Proceedings of DETC'08
ASME 2008 Design Engineering Technical Conferences and Computers and Information in Engineering Conference
New York City, New York, August 3 - 6 2008

DETC2008/CIE-49576

LAYERED DEPTH-NORMAL IMAGES FOR COMPLEX GEOMETRIES – PART TWO: MANIFOLD-PRESERVED ADAPTIVE CONTOURING

Charlie C.L. Wang
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong, China
cwang@mae.cuhk.edu.hk

Yong Chen
University of Southern California
Los Angeles, CA 90089, United States
yongchen@usc.edu

ABSTRACT

We present an adaptive contouring approach to generate contour surface from solid models represented by Layered Depth-Normal Images (LDNI) sampled in three orthogonal directions. Our contouring algorithm builds an octree structure for mesh generation in a top-down manner: starting from the bounding box of a LDNI solid model, the cells are recursively subdivided into smaller sub-cells based on the topology and geometry criteria of refinement until both of the requirements, the topology in cell is simple and the geometry approximation error is less than a user defined tolerance, are satisfied. The subdivision also stops when the processed cells reach the finest resolution of LDNI models. In order to overcome the topology ambiguity inside a cell that leads to the occurrence of non-manifold entities, we analyze the possible inside/outside configurations of cell-nodes and exploit two strategies to generate manifold-preserved mesh surfaces. Moreover, the most time-consuming step of our contouring algorithm – the construction of octree structure can be easily parallelized to run under a computer framework with multiple-processors and shared memory. Several examples have been tested in the paper to demonstrate the success of our method.

KEYWORDS: adaptive contouring, two-manifold preserved, parallel implementation, implicit representation, solid modeling.

1. INTRODUCTION

The manipulation of solid models is widely used in many applications of design, manufacturing, visualization, analysis, and entertainment. Benefited from the compact and intuitive mathematical representation, the solid modelling operations developed on implicit representation are usually robust and easy to implement. However, it is in general time-consuming to convert models between the boundary representation (B-rep) and the implicit representations. The newly proposed Layered Depth-Normal Images (LDNI) representation in [1] is a sparse implicit representation that can be easily obtained from a B-rep model with the help of graphics hardware accelerated sampling (e.g., using the depth peeling process in [2]). Nevertheless, the LDNI to mesh conversion technique presented in [1] works on the uniform grids at the finest resolution of LDNI, so that it generates too many polygons and is very time-consuming. As the solid models still need to have the B-rep in many downstream CAD/CAM applications (e.g., CNC tool path generation, rapid prototyping, parting line generation of mold design, etc.), the lack of an efficient contouring method blocks the application of LDNI representation in CAD/CAM. This motivates our work presented here – to develop an adaptive contouring method for LDNI solid models.

Furthermore, many existing methods in literature may generate nonmanifold surfaces (e.g., [3-5]). More specifically, on the resultant mesh surface, some of the approaches will generate edges shared by more than two polygons, or vertices the neighbourhood of which is not topologically equivalent to a disk. Although there have been several approaches that claim to produce manifold contours (ref. [6-8]), nonmanifold edges and vertices can still appear in the adaptive setting when working on LDNI with a limited resolution. This is different from directly constructing the adaptive hierarchical structure from B-rep or analytical implicit surfaces, where the resolution of sampling can be unlimited.

To solve the above problems, we propose a top-down refinement algorithm to generate an octree for contouring a solid model represented by Layered Depth-Normal Images (LDNI). The refinement of cells is recursively given until the topology in a cell is simple and the geometry approximation error of the generated mesh to the LDNI solid is less than a user defined tolerance. In order to generate manifold-preserved mesh surface, we analyze the possible inside/outside configurations of cell-nodes and exploit two strategies to construct meshes inside a cell – one method prefers to generate gaps and the other tries to create thin-shells in the smallest cells. Without changing the method too much, the most time-consuming step of our adaptive contouring algorithm – the octree construction is modified to a parallel algorithm running under a computer framework with multiple processors and shared memory.

1.1 Related Work

The work presented in this paper relates to several previous researches in the areas of: volume-based representation of solid modelling, contouring method for the volumetric representation, dexel-representation and contouring, and the Layered Depth Image (LDI).

To overcome the robustness problem in the solid modelling approaches based on B-rep (see [9] and [10] for the surveys), many approaches adopted volumetric data to approximate the solid modelling operations (e.g., [4, 11, 12]). Voxel based solid model [13] is the simplest volumetric representation, which is soon replaced by the uniform and adaptive distance-fields [14]. However, as mentioned by Kobbelt et al. in [15], even if an over-sampling is applied, the aliasing error along sharp features on a distance field cannot be absolutely eliminated since the surface normals in the reconstructed model usually do not converge to the normal field of the original model. Therefore, recently developed volumetric approaches always encode both the distance and the normal vectors which are called *Hermite data* (e.g., [3, 6, 7, 8, 15]). The samples stored in LDNI are also a sort of Hermite data (see [1]).

There are dozens of algorithms in literature trying to generate two-manifold polygonal mesh surfaces from a volumetrically represented solid model. The Marching Cubes (MC) algorithm [16] is the first approach in the literature to generate a polygonal mesh surface from an implicit surface. The latter variations of MC algorithms (e.g., [17-20]) all devoted to solve the topological inconsistent problems that may generate gaps or nonmanifold entities on the resultant meshes. In order to reconstruct sharp edges and corners on the resultant meshes, the authors in [3-8] employed the dual contouring techniques to generate an isosurface on Hermite data. Among them, the methods in [6, 7] adopted the subdivision method to ensure that the finest cell contains only simple topological structure. However, this method cannot be applied here as the implicit representation in LDNI has limited resolution, therefore the cells cannot be split into sub-cells without limit. Also, different from [8] the purpose of which is to cluster vertices as so to simplify a given mesh surface, there is not an underlying mesh surface to govern the preservation of manifold topology here. Thus, some new criteria for manifold preservation need to be investigated.

Another line of research related to our work is the so-called ray-rep in the solid modelling literature [21-24]. Menon and Voelcker sampled the solid models into parallel rays tagged with h-tag (i.e., the information of half-space at the endpoints of rays) in [22] so that the completeness of ray-rep can be generated. The conversion algorithm between ray-rep and B-rep or CSG is also given in [22]. As mentioned in [23], ray-rep can make things easy in the applications involving offsets, sweeps, and Minkowski operations. However, different from our LDNIs that sample a solid model along three orthogonal directions together with normal vectors, the ray-rep only stores depth values without surface normals in one ray direction. Furthermore, the algorithm presented in [22] to convert models from ray-rep to B-rep does not take the advantage of structurally stored information so it involves a lot of global search and could be very time-consuming. The recently developed method in [24] used a local searching algorithm to reconstruct B-rep from the ray-rep (i.e., dexels). Again, they worked on the finest resolution, thus surfaces with too many polygons are generated. Furthermore, the preservation of two-manifold has not been considered in [24].

The Layered Depth-Normal Images (LDNI) representation is stimulated by the work presented in [2], where a fast collision detection approach for solid models was developed based on the decomposition of Layered Depth Images (LDI) [25]. Similar to the LDI decomposition, the implementation of LDNI sampling can be accelerated in graphics hardware using OpenGL (ref. [1]).

1.2 Contributions

The techniques developed in this paper contribute in the following three aspects:

- We present a new contouring algorithm that adaptively generates mesh surfaces to approximate the boundary of solid models represented by Layered Depth-Normal Images (LDNI) with the help of octree structure.
- The construction of mesh surface inside a leaf cell of the octree is well designed so that the two-manifold topology is preserved on the resultant surface – two strategies (one preferring gaps and the other for creating thin-shells) are exploited.
- A parallel implementation for the octree construction from LDNI, which is the most time-consuming step in the contouring procedure, is also investigated.

These lead to the function that allows us to efficiently and effectively reconstruct the boundary representation – mesh surface from a LDNI solid model.

The remainder of the paper are organized as follows. To be self-contained, the LDNI representation of solid models is briefly explained in section 2. Section 3 describes our adaptive

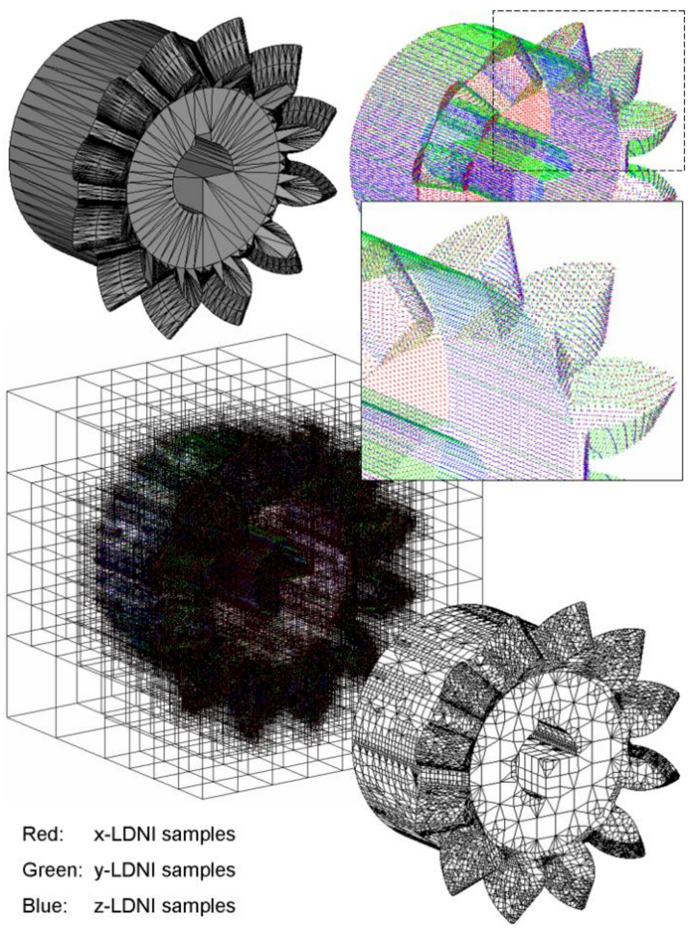

Figure 1: A gear model sampled into LDNI representation with 129×129 resolution (top-right), the octree (bottom-left) for contouring and the resultant mesh surface (bottom-right).

contouring algorithm which consists of two steps: 1) the octree construction and 2) the mesh generation. The topology of mesh surfaces generated inside a leaf cell is analyzed in section 4, and two methods for constructing manifold-preserved mesh surfaces are then presented where one prefers gaps and the other creates thin-shell structures. Details for parallelizing the construction of octree are presented in section 5. Lastly, some experimental results are given in section 6.

2. LAYERED DEPTH-NORMAL IMAGES (LDNI)

A representation, named as *Layered Depth-Normal Images* (LDNIs), that implicitly encode the shape of solid models has been introduced in [1] based on the following definitions.

Definition 1 A single layered depth-normal image (*e*-LDNI) with a specified viewing direction *e* is a two-dimensional image with $w \times w$ pixels, where each pixel contains a sequence of four-components-vectors (d, n_x, n_y, n_z). At each sample, d specifies the depth from the intersection (between a ray along the viewing direction and the surface under sampling) to the viewing plane, (n_x, n_y, n_z) is the surface normal at the intersection point, and the samples are sorted in ascending order by the value of d.

Note that the samples come in pairs for a solid model – one enters the model while the other runs out of the model.

Definition 2 Letting *x*-, *y*- and *z*- specify the images sampled perpendicular to *x*-, *y*- and *z*-axis respectively, a set of structured layered depth-normal images (LDNI) consists of *x*-LDNI, *y*-LDNI and *z*-LDNI with the same resolution $w \times w$, and the images are located to let the intersections of their rays intersect at the $w \times w \times w$ nodes of uniform grids in $\Re^3$.

Assumption 1 The sampling rate $w \times w$ of each LDNI satisfies that $w=2^n+1$ with n being an integer.

In the rest of this paper, we simply conduct LDNI to denote the structured layered depth-normal images. Details about how to sample a solid model from surfaces into LDNI can be found in [1]. Figure 1 shows an example of LDNI sampled from a mesh model. We have also developed the method of using LDNI to model complex objects in [26].

3. ADAPTIVE CONTOURING

An adaptive contouring algorithm is developed in this section to generate the contour mesh surface of a LDNI solid model with the help of a hierarchical structure – octree. The contouring algorithm consists of two major steps: 1) octree construction and 2) mesh generation.

3.1. Octree Construction

Starting from the root cell (i.e., bounding box of the model), the cells are recursively refined into eight sub-cells based on the condition of 1) the topology simplicity and 2) the geometry approximation error. By Assumption 1, we can let twelve edges of each cell C overlap with a ray that samples the model into LDNI. Therefore, the intersected Hermite data points between the cell-edge and the solid model can be efficiently and easily detected on the LDNI representation.

Definition 3 For a cell edge $E \subset C$ starting from the depth value s and ending at the depth value t overlapped with a ray, the sample (d, n_x, n_y, n_z) on this ray is classified to intersect with this cell edge E if $d \in [s,t)$.

Note that d, s and t are converted into integer according to the finest precision (as [4]) to robustly classify the samples of LDNI. This definition avoids double counting of a sample in different cells when the sample exactly overlaps the endpoint of a cell edge. Similarly, the intersection between a cell face and a LDNI solid model can be easily detected by the samples on the ray perpendicular to the cell face. All above detections and the detection of whether a Hermite data point is inside a cell can be determined in a constant time complexity.

Definition 4 For all Hermite samples in a cell C, a point v_c, the position of which minimizes the quadric error function (QEF) [3], is defined as the *error-minimizing* point of this cell.

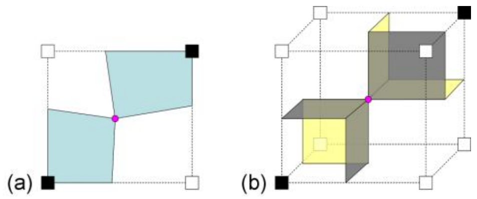

Figure 2: Illustration for (a) the face ambiguous topology and (b) the voxel ambiguous topology that leads to nonmanifold edges and vertices.

The method for computing the position of v_c can be found in [27]. We initially place v_c at the average position v_{avg} of all Hermite samples in C and then compute the optimal position v_{opt} by the singular value decomposition (SVD). If the distance between v_{avg} and v_{opt} is greater than ε_s , we truncate the movement of v_c from v_{avg} to v_{opt} by the length ε_s . In our implementation, we choose the value of ε_s as twice of a pixel's width in LDNI.

The refinement of cells is stopped based on two criteria.

Criterion 1 The surface of a LDNI solid included in a cell C can be guaranteed to have a disk-like topology.

Criterion 2 The distance between v_c and the planes defined by all Hermite samples in a cell C is not greater than a user defined tolerance ε_g .

If either of the criteria is not satisfied, the cell C is refined into eight sub-cells until it reaches the finest resolution of the given LDNI. Criterion 2 ensures that the geometry error between the contour surface and the LDNI solid model is controlled, whereas Criterion 1 prevents the generation of nonmanifold entities on the resultant meshes (details will be given in the next section).

3.2. Mesh Generation

The mesh generation on an octree follows the strategy given in [3]. On the minimal edges whose two endpoints are with different inside/outside status, the polygonal faces are constructed by connecting the minimal-error points in the cells neighboring to the minimal edges. The orientation of a face should let its normal pointing to the *outside* endpoint on the minimal edge. Note that as the inside/outside status of a cell node can be detected by *x*-, *y*- and *z*-LDNI independently, the results may be not compatible because of the truncation error in sampling. As in [1], we determine the status of a cell node by the majority voting rule. Following [3], contouring octrees requires three functions *cellProc*[c], *faceProc*[c_1,c_2] and *edgeProc*[c_1,c_2,c_3,c_4]. The mesh generation can be completed by calling the *cellProc* function with the root cell. The function *cellProc* conducts eight calls to *cellProc*, twelve calls to *faceProc* and six calls to *edgeProc* with the eight sub-cells of c. The function *faceProc* gives four calls to *faceProc* and four calls to *edgeProc*. Lastly, *edgeProc* calls *edgeProc* twice if it has not reached a minimal edge. The recursive calls to *edgeProc* function terminate when all of the input cells are leaf cells.

4. MANIFOLD-PRESERVED MESH CONSTRUCTION

To ensure the manifold topology on resultant meshes, the refinement criteria and the in-cell mesh generation strategies are investigated in this section.

4.1. Refinement Criteria for Manifold Preservation

It is not difficult to find that our algorithm presented in above section is a variation of the dual contouring in [3]. Such a contouring algorithm will generate nonmanifold vertices and edges for all of the ambiguous sign configurations in the original MC algorithm [1]. This is because each cell only contains one minimal-error point, therefore it will only generate one vertex on the resultant mesh surface. Such configurations can be detected by the face ambiguous configuration and the voxel ambiguous configuration [7, 28].

Definition 5 For the face of a cell, when the signs at nodes alternate during anticlockwise (or clockwise) traversal, the configuration of this cell is in the *face ambiguous*.

Definition 6 For a cell, when any pair of diagonally opposite nodes has one sign while the other vertices have a different sign, the configuration of this cell is in the *voxel ambiguous*.

Figure 2 illustrates the configurations for face ambiguous and voxel ambiguous. Thus, the following criterion is employed to check whether the topology inside a cell C is simple.

Criterion 1(a) The inside/outside configuration of nodes of a given cell C leads to neither face nor voxel ambiguity.

Only checking the topology ambiguity of a cell by the configuration of cell nodes is not enough to guarantee that the topology inside a cell is disk-like. Therefore, the following three supplementary criteria are used together with Criterion 1(a) to ensure the topology inside a cell simple.

Criterion 1(b) Each edge of a given cell C has either one or none intersection with the LDNI solid.

Here, the number of intersections is evaluated according to Definition 3.

Criterion 1(c) All empty (or solid) faces of a cell C have no intersection with the LDNI solid.

Criterion 1(d) An empty (or solid) cell C contains no sample from the LDNI solid.

In short, if a cell C satisfies all the criteria 1(a)-(d), C contains simple topology surface so that does not need to be refined. Otherwise, C is subdivided into eight sub-cells.

4.2. Manifold-Preserved Mesh Generation in Cell

The above refinement criteria will keep refining the cells in order to let the topology of contour surface inside a cell be simple. However, as the resolution of a solid model represented

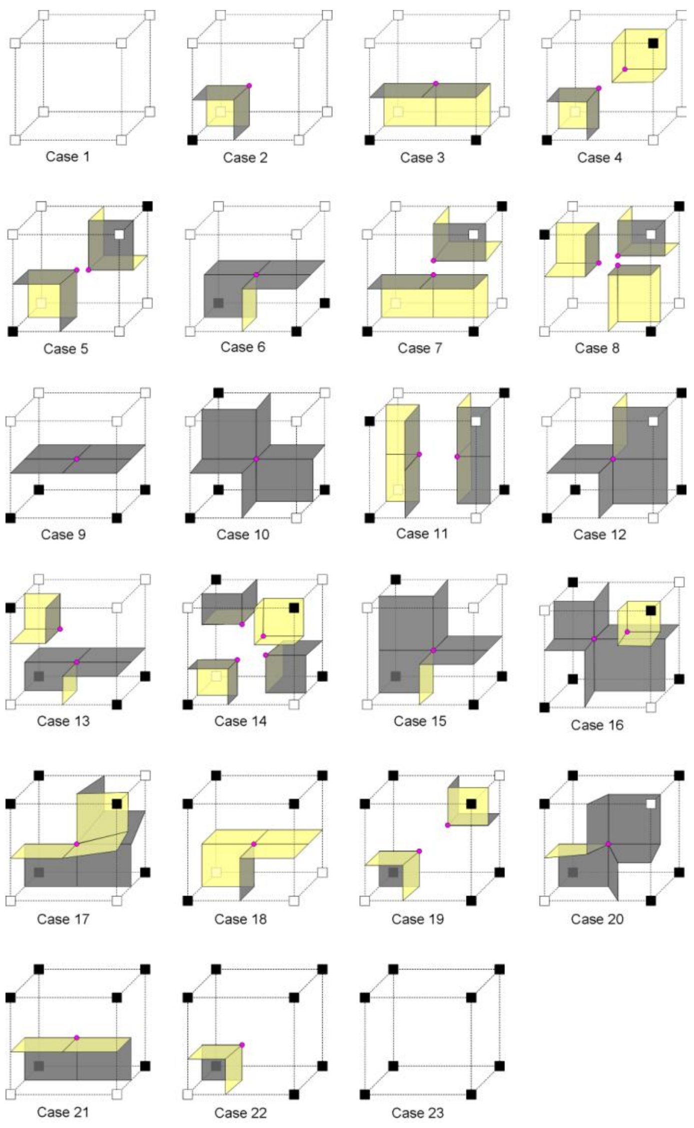

Figure 3: All configurations for constructing gap preferred mesh surfaces inside a cell.

by LDNI is limited by the sampling rate, it is not surprising that some cells have not satisfied the criteria 1(a)-(d) even when the refinement reaches the finest resolution of LDNI. We process such cells by the followings:

- For the edges with multiple intersection points, if both of their endpoints are inside (or outside), the intersection points are simply neglected.
- For the edges with multiple intersection points, if their endpoints are with different inside/outside status, the intersection samples whose normal vectors are compatible to the direction of node configuration are searched and the one nearest to the middle point of edge is selected as the only intersection sample on this edge.
- The samples that only intersect the face or volume of the cell are neglected.
- Based on the node configurations, multiple vertices are inserted into the cells and are associated with different cell edges.

Among these processes, the most difficult one is how to insert multiple vertices and associate them with different edges. The following definitions and rules are exploited for this purpose.

Definition 7 For a cell edge, if one of its endpoints is inside whereas the other is outside, the edge is named as *intersect-edge*.

Definition 8 For a cell edge, if both of its nodes are inside, the edge is named as *solid-edge*; whereas both of its nodes are outside, the edge is named as *empty-edge*.

A color flooding algorithm is developed to cluster the nodes of a cell based on their configurations. All outside nodes linked by empty-edges are grouped into the same cluster, and all inside nodes linked by solid-edges are also grouped together.

Definition 9 The set of inside node clusters is defined as ϕ_{in}, and the set of outside node clusters is denoted by ϕ_{out}. $|\phi|$ defines the number of clusters in a set ϕ.

Definition 10 If $|\phi_{in}|=2$, d_{in} defines the minimal distance between the clusters of inside nodes; d_{out} is the minimal distance between the clusters of outside nodes when $|\phi_{out}|=2$. Here, the distance is measured along cell edges.

Two strategies are developed to process topology ambiguity in a leaf cell: 1) gap preferred and 2) thin-shell preferred.

Table 1: Parameters for Different Configurations in the Gap Preferred Mesh Construction

Configuration	Value of Parameters
Case 1	$\|\phi_{in}\|$=0, $\|\phi_{out}\|$=1
Case 2,3,6,9,10,12,15,18,21,22	$\|\phi_{in}\|$=1, $\|\phi_{out}\|$=1
Case 4,5,7	$\|\phi_{in}\|$=2, $\|\phi_{out}\|$=1
Case 8	$\|\phi_{in}\|$=3, $\|\phi_{out}\|$=2, d_{out}=2
Case 11,13	$\|\phi_{in}\|$=2, $\|\phi_{out}\|$=2, d_{out}=2
Case 14	$\|\phi_{in}\|$=4, $\|\phi_{out}\|$=4
Case 16	$\|\phi_{in}\|$=2, $\|\phi_{out}\|$=3
Case 17,20	$\|\phi_{in}\|$=1, $\|\phi_{out}\|$=2, d_{out}=2
Case 19	$\|\phi_{in}\|$=1, $\|\phi_{out}\|$=2, d_{out}=3
Case 23	$\|\phi_{in}\|$=1, $\|\phi_{out}\|$=0

All possible configurations for constructing gap preferred surfaces inside a cell are listed in Figure 3, and their corresponding values of $|\phi_{in}|$, $|\phi_{out}|$ and d_{out} are listed in

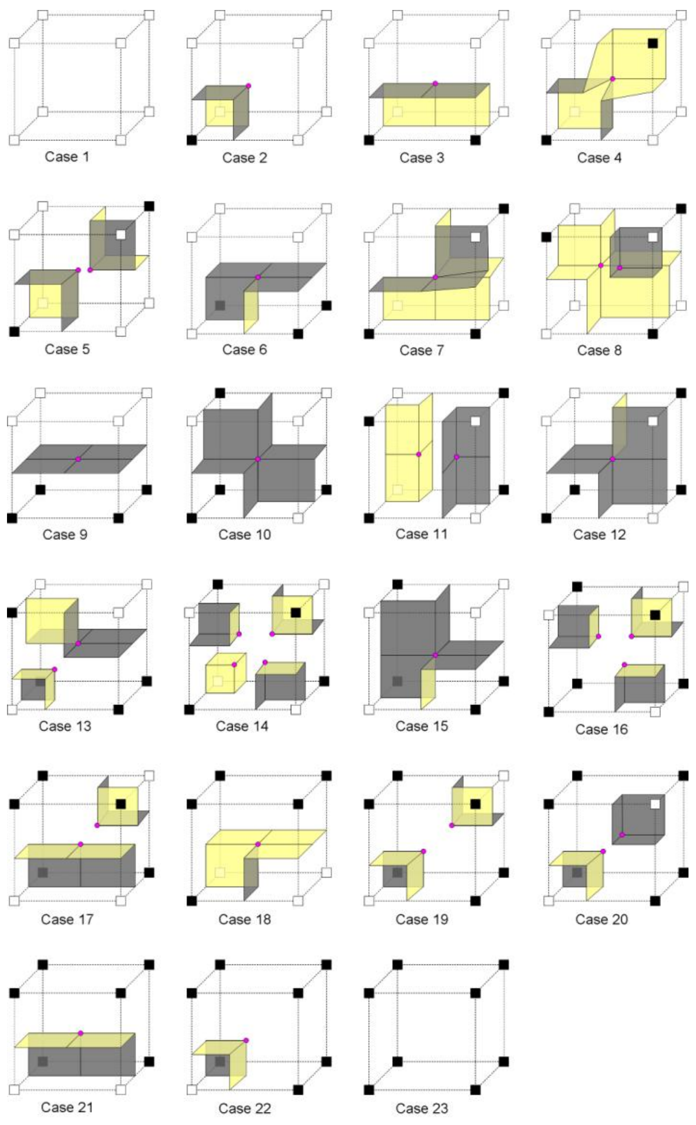

Figure 4: All configurations for constructing thin-shell preferred mesh surfaces inside a cell.

Table 1. The mesh surfaces are constructed by the following rules.

Rule 1 Vertices are only constructed in the cells with neither $|\phi_{in}|=0$ nor $|\phi_{out}|=0$.

Rule 2a For those cubes with $|\phi_{out}|=2$ and $d_{out}=3$, two vertices are created in the cell and each of the vertices is associated with the intersect-edge linking to a cluster of *outside* nodes.

Rule 3a For those cubes either $|\phi_{out}|\neq 2$ or $d_{out}\neq 3$, $|\phi_{in}|$ vertices are created in the cell and each of the vertices is associated with the intersect-edge linking to a cluster of *inside* nodes.

Rule 4 Polygonal faces are created on every intersect-edge by linking vertices in its neighboring cells associated with this edge, and the orientation of faces is pointing to the *outside* node on the edge.

Note that the quadrilateral faces will be constructed when all neighboring cells to an intersect-edge are in the same level in octree, and triangular faces will be generated when they belong to different levels. The gap preferred mesh construction inside a cell by above rules has exactly the same result as what Nielson proposed in [19].

All possible configurations for constructing thin-shell preferred surfaces inside a cell are listed in Figure 4, and their corresponding values of $|\phi_{in}|$, $|\phi_{out}|$ and d_{in} are listed in Table 2. Then, the mesh surfaces are constructed by the same rule 1 and 4 but different rules 2 and 3. Rule 2b and 3b here are in fact an inversed version of rule 2a and 3a above.

Rule 2b For those cubes with $|\phi_{in}|=2$ and $d_{in}=3$, two vertices are created in the cell and each of the vertices is associated with the intersect-edge linking to a cluster of *inside* nodes.

Rule 3b For those cubes either $|\phi_{in}|\neq 2$ or $d_{in}\neq 3$, $|\phi_{out}|$ vertices are created in the cell and each of the vertices is associated with the intersect-edge linking to a cluster of *outside* nodes.

Table 2: Parameters for Different Configurations in the Thin-Shell Preferred Mesh Construction

Configuration	Value of Parameters
Case 1	$|\phi_{in}|=0,\ |\phi_{out}|=1$
Case 2,3,6,9,10,12,15,18,21,22	$|\phi_{in}|=1,\ |\phi_{out}|=1$
Case 4,7	$|\phi_{in}|=2,\ |\phi_{out}|=1,\ d_{in}=2$
Case 5	$|\phi_{in}|=2,\ |\phi_{out}|=1,\ d_{in}=3$
Case 8	$|\phi_{in}|=3,\ |\phi_{out}|=2$
Case 11,13	$|\phi_{in}|=2,\ |\phi_{out}|=2,\ d_{in}=2$
Case 14	$|\phi_{in}|=4,\ |\phi_{out}|=4$
Case 16	$|\phi_{in}|=2,\ |\phi_{out}|=3,\ d_{in}=2$
Case 17,19,20	$|\phi_{in}|=1,\ |\phi_{out}|=2$
Case 23	$|\phi_{in}|=1,\ |\phi_{out}|=0$

The mesh surface produced by the thin-shell preferred (interior connectivity optimizing) strategy has fewer gaps than the result from the gap preferred method. An example has been given in Figure 5 for comparing the resultant contour surfaces from the gap preferred and the thin-shell preferred contouring methods. The heuristic of choosing which strategy is left to users.

4.3. Details of Manifold Preservation

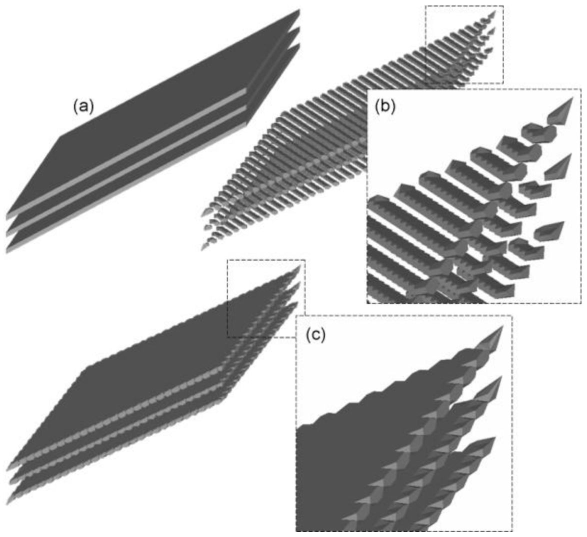

Figure 5: A contouring example from (a) the given model in the LDNI representation with 65×65 resolution, (b) the mesh generated by the gap preferred strategy, and (c) the mesh generated by the thin-shell preferred rules.

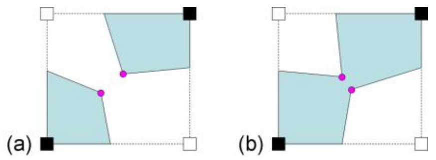

Figure 6: The consistent configuration across the face with ambiguous topology will ensure the gap-free contour surface: (a) the configuration for the gap preferred strategy and (b) the configuration for the thin-shell preferred strategy.

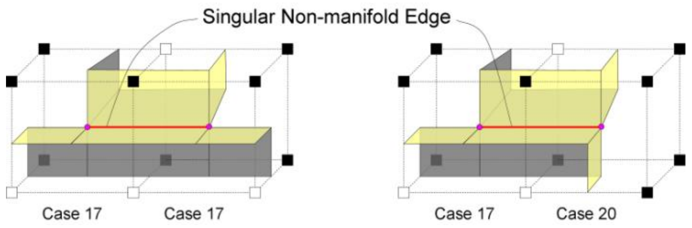

Figure 7: Example scenario that non-manifold edges could be generated by the configurations in Figure 3.

Some implementation details and special cases of our manifold-preserved mesh construction algorithm are discussed in this section.

Gap-Free When constructing the mesh surface inside a cell following the gap preferred configurations in Figure 3, the mesh across the faces with ambiguous topology is consistent as the one shown in Figure 6(a). When the mesh surface is generated according to the thin-shell preferred configurations in Figure 4, the constructed surface across topologically ambiguous faces will always be the configuration as shown in Figure 6(b). For those contoured surface across the topologically consistent faces, no gap will be generated either as there is no ambiguity. Because of such consistency, the final contour surface is gap-free.

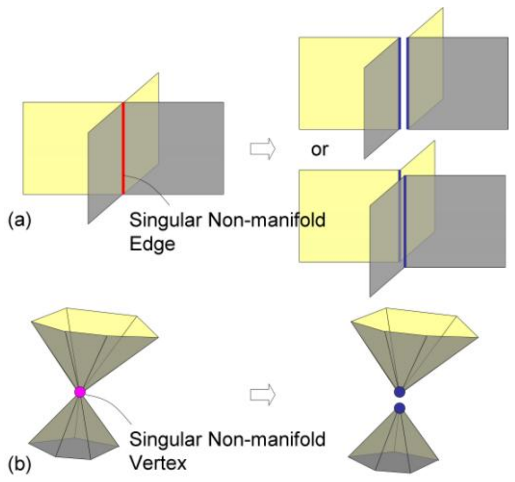

Figure 8: Correction of non-manifold entities: (a) singular non-manifold edge, and (b) singular non-manifold vertex.

Non-manifold Edge Although Nielson claimed in [19] that the configuration as in Figure 3 will not generate non-manifold entities, we still found non-manifold edges by simply following his method. The problem comes from the case 17 and 20 of the gap preferred configurations. As shown in Figure 7, when two neighboring cells are in case 17 or 20, a non-manifold edge which is shared by four faces will be created. Similar scenario occurs for thin-shell preferred configurations when neighboring cells are in case 4 or 7. Such problem can be well solved by some clever implementation. Our implementation adopts the data structure in [29], where an edge shared by two faces is assigned as positive direction in its left face and negative direction in its right face. Also, every vertex contains a list of adjacent edges. For the example as shown in Figure 7, four faces around the non-manifold edge are constructed one by one. When constructing an edge *e* pointing from v_s to v_e, we search the adjacent edge lists of its two endpoints, v_s and v_e. If there is an existing edge *e** connecting v_s and v_e, we processing as follows:

- If *e** also pointing from v_s to v_e, *e** is used as the edge only when its *left* face is null; otherwise, a new edge e is constructed.
- If *e** inversely pointing from v_e to v_s, *e** is used as the edge only when its *right* face is null; otherwise, a new edge *e* is constructed.

By this way, the faces around a non-manifold edge will automatically be clustered into two pairs of normal compatible faces (i.e., see the illustration in Figure 8(a)).

Non-manifold Vertex When the topology inside every cell is really disk-like, the above algorithm will not generate any non-manifold entities. However, as the checking of criteria 1(b)-

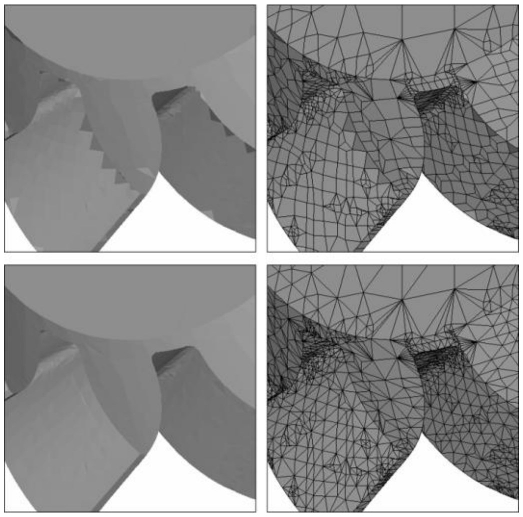

Figure 9: Splitting quadrilateral faces (top row) into triangles (bottom row) to improve the visualization result.

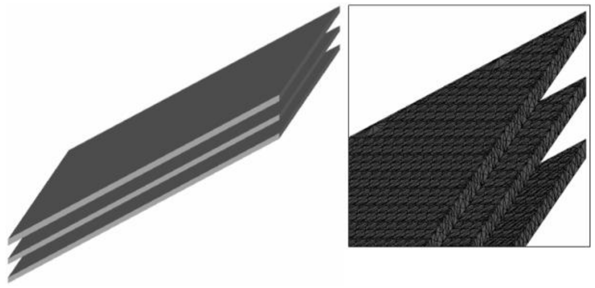

Figure 10: Contouring result of the thin-shell model from a LDNI solid model in 129×129 resolution.

(d) is based on numerical computation, the truncation error may lead to the misclassification of cells. Therefore, although it seldom happens, non-manifold vertices as shown in Figure 8(b) may be generated in some extreme cases. For these non-manifold vertices, we add a post-processing step to correct the topology around them. Firstly, the faces adjacent to every vertex are clustered into groups where faces are linked by manifold edges. If there are more than one group of faces around a vertex *v*, *v* is a non-manifold vertex. To correct the topology around *v* to be manifold, the faces in different groups are separated by duplicating new vertices (as illustrated in Figure 8(b)). While creating faces during mesh contouring, the Hermite sample corresponding to every face is also stored together with the constructed face. Lastly, the duplicated vertices are repositioned by the Hermite samples in the faces adjacent to them.

By these steps, both the gap preferred and the thin-shell preferred methods can generate manifold-preserved contour surface for solid models in LDNI representation. However, as the vertices in quadrilateral faces may not be coplanar, some aliasing error or incorrect normal vector is generated along sharp edges (e.g., the top row of Figure 9). The quadrilateral faces must be split into triangular ones.

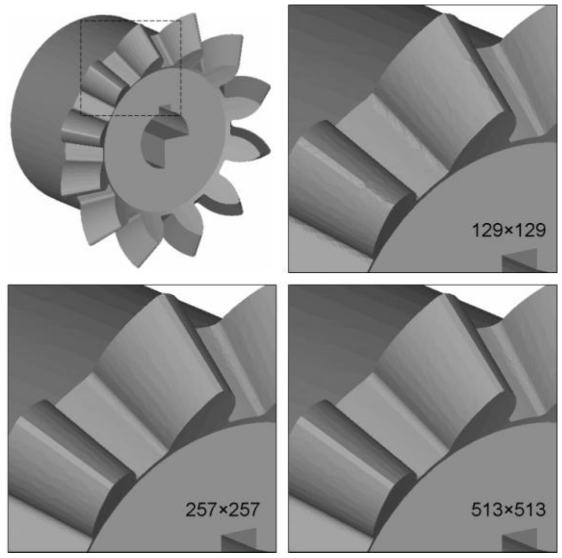

Figure 11: Contouring results of the gear model from LDNI with different resolutions.

Splitting Quadrilateral Face There are two ways to split a quadrilateral face f_q into two triangular ones along the diagonal. We choose the way which will let the new triangles' normal more consistent to the normal vectors at faces adjacent to the four edges of f_q (i.e., the sum of normal variation is minimized). If two ways of splitting have the same normal variation, f_q is split along the shorter diagonal. By this way, the aliasing along sharp edges can be recovered. See the example shown in the bottom row of Figure 9.

5. PARALLELIZATION OF OCTREE CONSTRUCTION

By performing a set of test operations, we found that the most time-consuming step in the contouring algorithm is octree construction, which takes about 70%-95% of the whole computing time. Nowadays, more and more dual-core, quad-core and even multiple processors are available on commercial PCs. Therefore, we parallelize the program of octree construction to speed up the computation. More specifically, we simply subdivide the root cell into eight sub-cells, construct the children cells of each sub-cell in different threads, and then put together the sub-trees generated in different threads. As the children cells' construction of each sub-cell is not dependent on other sub-cells, we do not even need to consider the mutex problem among threads in our parallelization. When running such a program on a computer framework of multiple-processors and shared memory, the computation in different threads will be performed in parallel on different processors. The program of mesh generation actually can also be parallelized in a similar way. However, as it does not take too much time compared to the octree construction step, running it in parallel will spend some additional time on the coordination between different threads which may even reduce the efficiency. Our tests also prove this. Thus, we parallelize the octree

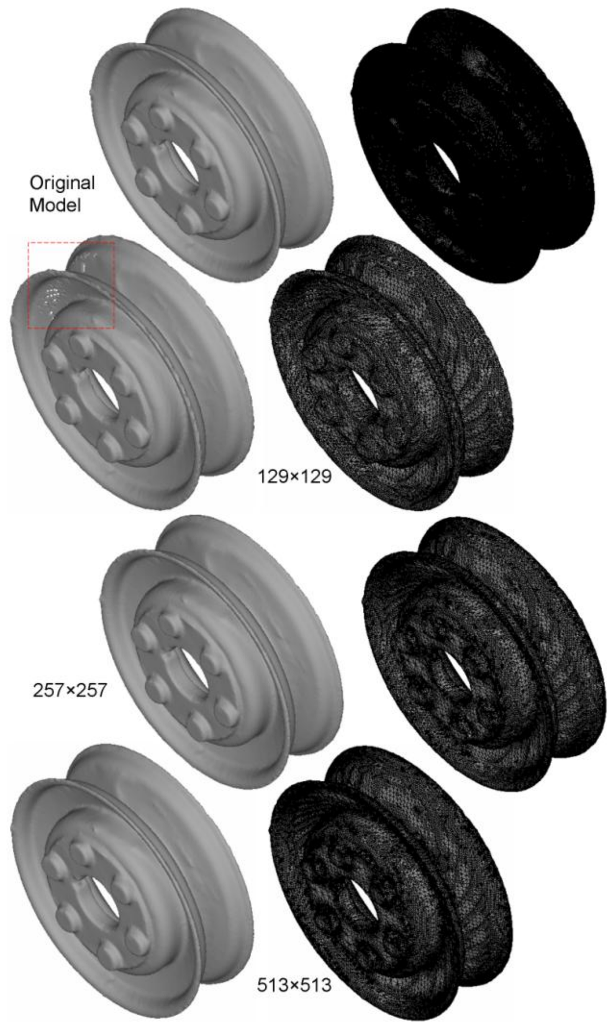

Figure 12: Contouring results of the pulley model from LDNI with different resolutions – unwanted holes are generated in the 129×129 resolution.

construction step only. In the current implementation, the program for constructing octree from LDNI solid model is paralleled with about 10 lines of C++ code using the POSIX thread library [30]. Experimental results will be shown in the next section.

6. RESULTS

We have implemented the proposed algorithm in a program in C++ plus OpenGL. The tests on adaptive contouring are conducted on various example models in LDNI representation at different resolutions. The first test is given for the thin-shell structure that has been shown in Figure 5. When increasing the resolution of LDNI from 65×65 into 129×129, the thin-shell structure can be successfully reconstructed (as in Figure 10). The second test is given for the gear model as previously shown in Figure 1 but in different resolutions at: 129×129, 257×257 and 513×513 (see Figure 11). The surface errors between contouring results and the original model in terms of $E_{\max}$ and E_{mean} are evaluated by the publicly available Metro tool [31]. Similar tests have also been given on three other models, pulley, vase-lion and filigree, which are with more complex geometries. The results are shown in Figures 12-14, and the statistics are listed in Table 3. It is not difficult to find that all models (even the ones with very complex geometries) can be successfully reconstructed by our contouring algorithm from the LDNI representation.

Table 3: Computational Statistics

Model	**Res.**	**Face #**	$E_{\max}$ (%)	E_{mean} (%)
Thin-shell	129×129	116,354	5.10×10^{-3}	5.95×10^{-5}
Gear	129×129	32,988	0.259	4.76×10^{-3}
	257×257	48.572	5.44×10^{-2}	3.10×10^{-3}
	513×513	58,868	8.21×10^{-2}	2.51×10^{-3}
Pulley	129×129	92,250	0.722	9.57×10^{-3}
	257×257	153,860	0.178	3.56×10^{-3}
	513×513	192,814	0.174	2.58×10^{-3}
Vase-lion	129×129	71,570	0.971	2.22×10^{-2}
	257×257	200,272	1.21	7.65×10^{-3}
	513×513	357,966	0.577	4.11×10^{-3}
Filigree	129×129	38,080	17.8	1.63×10^{-2}
	257×257	104,846	0.229	5.69×10^{-3}
	513×513	178,404	0.268	3.58×10^{-3}

* Note that the surface errors are reported with reference to the diagonal length of bounding box. We choose ε_g as 1/2000 of bounding box's width to be the approximation tolerance.

Table 4: Computing Time in Second

Model	**Single-core HyperThread CPU**			
	Octree Construction		*Mesh Generation*	*LDNI Samples**
	Sequential	Parallel		
Gear	20.8 s	9.69 s	0.422 s	884 k
Pulley	16.2 s	8.48 s	1.09 s	1,027 k
Vase-lion	16.9 s	11.9 s	2.11 s	734 k
Filigree	6.52 s	4.79 s	1.08 s	437 k
Model	**Quad-core CPU**			
	Octree Construction		*Mesh Generation*	*LDNI Samples**
	Sequential	Parallel		
Gear	2.39 s	0.828 s	0.171 s	884 k
Pulley	2.97 s	1.13 s	0.469 s	1,027 k
Vase-lion	4.03 s	1.47 s	0.859 s	734 k
Filigree	2.17 s	0.735 s	0.453 s	437 k
Model	**Two Quad-core CPUs**			
	Octree Construction		*Mesh Generation*	*LDNI Samples**
	Sequential	Parallel		
Gear	2.08 s	0.500 s	0.156 s	884 k
Pulley	2.56 s	0.640 s	0.421 s	1,027 k
Vase-lion	3.37 s	0.671 s	0.811 s	734 k
Filigree	1.83 s	0.343 s	0.421 s	437 k

* Note that all the models are computed on LDNI in 513×513.

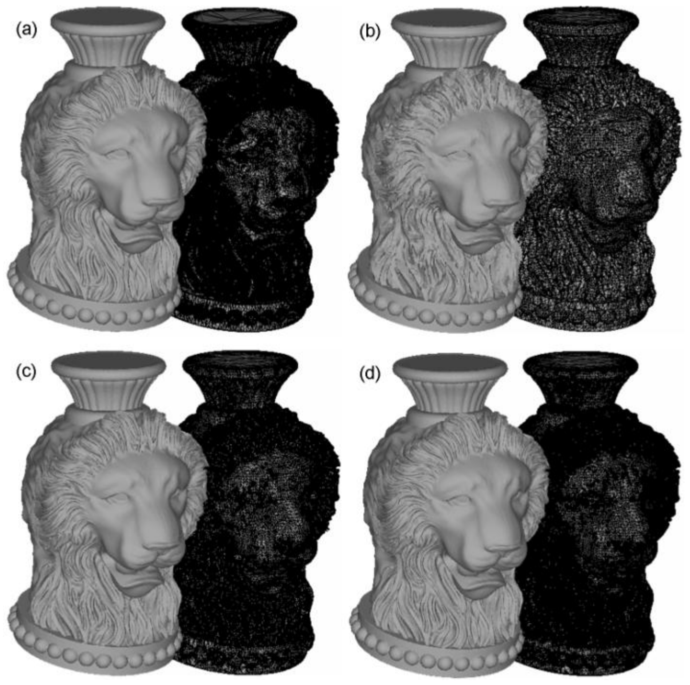

Figure 13: Contouring results of the vase-lion model from LDNI with different resolutions: (a) original model, (b) 129×129, (c) 257×257, and (d) 513×513.

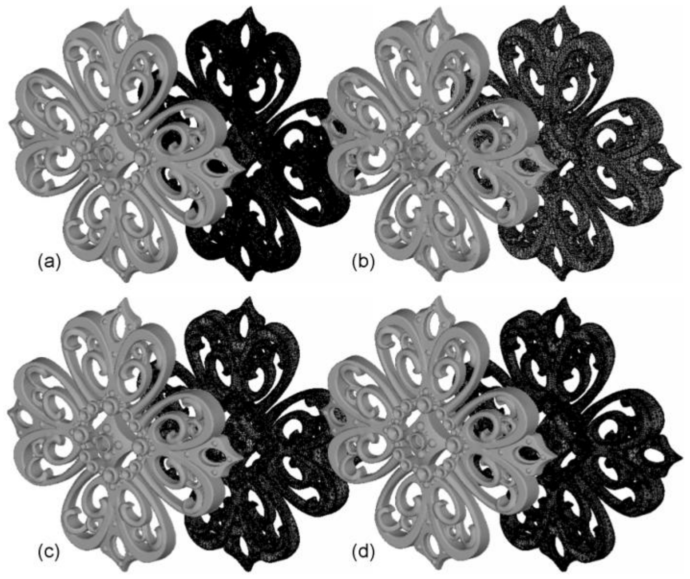

Figure 14: Contouring results of the filigree model from LDNI with different resolutions: (a) original model, (b) 129×129, (c) 257×257, and (d) 513×513.

The speed of computation has also been evaluated on three computers: the first one is with Intel Pentium IV 3.0GHz CPU (with HyperThread technology) + 2GB RAM, the second PC is with Intel Core2 Quad CPU Q6600 2.4GHz + 2GB RAM, and the third one is with two Intel Xeon Quad CPU E5440 2.83GHz + 8GB RAM. Table 4 presents computing times with and without parallelization of the octree construction. With the parallel programming of octree construction, the computing time has been shortened from 27% to 54% even on the single-core CPU with Hyper-Thread technology.

7. CONCLUSION AND DISCUSSION

This paper focuses on developing a manifold-preserved adaptive contouring algorithm to generate mesh surface for a given solid model in Layered Depth-Normal Images (LDNI) representation. The whole algorithm consists of two major steps: the octree construction and the mesh generation. The generated mesh surfaces are adaptive to shape error and topology ambiguity, and the two-manifold topology is preserved. Furthermore, the octree construction step can be easily parallelized on a computer with multiple processors and shared memory. Success of the proposed algorithm has been demonstrated from experimental tests.

The major drawback of our algorithm is that the features whose dimension is less than the size of a smallest cell cannot always be recovered correctly or sometime they are even totally neglected (e.g., the wanted holes in the second row of Figure 12). One of our future works is to develop a new mesh generation strategy inside a cell. The method presented in [6] will be considered and improved, thus can guarantee two-manifold topology on the resultant mesh surface. Another possible future work is to further extend this algorithm into one that can run on the PC cluster with distributed memory, which is very important for processing extremely large data set.

ACKNOWLEDGMENTS

The first author would like to acknowledge the support by CUHK Direct Research Grant CUHK/2050400 and the AIM@SHAPE Shape Repository for sharing some models used in this paper. The second author would like to acknowledge the support by the James H Zumberge Faculty Research and Innovation Fund at the USC.

REFERENCES

[1] Wang C.C.L. and Chen Y., 2008, “Layered Depth-Normal Images: a sparse implicit representation of solid models”, Computer-Aided Design, submitted.
[2] Heidelberger B., Teschner M., and Gross M., 2003, “Volumetric collision detection for deformable objects”, Technical Report No.395, Computer Science Department, ETH Zurich.
[3] Ju T., Losasso F., Schaefer S., and Warren J., 2002, “Dual contouring of Hermite data”, ACM Transactions on Graphics, vol.21, no.3, pp.339-346.
[4] Chen Y., 2007, “An accurate sampling-based method for approximating geometry”, Computer-Aided Design, vol.39, no.11, pp.975-986.
[5] Schaefer S. and Warren J., 2004, “Dual marching cubes: primal contouring of dual grids”, In Proc. of Pacific Graphics 2004, pp.70-76.

[6] Varadhan G., Krishnan S., Kim Y.J., and Manocha D., 2003, "Feature-sensitive subdivision and isosurface reconstruction", In Proc. of IEEE Visualization 2003, pp.99-106.
[7] Varadhan G., Krishnan S., Sriram T.V.N., and Manocha D., 2004, "Topology preserving surface extraction using adaptive subdivision", In Proc. of Eurographics Symposium on Geometry Processing 2004, pp.235-244.
[8] Schaefer S., Ju T., and Warren J., 2007, "Manifold dual contouring", IEEE Transactions on Visualization and Computer Graphics, vol.13, no.3.
[9] Rossignac J.R. and Requicha A.A.G., 1999, "Solid modeling", Encyclopedia of Electrical and Electronics Engineering, John Wiley and Sons.
[10] Hoffmann C.M., 2001, "Robustness in geometric computations", ASME Journal of Computing and Information Science in Engineering, vol.1, pp.143-156.
[11] Kim Y.J., Varadhan G., Lin M.C., and Manocha D., 2003, "Fast swept volume approximation of complex polyhedral models", Proceedings of the 8th ACM Symposium on Solid Modeling and Applications, pp.11-22.
[12] Chen Y., Wang H., Rosen D., and Rossignac J.R., 2007, "A point-based offsetting method of polygonal meshes", ASME Journal of Computing and Information Science in Engineering, submitted.
[13] Tiede U., Shiemann T., and Hoehne K., 1998, "High quality rendering of attributed volume data", In Proc. of IEEE Visualization'98, pp.255-262.
[14] Jones M.W., Baerentzen J.A. and Sramek M., 2006, "3D distance fields: a survey of techniques and applications", IEEE Transactions on Visualization and Computer Graphics, vol.12, no.4, pp.581-599.
[15] Kobbelt L.P., Botsch M., Schwanecke U., and Seidel H.-P., 2001, "Feature sensitive surface extraction from volume data", In Proc. of SIGGRAPH 2001, pp.57-66.
[16] Lorensen W. and Cline H., 1987, "Marching cubes: a high resolution 3D surface construction algorithm", Computer Graphics, vol.21, pp.163-169.
[17] Nielson G.M. and Hamann B., 1991, "The asymptotic decider: resolving the ambiguity in marching cubes", Proceedings of IEEE Visualization'91, pp.83-91.
[18] Lewiner T., Lopes H., Vieira A.W., and Tavares G., 2003, "Efficient implementation of marching cubes' cases with topological guarantees", Journal of Graphics Tools, vol.8, pp.1-15.
[19] Nielson G.M., 2004, "Dual marching cubes", In Proc. of IEEE Visualization 2004, pp.489-496.
[20] Andujar C., Brunet P., Chica A., Navazo I., Rossignac J., and Vinacua A., 2004, "Optimizing the topological and combinatorial complexity of isosurfaces," Computer-Aided Design, vol.37, no.8, pp.847-857.
[21] Ellis J.L., Kedem G., Lyerly T.C., Thielman D.G., Marisa R.J., Menon J.P., and Voelcker H.B., 1991, "The ray casting engine and ray representatives", In Proc. of ACM Symposium on Solid Modeling and Applications 1991, pp.255-267.
[22] Menon J.P. and Voelcker H.B., 1995, "On the completeness and conversion of ray representations of arbitrary solids", In Proc. of ACM Symposium on Solid Modeling and Applications 1995, pp.175-286.
[23] Hartquist E.E., Menon J.P., Suresh K., Voelcker H.B., Zagajac J., 1999, "A computing strategy for applications involving offsets, sweeps, and Minkowski operations", Computer-Aided Design, vol.31, no.3, pp.175-183.
[24] Zhang W., Peng X., Leu M.C., and Zhang W., 2007, "A novel contour generation algorithm for surface reconstruction from dexel data", ASME Journal of Computing and Information Science in Engineering, vol.7, no.3, pp.203-210.
[25] Shade J., Gortler S., He L.-W., and Szeliski R., 1998, "Layered depth images", In Proc. of SIGGRAPH 98, pp.231-242.
[26] Chen Y. and Wang C.C.L., "Layered Depth-Normal Images for complex geometries – part one: accurate sampling and adaptive modeling", In Proc. of ASME IDETC/CIE 2008 Conference, 28th Computers and Information in Engineering Conference, New York City, New York, August 3-6, 2008.
[27] Schaefer S. and Warren J., 2002, "Dual contouring: 'the secrete sauce'", Technical Report.
[28] Wilhelms J. and van Gelder A., 1990, "Topological considerations in isosurface generation extended abstract", In Proc. of the 1990 Workshop on Volume Visualization, pp.79-86.
[29] Wang C.C.L., Wang Y., and Yuen M.M.F., "Feature-based 3D non-manifold freeform object construction", Engineering with Computers, vol.19, no.2-3, pp.174-190, 2003.
[30] POSIX Threads for Win32 (Open Source), http://sourceware.org/pthreads-win32/.
[31] Cignoni P., Rocchini C., and Scopigno R., 1998, "Metro: measuring error on simplified surfaces", Computer Graphics Forum, vol.17, no.2, pp.167-74.