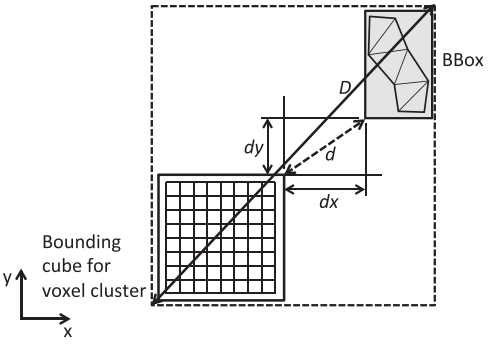

Fig. 5. Shortest distance d and longest distance D between two axis-aligned boxes.

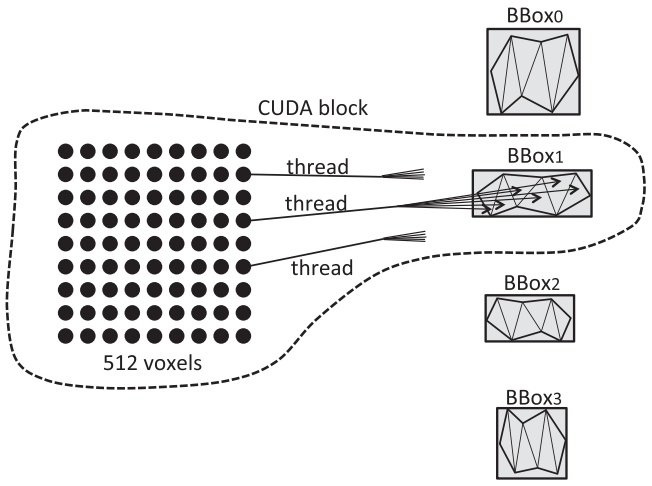

Fig. 6. Block and thread definitions.

Step 2: For each point on the visible surfaces, the thickness/clearance value at the point is computed by using the modified ray casting method.

Step 3: Each point is painted with a unique color according to the thickness/clearance value to complete the visualization.

Details of each step are provided in the following subsections along with an example for thickness visualization with the distance field of internal voxels. The same algorithm is applicable to clearance visualization.

5.1. Point generation on visible surfaces

Points densely covering the visible surfaces of the objects can be generated on the basis of the information of the viewing frustum for projecting 3D shapes onto two-dimensional (2D) frame buffer [42]. Projection is a process for transforming (x, y, z) coordinates of 3D objects into 2D (i, j) coordinates, where i and j denote the pixel position in the frame buffer. A viewing frustum represents the prismatic region of space in the modeling domain that may appear on the display. A local coordinate frame for the projection is defined in association with the viewing frustum. Its origin coincides with the eye position, and its x-axis and y-axis are aligned with respect to the horizontal direction and vertical direction of the display. The z-axis is oriented such that its negative direction is the same as the viewing direction.

In the following explanation, the use of the orthogonal projection is assumed in the rendering process. Our point generation concept is not limited to the orthogonal projection; it is also applicable to the perspective projection with small modifications. Consider a frame buffer after rendering an image from which the hidden surfaces of objects have been eliminated. For each pixel in the frame buffer, a point is generated on the visible surfaces of the objects. The x and y coordinates of the point in the local coordinate frame are computed on the basis of the pixel position (i, j), the resolution of the display, and the left, right, top, and bottom clipping plane information of the viewing frustum. The z coordinate is determined by the depth value of the pixel, and the near and far clipping plane information of the frustum. The obtained coordinates in the projection coordinate frame can be

transformed into coordinates in the modeling domain by a simple matrix operation.

In our thickness/clearance computation method that is explained in the next subsection, the surface normal vector is required at the visible points. For each surface polygon, a unique identification number of 24 bits is assigned. Before the rendering, the normal vector of the polygon is recorded in association with its identification number. In the rendering operation, each polygon is painted with a unique color corresponding to its identification number. Such colors are generated by recognizing the 24-bit identification number as a series three 8-bit numbers representing red, green, and blue components of the color. After the rendering, the (r, g, b) information of each pixel is obtained from the frame buffer. Its corresponding identification number is determined by a simple inverse operation from (r, g, b) to a 24-bit number. The normal vector at the visible point corresponding to the pixel is obtained by retrieving the recorded normal vector using the identification number as the key.

5.2. Maximum inscribed sphere in distance field

Consider the determination of the maximum inscribed sphere contacting a visible surface at point $\boldsymbol{p}$. The diameter of the sphere corresponds to the thickness at $\boldsymbol{p}$ in the sphere method. Fig. 7 shows the distance field of internal voxels of an object. For simplicity, distances are denoted as integer values in the figure. The actual values are floating point numbers. Cast a ray from $\boldsymbol{p}$ in a direction opposite to the surface normal at $\boldsymbol{p}$. The center points of spheres contacting the surface at $\boldsymbol{p}$ must be located on the ray.

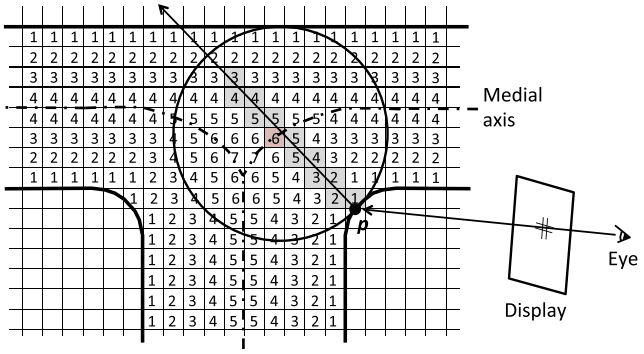

Fig. 7. Modified ray casting method.

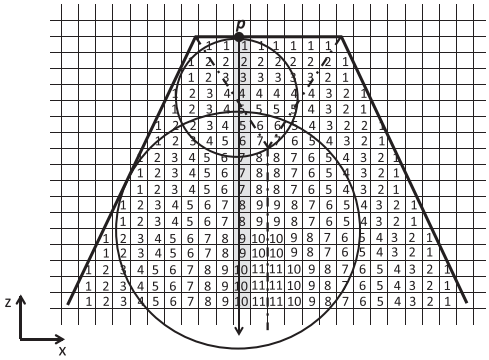

Fig. 8. Necessity of an additional termination condition for ray casting.

Distance values are sampled along the ray at certain intervals. At each sampling location, the distance value is computed by interpolation of distance values at the voxels around the location. This value represents the radius of the inscribed sphere whose center point is at that location. If the location is sufficiently close to $\boldsymbol{p}$, the obtained sphere contacts the object surface at $\boldsymbol{p}$.

As the sampling points move into the object, the distance value increases and a larger inscribed sphere can be defined. The maximum inscribed sphere contacting the surface at $\boldsymbol{p}$ is obtained when the sampling location reaches the medial axis of the object. After passing the medial axis, the distance value at the sampling location starts decreasing. The maximum inscribed sphere contacting the surface at $\boldsymbol{p}$ is thus determined by tracing the ray in the distance field to detect the first peak value of the distance. In the clearance computation at a surface point, a ray is cast from the point along the surface normal direction. It is traced in the distance field with external voxels to detect the first peak value in the field. The obtained distance value corresponds to the radius of the maximum circumscribed sphere contacting the surface at that point.

We refer to the sphere determination method described above as gmodified ray castingh. This method generally returns reasonable thickness/clearance values; however, it is not a perfect strategy. Fig. 8 shows an exceptional case. Consider a vertical protrusion (the figure shows the section). A ray is cast along the negative z-axis direction from a point $\boldsymbol{p}$ at the top. In this case, the distance value continues to increase after the ray passes the medial axis. After passing the medial axis, the derived sphere does not contact the surface at $\boldsymbol{p}$, as shown in Fig. 8. In order to reject such cases, an additional condition for terminating the tracing is necessary.

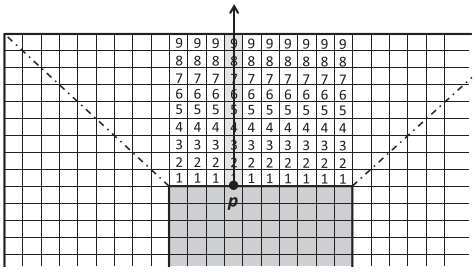

Fig. 9. Requirement of additional termination condition of ray casting in clearance visualization.

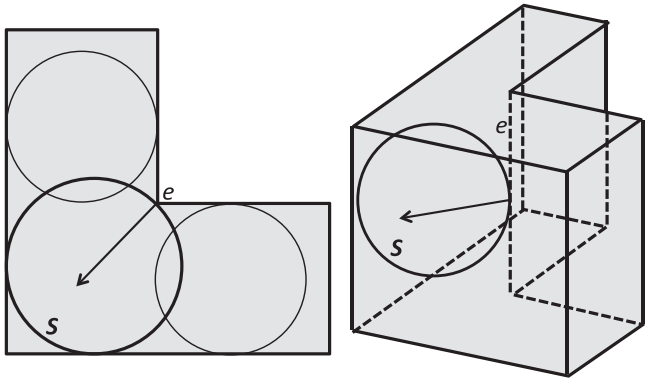

Fig. 10. Casting direction of a ray for computing the radius of a sphere contacting a concave edge e.

Since the sphere must contact the starting point $\boldsymbol{p}$, the distance value at the sampling point on the ray must be equal to the distance dist between the sampling point and $\boldsymbol{p}$. If dist becomes greater than the distance value at the sampling location, the ray has already passed the medial axis and the tracing operation must terminate.

In the clearance visualization, a ray is cast from a surface point along the normal vector direction at that point. No local peak value is detected for some surface points, for example, a point $\boldsymbol{p}$ in Fig. 9. As shown in the figure, distance values along the vertical ray starting from $\boldsymbol{p}$ simply increase until it reaches the border of the distance field. In order to address this situation, we further modify the ray casting algorithm. The local peak value detection process in the clearance visualization terminates when the value becomes greater than $D-\varepsilon$, where ε corresponds to the cell size. D is the expansion size of the axis-aligned box defined in the conversion process from the input polyhedral model to a cell decomposition model with external voxels.

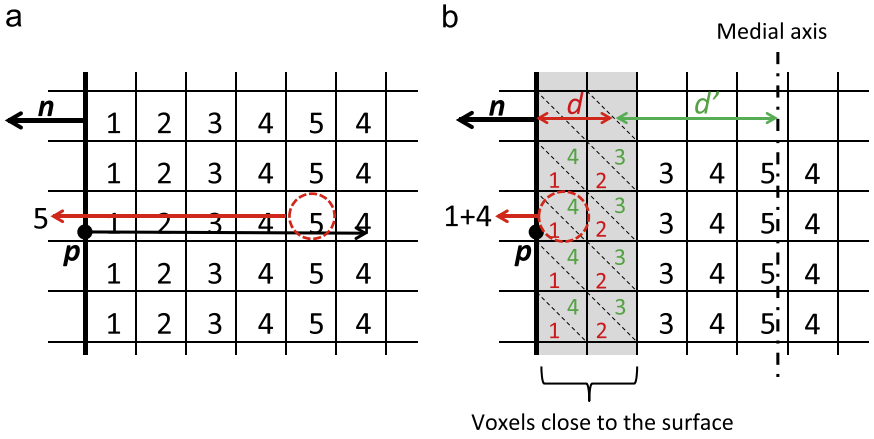

Fig. 11. Improvement for reducing the thickness computation time.

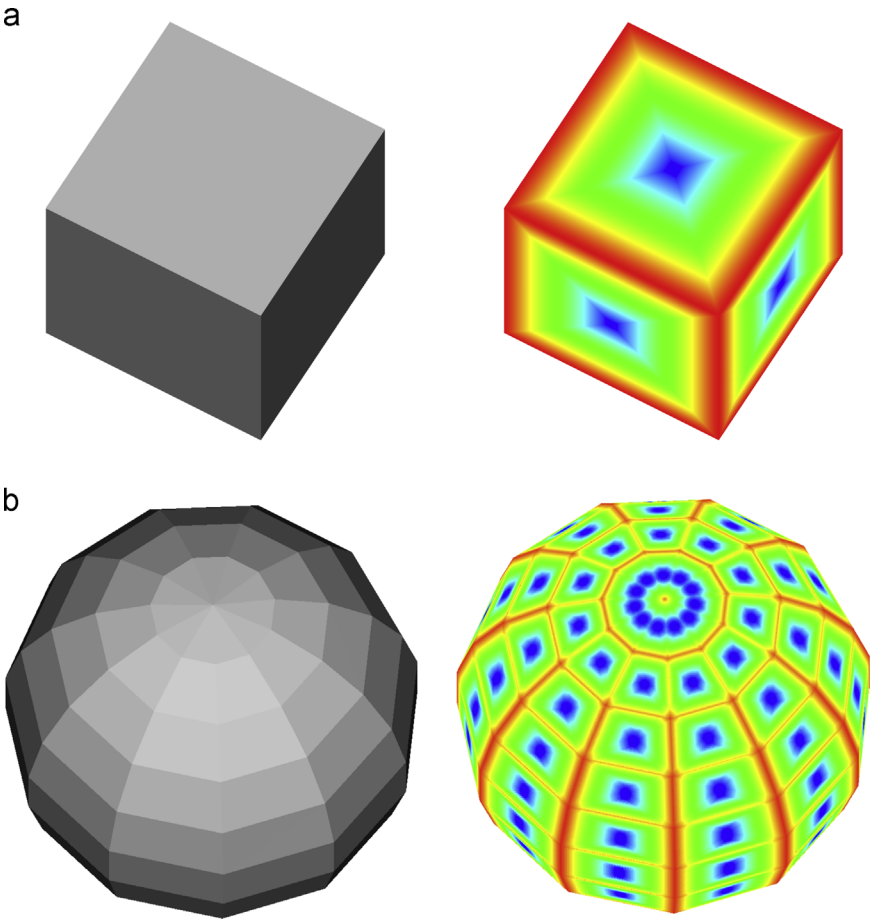

Fig. 12. Thickness visualization results of simple models.

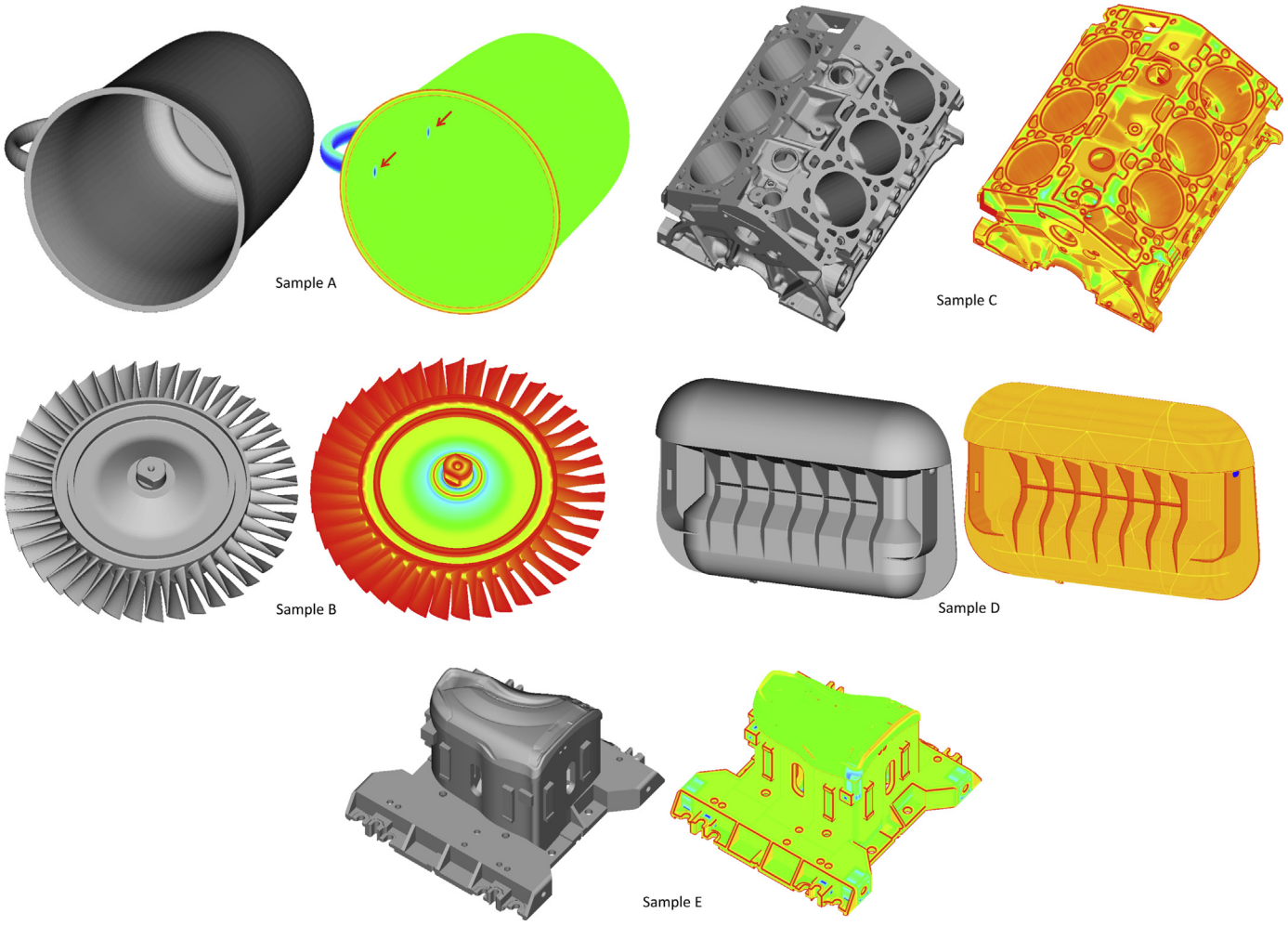

Fig. 13. Thickness visualization results of sample parts.

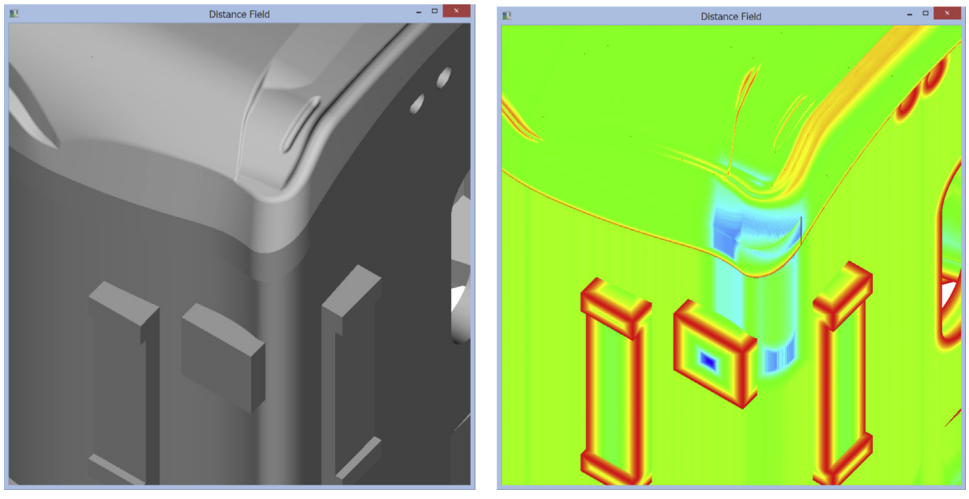

Fig. 14. Close-up view of sample E.

Since the casting distance of the ray from the surface point is limited by the value D in the clearance visualization, the distance computation for external voxels located farther than D from the object surface is not necessary in constructing the distance field. In the culling operation using the AABB tree, which has been explained in Sections 4.2 and 4.4, we introduce an additional

condition specialized for the distance computation of external voxels. In each layer in the breadth-first traversal of the AABB tree, the shortest distances between the cube enclosing the voxel cluster and all AABBs in the layer are computed. If the distance between the cube and an AABB is greater than D, then the polygons in the AABB can be ignored in the distance computation of the external voxels in the cube because they do not contribute to the clearance visualization result. This new culling condition serves to effectively reduce the computation cost of the distance field of external voxels.

The proposed visualization algorithm has one limitation. In the thickness visualization case, the maximum inscribed sphere can contact a concave edge e of an L-shaped object (as shown in Fig. 10). This sphere (sphere $\boldsymbol{S}$ in the figure) has a slightly larger diameter than the other spheres contacting its adjacent polygons. Therefore, the thickness along e becomes larger than that of the polygons. Because our current method casts a ray in a direction opposite to the surface normal, it cannot derive a larger thickness value along e, where no surface normal can be obtained. To overcome this limitation, we must improve our algorithm so that it can cast a ray from a point on such concave edges. A good candidate for the direction of this ray is the bisector of the corners of two adjacent polygons of the edge (see Fig. 10).

5.3. Performance improvements

Since the thickness computation for each surface point is mutually independent, the use of the parallel processing capability of the GPU is a natural choice for improving the visualization performance. Herein, we develop another mechanism for improving the visualization performance. The most time-consuming task in the visualization is the searching operation for a local peak value in the ray casting (see Fig. 11(a)). This task is necessary whenever the viewing position and direction are changed. In order to implement faster thickness visualization, we modify the definition of the distance field. In the original definition, each voxel records a distance value to the closest boundary polygon. In addition to the distance value d, another distance value d' between the voxel and the medial axis is recorded in voxels that are sufficiently close to the object surface (see Fig. 11(b)). Once such information is recorded in the cells, the ray casting operation from a surface point $\boldsymbol{p}$ can be quickly terminated at such voxels near the surface, and it can return $d+d'$ as the radius of the maximum inscribed sphere contacting the surface at $\boldsymbol{p}$.

After the distance field has been computed, voxels sufficiently close to the object surface are selected. In the current implementation, voxels whose distance value is less than 10% of the

Table 1
Required time for distance field construction for thickness visualization.

Model	Number of polygons	Resolution	Number of internal voxels	Time for voxelization (s)	Time for DF computation (s)
A	17,970	1238 × 1122 × 993	82,181,034	1.28	44.50
B	197,450	1265 × 1263 × 397	81,638,305	1.48	55.94
C	86,686	986 × 812 × 593	81,280,879	1.18	68.44
D	1,708,000	470 × 2249 × 1363	82,832,480	3.21	84.58
E	1,972,196	1007 × 1053 × 691	81,005,288	3.98	104.59

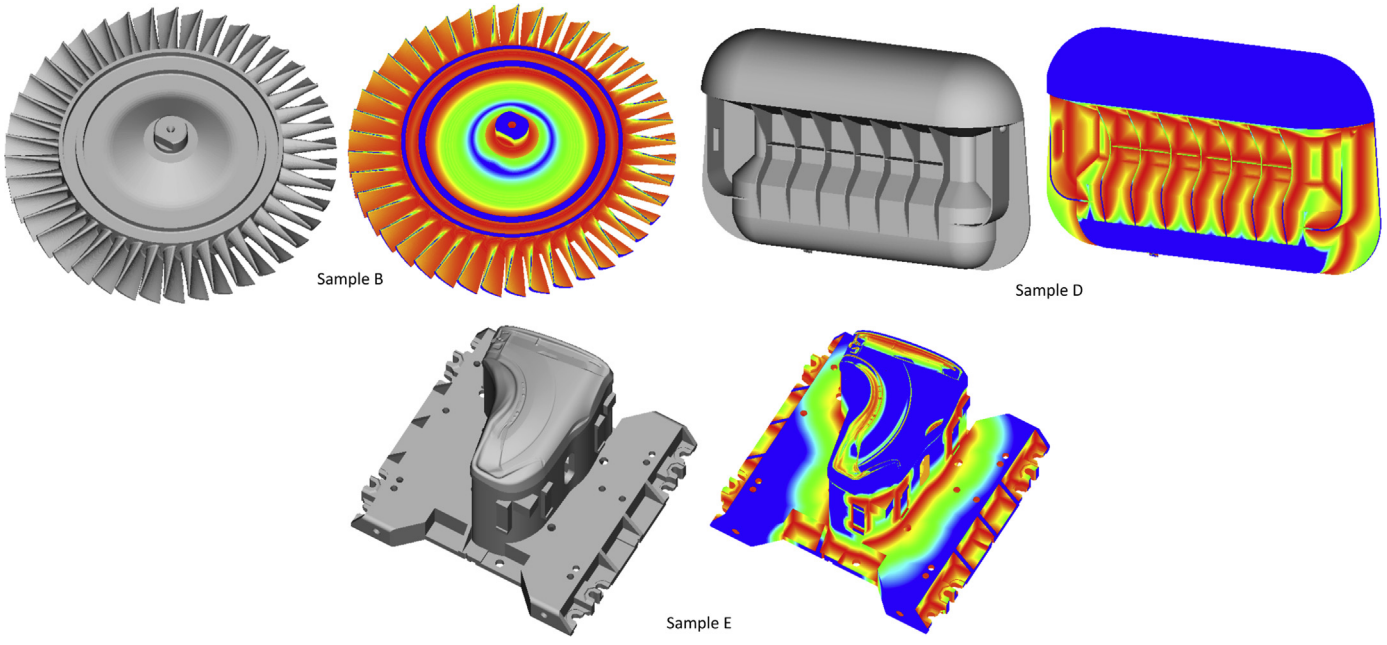

Fig. 15. Clearance visualization results of sample parts.

Table 2
Required time for the distance field construction for the clearance visualization.

Model	Number of polygons	Resolution	Number of external voxels	Time for voxelization (s)	Time for DF computation (s)
A	17,970	$469 \times 437 \times 400$	79,858,211	0.45	59.57
B	197,450	$554 \times 553 \times 271$	79,916,355	0.58	153.37
C	86,686	$518 \times 450 \times 364$	79,553,466	0.53	81.83
D	1,708,000	$269 \times 652 \times 462$	79,904,943	0.74	794.26
E	1,972,196	$471 \times 487 \times 362$	79,567,659	1.52	876.26

maximum thickness are assumed to be close to the object surface. For the center point of each selected voxel, the modified ray casting operation is applied to detect the local peak value in the distance field along the ray. The obtained distance value minus the original value d stored in the voxel corresponds to the distance d' between the voxel and the medial axis. This additional distance information is recorded in the voxel to enable future fast visualizations. The ray direction must be specified in the modified ray casting operation. The closest polygon to each voxel can be detected while computing the distance field. The negative direction of the closest polygon's surface normal $\boldsymbol{n}$ becomes the ray direction for the voxel, as shown in Fig. 11.

6. Computational experiments

A distance field computation system and a thickness/clearance visualization system were implemented using Microsoft Visual C++ and CUDA 6.5 respectively. In our computational experiments, a Windows 8.1 64-bit PC with Intel Core i7 4.0 GHz CPU, 32 GB main memory, and nVIDIA GTX-980 was used.

Fig. 12 shows the thickness evaluation results of a cube and a sphere in rough tessellation. Before the distance field computation, the models were converted into their equivalent voxel representations. The resolution of the voxel model was determined such that the total number of internal voxels was nearly 80 million. In Fig. 12, red is assigned to zero thickness and blue is assigned to the maximum thickness. Expected thickness visualization results were obtained for the models. As explained earlier (Fig. 1(b)), thinner zones (red zones) with small inscribed spheres are extracted along the edges of the cube and the sphere.

Fig. 13 shows the results obtained by applying voxels our system to rather complex sample models. In the visualization result of the cup model (sample model A in Fig. 13), two blue zones (thicker zones) are extracted in the inner wall of the cup where the handle is connected to the cup. As shown in Fig. 1(a), such a connecting part can voxel contain a larger sphere than its adjacent wall part. In the injection molding of a plastic cup, such connecting parts are known as typical zones containing sink marks because thicker voxel zone shrinks more than thin walls. By using the thickness voxel visualization, a designer can detect possible sink marks in the early shape design stage. Fig. 14 shows a close-up view of the sample part E shown in Fig. 13. Very fine and precise thickness voxel visualization is realized. For displaying the pictures, a window of 768×768 pixels is used. By using the modified ray casting program with the improvements voxel described in Section 5.3, the required time for displaying a picture is less voxel than 200 ms, which is sufficiently short for viewing the thickness from various direction and positions at a near-interactive rate.

Table 1 lists the time required for computing the distance field (DF) for internal voxels of sample models. As shown in the table, the distance field in the case of a uniform resolution with nearly 80 million voxels is computed within a few minutes, sufficiently fast for practical use. We execute the distance field computation using a system based on our previous method [13] under the same condition. The performance of the new method is at most 10% better than our previous method, especially for computing the distance field of complex models with near 2 million polygons (sample models D and E).

Fig. 15 shows some examples of the clearance visualization results. Here, red is assigned to zero clearance and blue is assigned to clearance D, which corresponds to the expansion value of the bounding box defined for the conversion of the polyhedral model to the voxel model. As shown in the figure, the narrow zones between the impeller blades in sample B, the narrow zones between the ribs in sample D, and the corner areas in sample E are painted red, as expected. Table 2 lists the required time for computing the distance field for the clearance visualization. Even though the distance computation of external voxels requires more time than that for internal voxels, it is sufficiently fast for practical applications.

7. Conclusions

This paper proposed a novel thickness/clearance visualization algorithm based on the distance field. Our method visualizes the thickness/clearance of 3D objects by using the points densely covering the visible surfaces of the objects. For each point, a maximum inscribed or circumscribed sphere contacting the surface at that point was computed, and it was painted with a unique color corresponding to the thickness/clearance value. The radius of the maximum contacting sphere was determined by our modified ray casting algorithm with the distance field. Further, some methods for improving the rendering speed were proposed. By using the improved method, thickness/clearance visualization at a near-interactive rate was achieved.

In addition, a parallel algorithm for constructing the distance field of a solid model using the GPU was developed. This algorithm is fast, precise, and applicable to complex models with numerous polygons and high-resolution voxels. In the distance field construction, the shortest distance between a point and the surface polygons of the model was computed many times. Similar sets of polygons are usually selected as close polygons for close voxels. Using this spatial coherence, the proposed parallel algorithm was designed to compute distances between a cluster of close voxels and the polygons selected by the culling operation with hierarchical AABB tree.

Conflict of interest

None declared.

References

[1] Beiter KA, Ishii K. Geometry-based index for predicting sink mark in plastic parts. Technical Report: ERC/NSM-P-91-61, Columbus (OH): Engineering Research Center for Net Shape Manufacturing, The Ohio State University; 1991; 140 p.

[2] Cocks D. *Die cast components: aid to efficient design*. London: Development Association Zinc; 1983.

[3] Sinha B. Efficient wall thickness analysis methods for optimal design of casting parts [Internet], Mumbai (India); 2007. Available from: ⟨http://geomcaliper.geometricglobal.com/images/file/EfficientWallThicknessAnalysis_GeomCaliper.pdf⟩[cited 2007].

[4] SOLIDWORKS [Internet], Waltham (MA); 2015. Available from: ⟨http://www.solidworks.com/⟩ [cited 22.01.2015].

[5] Dassault Systemes [Internet]. Velizy-Villacoublay, France; 2015. Available from: ⟨http://www.3ds.com/⟩[cited 22.01.2015].

[6] GlobalAutoRegs [Internet]. UN Regulations; 2015. Available from: ⟨http://www.globalautoregs.com/unece⟩ [cited 13.04.2015].

[7] Japan Automobile Standards Internationalization Center (JASIC) [Internet]; 2014. Available from: ⟨http://www.jasic.org/e/index_e.htm⟩[cited 27.09.2014].

[8] Moller T, Haines E. *Real-time rendering*. Natick (MA): A K Peters; 1999.

[9] nVIDIA. CUDA compute unified device architecture programming guide [Internet], Santa Clara (CA); 2007. Available from: ⟨http://developer.download.nvidia.com/compute/cuda/1_0/NVIDIA_CUDA_Programming_Guide_1.0.pdf⟩[cited 2007].

[10] Subburaj K, Patil S, Ravi B. Voxel-based thickness analysis of intricate objects. *International Journal of CAD/CAM* 2006;**6**(1).

[11] Giblin PJ, Kimia BB. A formal classification of 3D medial axis points and their local geometry. *IEEE Transaction on Pattern Analysis and Machine Intelligence* 2004;**26**(2)238–51.

[12] Lu SC, Rebello AB, Miller RA, Kinzel GL, Yagel R. A simple visualization tool to support concurrent engineering design. *Computer-Aided Design* 1997;**29**(10)727–35.

[13] Inui M, Umezu N, Kobayashi K. Parallel distance field computation with GPU and its application for evaluating part thickness. In: *Proceedings of ISCIE/ASME 2014 International Symposium on Flexible Automation*; 2014 July 14–16; Hyogo, Japan; Article No. 40.

[14] Latombe JC. *Robot motion planning*. Boston (MA): Kluwer Academic Publishers; 1991. 651 p.

[15] Spitz SN, Spyridi AJ, Requicha AAG. Accessibility analysis for planning of dimensional inspection with coordinate measuring machines. *IEEE Transactions on Robotics and Automation* 1999;**15**(4)714–27.

[16] Morimoto K, Inui M. A GPU based algorithm for determining the optimal cutting direction in deep mold machining. In: *Proceedings of IEEE International Symposium on Assembly and Manufacturing*; 2007 July 22–25; Ann Arbor (MI); p. 203–8.

[17] Demiralp C, Marai GE, Andrews S, Laidlaw DH, Crisco JJ, Grimm C. Modeling and visualization of inter-bone distances in joints. In: IEEE Visualization, Work in Progress Sessions; 2001 October 21–26; San Diego (CA); p. 24–5.

[18] Dick C, Burgkart R, Westermann R. Distance visualization for interactive 3D implant planning. *IEEE Transaction of Visualization and Computer Graphics* 2011;**17**(12)2173–82.

[19] Jones MW, Baerentzen JA, Sramek M. 3D distance fields: *a survey of techniques and applications*. *IEEE Transaction on Visualization and Computer Graphics* 2006;**12**(4)581–99.

[20] Rosenfeld A, Pfaltz JL. Sequential operations in digital picture processing. *Journal of the ACM* 1966;**13**(4)471–94.

[21] Cuisenaire O. Distance transformations: fast algorithms and applications to medical image processing [Ph.D. Thesis]. Louvain-la-Neuve (Belgique): Universite Catholique de Louvain; 1999; 213 p.

[22] Rhodes F. Discrete Euclidean metrics. *Pattern Recognition Letters* 1992;**13**(9)623–8.

[23] Danielsson PE. Euclidean distance mapping. *Computer Graphics and Image Processing* 1980;**14**(9)227–48.

[24] Mullikin JC. The vector distance transform in two and three dimensions. *CVGIP: Graphical Models and Image Processing* 1992;**54**(6)526–35.

[25] Zhao H. A fast sweeping method for Eikonal equations. *Mathematics of Computation* 2004;**74**(250)603–27.

[26] Chang B, Cha D, Ihm I. Computing local signed distance fields for large polygonal models. *Computer Graphics Forum* 2008;**27**(3)799–806.

[27] Payne BA, Toga AW. Distance field manipulation of surface models. *Computer Graphics and Applications* 1992;**12**(1)65–71.

[28] Gu'eziec A. Meshsweeper: *dynamic point-to-polygonal mesh distance and applications*. *IEEE Transactions on Visualization and Computer Graphics* 2001;**7**(1)47–61.

[29] Sigg C, Peikert R, Gross M. Signed distance transform using graphics hardware. In: *Proceedings of IEEE Visualization*; 2003 October 19–24; Seattle (WA); p. 83–90.

[30] Okabe A, Boots B, Sugihara K. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. New York NY: John Wiley & Sons; 1992. 696 p.

[31] Hoff KE, Culver T, Keyser J, Lin M, Manocha D. Fast computation of generalized voronoi diagrams using graphics hardware. In: *Proceedings of ACM SIGGRAPH*; 1999 August 8–13; Los Angles (CA); p. 277–86.

[32] Sud A, Manocha D. Fast distance field computation using graphics hardware. UNC Computer Science Technical Report: TR03-206, Chapel Hill (NC): University of North Carolina; 2003; 26 p.

[33] Sud A, Govindaraju N, Gayle R, Manocha D. Interactive 3D distance field computation using linear factorization. In: *Proceedings of the Symposium on Interactive 3D Graphics and Games*; 2006 Mar 14–17; Redwood City (CA); p. 117–24.

[34] Frisken SF, Perry RN, Pockwood AP, Jones TR. Adaptively sampled distance fields: a general representation of shape for computer graphics. In: *Proceedings of ACM SIGGRAPH*; 2000 July 23–28; New Orleans (LA); p. 249–54.

[35] Kim YJ. Exact and adaptive signed distance fields computation for rigid and deformable models on GPUs. *IEEE Transaction on Visualization and Computer Graphics* 2014;**20**(5)714–25.

[36] Bastos T, Celes W. GPU-accelerated adaptively sampled distance fields. In: Proceedings of IEEE International Conference on Shape Modeling and Applications; 2008 June 4–6; Stony Brook (NY); p. 171–8.

[37] Watt A, Policarpo F. *The Computer Image*. Boston MA: Addison-Wesley; 1998. 784 p.

[38] Levoy M. Efficient ray tracing of volume data. *ACM Transaction on Graphics* 1990;**9**(3)245–61.

[39] Ney DR, Fishman EK, Magid D, Drebin RA. Volume rendering of computed tomography data; principles and techniques. *IEEE Computer Graphics and Applications* 1990;**10**(2)24–32.

[40] Liu S, Wang CCL. Fast intersection-free offset surface generation from freeform models with triangular meshes. *IEEE Transaction on Automation Science and Engineering* 2011;**8**(2)347–60.

[41] Hard JC. Sphere tracing: *a geometric method for the anti-aliased ray tracing of implicit surfaces*. *The Visual Computer* 1996;**12**(10)527–45.

[42] OpenGL Architecture Review Board. OpenGL(R) Reference Manual: The Official Reference Document to OpenGL. Boston (MA): Addison-Wesley; 1999; 704 p.