Journal of Computational Design and Engineering, Vol. 1, No. 2 (2014) 88~95
www.jcde.org

Quadrilateral mesh fitting that preserves sharp features based on multi-normals for Laplacian energy

Yusuke Imai*, Hiroyuki Hiraoka and Hiroshi Kawaharada
Department of Precision Mechanics, Chuo University, 1-13-27 Kasuga, Bunkyo-ku, Tokyo 112-8551, Japan

(Manuscript Received September 16, 2013; Revised October 24, 2013; Accepted November 1, 2013)

Abstract

Because the cost of performance testing using actual products is expensive, manufacturers use lower-cost computer-aided design simulations for this function. In this paper, we propose using hexahedral meshes, which are more accurate than tetrahedral meshes, for finite element analysis. We propose automatic hexahedral mesh generation with sharp features to precisely represent the corresponding features of a target shape. Our hexahedral mesh is generated using a voxel-based algorithm. In our previous works, we fit the surface of the voxels to the target surface using Laplacian energy minimization. We used normal vectors in the fitting to preserve sharp features. However, this method could not represent concave sharp features precisely. In this proposal, we improve our previous Laplacian energy minimization by adding a term that depends on multi-normal vectors instead of using normal vectors. Furthermore, we accentuate a convex/concave surface subset to represent concave sharp features.

Keywords: CAD model; Hexahedral mesh; Sharp feature; Fitting algorithm; Multi-normalvectors

1. Introduction

In manufacturing, the cost of computer simulations is lower than testing actual prototypes. Thus, most manufacturers run simulations, which require volume meshes. Up until a decade ago, the simulation process started from surface meshes that were made using computer aided design (CAD) software. However, there were differences in the shape between the actual products and the CAD model, a result of manufacturing factors such as springback in production presses. Even if manufacturers used metal dies with the same shapes as in the CAD models, the parts obtained from the processes such as press working do not have the same shape as the CAD model. Thus, the actual products are different from the CAD model. For realistic simulations, the CAD model must be identical to the products. Today, the simulation process starts from point clouds scanned from the actual products. This process is called reverse engineering.

Tetrahedral/hexahedral meshes generated from such point clouds directly affect the results in finite element method (FEM) analysis. Hexahedral meshes are important because they are superior to tetrahedral meshes (see Figure 1(a)) in terms of accurate analysis. Thus, in this paper, we use hexahedral volume meshes (see Figure 1(b)) whose elements are only hexahedral cells (called all-hexahedral meshes) and consider their surface.

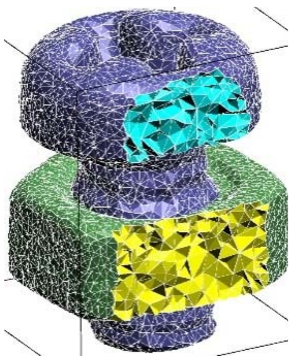
In the structural analysis between two objects, the peak stress occurs near or around the contact regions. Such regions are often sharp features. A sharp feature is typically a cusp part (such as an edge or point) of an object. Thus, to obtain accurate simulation results, the surface mesh must represent sharp features.

In this paper, we consider a voxel-based hexahedral mesh generation algorithm [1, 2]. In addition, the volume mesh for the FEM must satisfy the constraint that all Jacobians are positive. The Jacobian is a triple scalar product $a \cdot (b \times c)$, where a, b and c are vectors (edges) adjacent to the corner vertex of a cell. Our ultimate goal is automatic hexahedral mesh generation without negative Jacobians. In this paper, we discuss the quadrilateral surface of a hexahedral mesh (The algorithm we investigated in previous studies is voxel-based [1, 2]). Thus, our inputs are the target surface mesh and the quadrilateral mesh that is the surface of the voxel mesh.

Generically, we can classify the methods of boundary-fitted hexahedral meshing as voxel-based [3, 4], advancing front [5-7], whisker waving [8], cycle elimination [9], medial axis-based [10], and sweep/mapped methods [11, 12], although other types exist. Hexahedral mesh generation algorithms can be fully or semi-automatic, but there is no scheme that guarantees all Jacobians will be positive. On the other

*Corresponding author. Tel.: +81-3-3817-1821, Fax.: + 81-3-3817-1820
E-mail address: imai@lcps.mech.chuo-u.ac.jp

doi: 10.7315/JCDE.2014.009


(a)

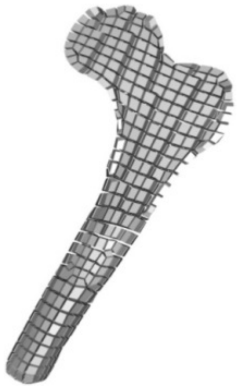

(b)

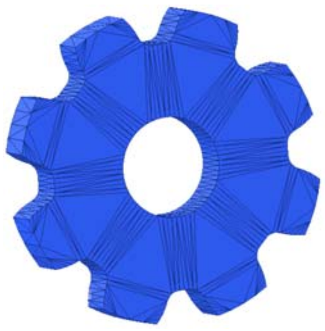
Figure 1. Tetrahedral/hexahedral meshes: (a) tetrahedral mesh, (b) hexahedral mesh.

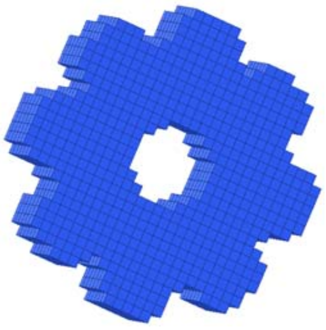
hand, there is a scheme that guarantees that positive Jacobians does exist for tetrahedral meshes. Before considering how to achieve positive Jacobians, we generate quadrilateral surfaces of the all-hexahedral meshes that represent the target surfaces. In this paper, we propose an automatic fitting algorithm with sharp features based on previous works [1, 2, 13].

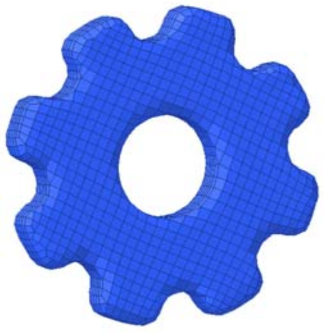
2. Previous hexahedral meshing

First, we summarize the previous hexahedral mesh generation algorithm [1, 2, 13]. The underlying algorithm can be broken down as follows.

1. Input target surface mesh (see Figure 2(a) [14]).
2. Generate voxels to wrap around the target surface using Polymender [15] (see Figure 2(b)).
3. Extract the boundary surface of voxels.
4. Fit the boundary surface of voxels (see the fitted surfaces in Figure 2(c) [1, 2] and Figure 2(d) [13]).
5. Determine the positions of the inner vertices.
6. Apply post-processing.
7. Output hexahedral mesh.


(a)


(b)


(c)

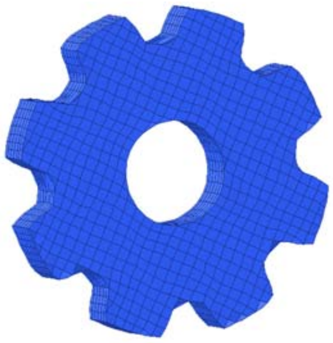

(d)

Figure 2. Gear: (a) target surface (triangle mesh), (b) voxels of gear, (c) quadrilateral mesh without sharp features, (d) with sharp features.

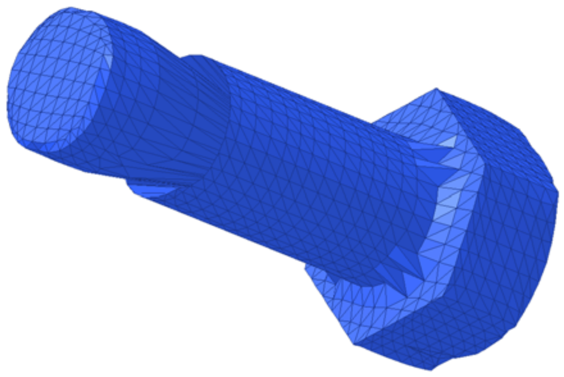


Figure 3. Bolt with reverse part.

In Step 4 above, we fit the boundary surface of voxels to the target surface mesh. We call this the fitting part. We proposed a fitting algorithm that did not represent sharp features (see Figure 2(c)) [1, 2]. Thus, we proposed a modified fitting algorithm [13]. The modified algorithm represents convex sharp features (see Figure 2(d)). However, it could not represent concave sharp features. Therefore, we are now proposing a new fitting algorithm based on multi-normal vectors and convex/concave accentuation.

3. Fitting algorithm

In this section, we explain the new fitting algorithm. Let $v_i^0, \mathrm{i} = 1,2,\cdots$ be the positions of the voxel's boundary vertices at step 0 (initial positions). The new fitting algorithm is iterative (k = 0 to T).

Algorithm 1:

1. Determine $v_i^{k+1}, \mathrm{i} = 1,2,\cdots$ by minimizing the following quadratic energy function [16], where $\boldsymbol{v}_i^k$ denotes the position of the boundary vertex $\boldsymbol{v}_i$ at step k, and w_i^{k+1} denotes the corresponding point on the input surface to $\boldsymbol{v}_i^k$.
2. Apply surface Laplacian smoothing to the (k+1)-th surface. If $k \neq$ T, then increase k by 1 and return to step 1.

Let E be the boundary edges of the boundary surface of voxels, (i,j) be an edge between $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ $|(i,j)|$ be the number of edges connecting to $\boldsymbol{v}_i$, and $\boldsymbol{nv}_i$ and nw_i^{k+1} be unit normal vectors at v_i^k on the k-th boundary surface and w_i^{k+1} on the target surface, respectively. $\|\quad\|$denotes the norm of a vector. $a \cdot b$ denotes the inner product between vectors a and b. T is the user-defined maximum iteration number ($T = 30$ in this paper). The minimizing of the quadratic energy function is as follows:

$$\min_{v_i^{k+1}} \sum_i \left\{ c_1 \left\| \left(v_i^{k+1} - \frac{1}{|(i,j)|} \sum_{(i,j)\in E} v_j^{k+1} \right) - \left(v_i^k - \frac{1}{|(i,j)|} \sum_{(i,j)\in E} v_j^k \right) \right\|^2 + c_2 \left\| v_i^{k+1} - v_i^k \right\|^2 + c_3 \left\| v_i^{k+1} - w_i^{k+1} \right\|^2 + c_4 \left\{ \sum_{(h,t)} \frac{\left(\frac{-nv_{ih}^{k+1}}{a_{ih}^{k+1}} \cdot nw_{it}^{k+1} \right)}{M_i} \right\} \right\} \tag{1}$$

Here, $c_1 = 1, c_2 = \frac{1}{2}, c_3 = 2, c_4 = 0.01.$

Surface Laplacian smoothing is used to prevent vertices from becoming congested near the sharp features of the target surface. Ordinary Laplacian smoothing, however, erases sharp features. Thus, we use feature-preserving Laplacian smoothing instead of ordinary Laplacian smoothing.

The first term in the above minimization is the difference between successive Laplacians. The second term is the difference between v_i^{k+1} and v_i^k. The third term is the difference between v_i^{k+1} and w_i^{k+1}. The fourth term is the non-matching rate between the multi-normal vectors $\mathrm{n}v_{ih}^k, h = 1,2,3\cdots$ and $\mathrm{n}w_{it}^{k+1}, t = 1,2,3\cdots$. We describe multi-normal vectors and the non-matching rate in detail in Section 4. In this term, the multi-normal vector $\mathrm{n}v_{it}^k, h = 1,2,3,\cdots$is a set of linear formulas of v_i^{k+1} (not unit vectors). In a previous study, we used a similar minimization (Eq. (2)) [13].

$$\min_{v_i^{k+1}} \sum_i \left\{ c_1 \left\| \left(v_i^{k+1} - \frac{1}{|(i,j)|} \sum_{(i,j)\in E} v_j^{k+1} \right) - \left(v_i^k - \frac{1}{|(i,j)|} \sum_{(i,j)\in E} v_j^k \right) \right\|^2 + c_2 \left\| v_i^{k+1} - v_i^k \right\|^2 + c_3 \left\| v_i^{k+1} - w_i^{k+1} \right\|^2 + c_4 \left\{ \left\| nv_i^{k+1} \right\|^2 - (nv_i^{k+1} \cdot nw_i^{k+1})^2 \right\} \right\} \tag{2}$$

The corresponding fourth term of Eq. (2) is a quadratic formula (not linear) that represents the non-matching rate between normal vectors $nv_i^{k+1}$and nw_i^{k+1}similarly. However, the value of the fourth term of Eq. (2) does not change by using $-nv_i^{k+1}$instead of nv_i^{k+1}. Therefore, let a normal vector n be a minimizer of the corresponding fourth term of Eq. (2). Then, $-n$ is also a minimizer, because the corresponding fourth term is quadratic. Figure 3 shows the resulting mesh with a reverse part. Thus, we set c_4 to be very small. In this case, the resulting mesh does not represent convex/concave sharp features (see Figure 5(c)). Figures 5(a) and 5(b) show the target surface and voxels of the bolt, respectively. Thus, we define a new fourth term of Eq. (1) as a linear formula.

In one approach [16], w_i^{k+1} is the closest point to v_i^k on the target surface mesh. In others [1, 2], however, w_i^{k+1} is the closest vertex on the target surface mesh. These two definitions of w_i^{k+1} are distinctly different. We increased the number of vertices on the target surface mesh by up-sampling [1, 2]. We chose subdividing triangles for up-sampling. As in the loop subdividing method, we subdivided each triangle on the target surface into four triangles. We applied this subdivision five times. Let S_j, $\mathrm{j} = 0,1,2,\cdots$ be vertices (including up-sampling) on the target surface. Let nS_j be the unit normal vector at S_j on the target surface.

The definition of w_i^{k+1} in this paper is:

$$w_i^{k+1} \equiv \arg\min_j . \left\{ \left\| v_i^k - s_j \right\| \left(1 - \frac{\alpha \sum_{(h,t)} \left(nv_{ih}^k \cdot ns_{jt} \right)}{N_i^k} \right) \right\} \tag{3}$$

This definition is based on a non-matching rate between nv_i^k and ns_j. In this case, $\alpha = 0.8$.

The old definition of w_i^{k+1} is [1, 2]:

$$w_i^{k+1} \equiv arg \min_j . \left\| v_i^k - s_j \right\| \tag{4}$$

Another old definition of w_i^{k+1} is [13]:

$$w_i^{k+1} \equiv arg \min_j . \left\{ \left\| v_i^k - s_j \right\| \left(1 - \alpha n v_i^k \cdot ns_j \right) \right\} \tag{5}$$

The normal vector $ns_j$of a candidate vertex s_j is closer to nv_i^k, and the vertex s_j tends to be chosen as w_i^{k+1}.

The new fitting algorithm consists of two parts. In the first part, we apply the fitting algorithm with Eq. (1). We set $c_4 = 0.0$ and $T = 2$. In the second part, we apply the algorithm with Eq. (1) with $c_4 = 0.01$, resetting K to 0, and $T = 30$ $(K = 0\ to\ 30)$.

4. Fitting algorithm that represents concave sharp features

In this section, we explain the multi-normal vector and convex/concave accentuation in the new fitting algorithm. First, we explain the non-matching rate and corresponding point w_i^{k+1} of v_i^k based on a multi-normal vector. In a previous study [13], we defined nv_i^k and nv_i^{k+1} as averages of normal vectors of triangles at step K and step K+1, respectively. Here, we realize that even if nv_i^k has the same direction as nw_i^{k+1}, the normal vector of each triangle (face) connected to v_i^k may not be parallel to the normal vector of each triangle (face) connected to w_i^{k+1}.

A face of the boundary surface of an all-hexahedral mesh is quadrilateral. We consider a quadrilateral face that has four triangles and four normal vectors. Therefore, we set the multi-normal vector of v_i^k as the normal vectors of such triangles to v_i^k. We combine several parallel vectors of this multi-normal vector into one vector and apply this operation repeatedly. As a result, this multi-normal vector has no subset of parallel normal vectors. For example, if all triangles adjacent to v_i^k are on a plane, then the multi-normal vector of v_i^k has only one element that is a normal vector of the plane.

We set nv_{ih}^k, $h = 1, 2, 3 \cdots$ as this contracted multi-normal vector. We apply this process similarly for S_j. We set the multi-normal vector $ns_{jt}, t = 1, 2, 3, \cdots$ as normal vectors of the triangles including s_j similarly. Let A_i^k, B_j be the numbers of normal vectors of nv_i^k, ns_j after the above process.

4.1 Convex/concave accentuation

In this subsection, we add a new step to Algorithm 1 of Section 3 at the second part (considering the normal vector) to represent concave sharp features. In concrete terms, before the determination of w_i^{k+1} in step 1 of Algorithm 1, we add convex/concave accentuation for the subset of the boundary surface of the all-hexahedral mesh.

To represent sharp features, v_i^{k+1} that connects sharp boundary edges corresponding to sharp features must be on the corresponding sharp edges. Therefore, it is important that the corresponding w_i^{k+1} is on the sharp edges. Therefore, before the minimization (decision of w_i^{k+1}), we accentuate the convex/concave subset of the boundary surface of the all-hexahedral mesh. Figure 4 shows the convex/concave accentuation. The black line denotes the target surface. The red line denotes the boundary surface of the all-hexahedral mesh. In this case, the red vertices are not on the convex and concave corners of the black line, and so the boundary surface does not represent sharp features.

To represent sharp features, we first calculate the Laplacian vector at v_i^k of the boundary surface. Let CC_i^k be a vector that is parallel to the average of $nv_{ih}^k, h = 1, 2, 3, \cdots, A_i^k$ whose norm is equal to the inner product between the average of the multi-normal vector and the Laplacian vector. If the inner product is negative, the direction of CC_i^k is opposite to the average of $nv_{ih}^k, h = 1, 2, 3, \cdots, A_i^k$. Let cl be the length of an edge of an initial voxel, and D be a user-defined coefficient (in this paper, we set this to 0.5).

Our convex/concave accentuation is $v_i^k {+}{=} D \times cl \times CC_i^k \times \left(\frac{1}{2}\right)^{k/3}$, where $i = 1, 2, 3 \cdots$ if K is a multiple of 3. If K is not a multiple of 3, we do not apply accentuation.

After the first part that does not depend on a normal vector, $\|v_i^T - w_i^T\|$ $i = 1, 2, 3 \cdots$ are almost 0, because the third term of $\left\|v_i^{k+1} - w_i^{k+1}\right\|$ in Eq. (1) becomes almost 0.

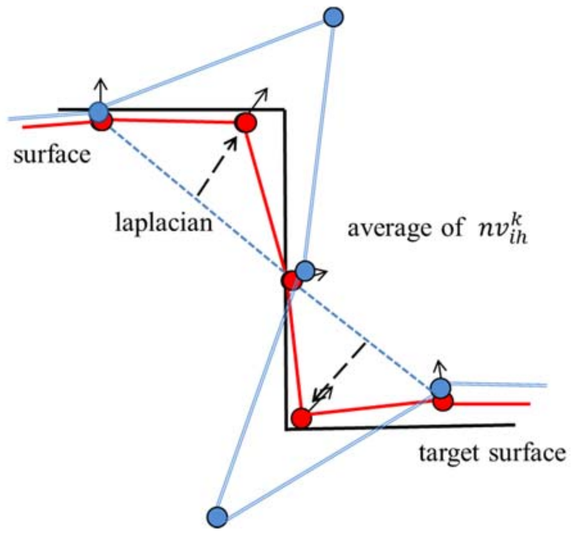


Figure 4. Convex/concave accentuation.

Thus, in the second part, without convex/concave accentuation, w_i^{k+1} does not change for all K because the inside term of minimizing on the right side of Eq. (3) is always almost 0 at the same s_j. Therefore, we use convex/concave accentuation to separate v_i^k from w_i^{k+1}. After this accentuation, the probability that w_i^{k+1} is on a corner (sharp feature) increases**.**

4.2 Matching rate between multi-normals

In this section, we explain the matching rate between multi-normal vectors $nv_{ih}^k, h = 1, 2, 3, \cdots, A_i^k$ and A_i^k, $ns_{jt}, \mathrm{t} = 1, 2, 3, \cdots, B_j$.

In order to do this, we calculate all the inner products between multi-normal vectors $nv_{ih}^k, h = 1, 2, 3, \cdots, A_i^k$ and $ns_{jt}, \mathrm{t} = 1, 2, 3, \cdots, B_j$ (See Table 1). The range of the inner product is [-1, 1], because nv_{ih}^k and ns_{jt} are unit vectors.

First, we obtain the maximums of each row and each column. We define $\sum_{(h,t)}\left(nv_{ih}^k \cdot ns_{jt}\right)$ as the sum of the maximums (bottom right cell of Table 1). In Table 1, we show the matching. The Red/Green cells are the maximum cells of the row/column. The Blue cells are the maximum cells of both the row and column. The bottom right cell shows the total of the row and column maximums.

We use this value as the matching rate between $nv_{ih}^k, h = 1, 2, 3, \cdots, A_i^k$ and $ns_{jt}, \mathrm{t} = 1, 2, 3, \cdots, B_j$. Let $N_i^k\left(= A_i^k + B_j\right)$ be the number of rows and columns in Table 1. The scaled matching rate $\sum_{(h,t)}\left(nv_{ih}^k \cdot ns_{jt}\right) / N_i^k$ is [-1, 1].

4.3 Fitting algorithm based on multi-normal vector

In this section, we explain Eq. (1) and Eq. (3) in detail. In Eq. (3), we use a scaled matching rate between multi-normal vectors nv_i^k and ns_j. The definition of w_i^{k+1} in this paper reflects the details of the target surface.

Table 1. Normal vector matching table.

	nv_{i1}^k	nv_{i2}^k	nv_{i3}^k	$\cdots$	$nv_{iA_i^k}^k$	Max
ns_{j1}	0.10	0.96	-0.02		0.38	0.96
ns_{j2}	0.75	-0.10	0.11	🟧	-0.29	0.82
$\vdots$			🟦		🟧	$\vdots$
ns_{jB_j}	0.05	0.80	0.36	🟩	0.55	0.80
Max	0.75	0.96	0.99	$\cdots$	0.75	9.64

Next, we explain the fourth term of Eq. (1). w_i^{k+1} is an element of $\{s_j\}_{j=1,2,\cdots}$. Thus, we have a multi-normal vector nw_{it}^{k+1} and a normal vector matching table for w_i^{k+1}. Let C_i^{k+1} be the number of normal vectors of nw_i^{k+1}. There is a multi-normal vector $nw_{it}^{k+1}, \mathrm{t} = 1, 2, 3, \cdots, C_i^{k+1}$.

In order to define the fourth term as linear formulas of v_i^k, we define $\sum_{(h,t)}\left(nv_{ih}^{k+1} \cdot nw_{it}^{k+1}\right)$ as the sum of the inner products between nv_{ih}^{k+1} (linear formula) and nw_{it}^{k+1} at the colored cells in the normal vector matching table for w_i^{k+1}. (At the Blue cells, we add the inner product to the sum twice).

Let a_{ih}^{k+1} be the twofold triangle area that corresponds to nv_{ih}^k, and $M_i^k\left(= A_i^k + C_i^{k+1}\right)$ be the number of rows and columns in the normal vector matching table for w_i^{k+1}. During minimizing, a_{ih}^{k+1} is constant. After the second part, we carry out post-processing to improve the quality of the resulting all-hexahedral mesh.

5. Result

In this section, we show the experimental results of our method. Figure 5(d) shows the boundary quadrilateral surface mesh of the all-hexahedral mesh (bolt model) calculated by our method. Figures 6(a), 6(b), 6(c), and 6(d) are the target surface, the boundary quadrilateral surface mesh of the bearing model using a previous method [13], voxels, and the all-hexahedral mesh calculated by our method for the bearing model, respectively. The concave sharp features of the bolt and bearing models are successfully represented. Similarly, our method can represent convex sharp features of the target surface mesh.

The smooth-feature model in Figure 7 has a convex sharp feature that diminishes gradually. Figures 7(a), 7(b), and 7(c) are the target surface, voxels, and result of our method for a smooth-feature model, respectively.

The computation times for our experiments were only a few seconds. Let $\hat{n}$ be the number of vertices of the boundary surface of voxels, and $\hat{N}$ be the number of vertices (including up-sampling) of the target surface. We solve the

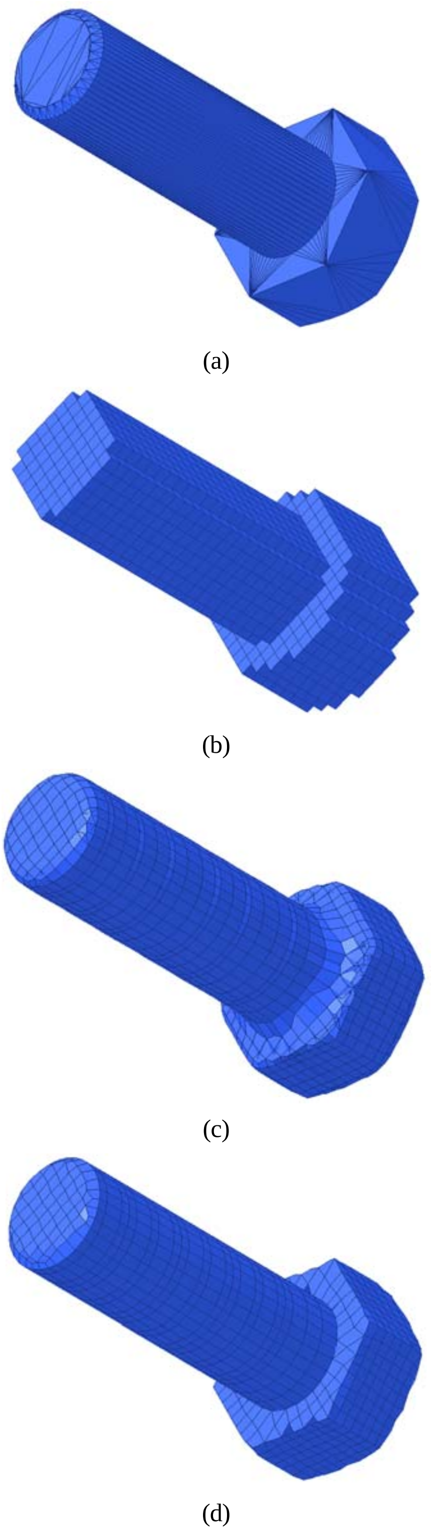


Figure 5. Bolt: (a) target surface, (b) voxels of bolt, (c) boundary quadrilateral surface of all-hexahedral mesh using a previous method [13], (d) all-Hexahedral mesh using our method.

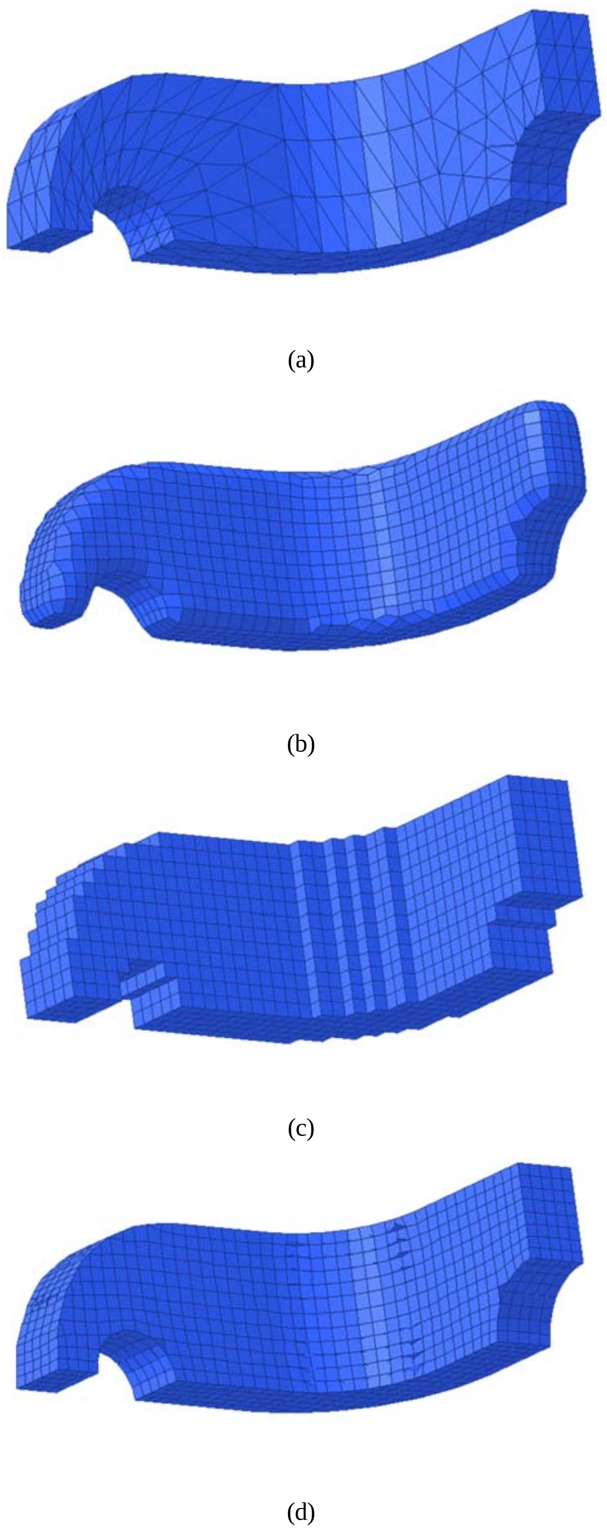
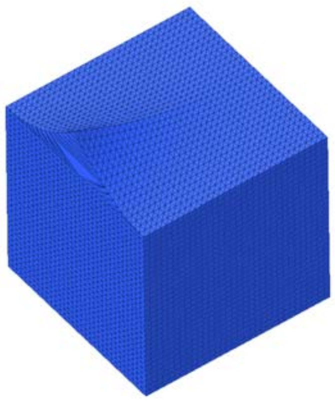
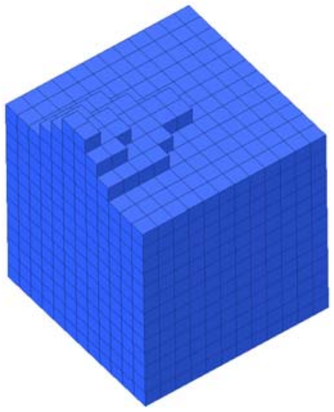


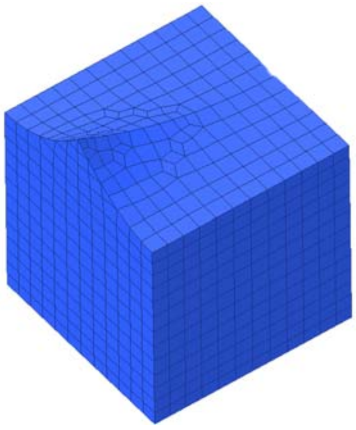
Figure 6. Bearing: (a) target surface, (b) boundary quadrilateral surface without, sharp features, (c) voxels, (d) boundary quadrilateral surface using our method



(a)



(b)



(c)

Figure 7. Smooth-feature: (a) target surface, (b) voxels, (c) result of our method.

linear system in the fitting part with a computation time of $O(\hat{n}^2)$. However, we can solve this quickly because the matrix of this linear system is sparse. The computation time of all other parts is $O(\hat{n}\hat{N})$. Thus, using octree, we decrease the actual computation time.

6. Conclusions

We have proposed a new fitting algorithm based on multi-normal vectors and convex/concave accentuation in order to represent convex/concave sharp features on a target surface. Using this algorithm for automatic hexahedral mesh generation, we obtained boundary surfaces of all-hexahedral meshes with convex/concave sharp features. In future work, we intend to determine the best coefficients for our algorithm. In addition to the surface mesh, we would like to expand this research to internal vertices.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number 24760124.

References

[1] Kawaharada H, Hiraoka H. Boundary stencils of volume subdivision for simulations. In: Proceedings of the ACDDE; 2011 Aug 27-29; Shanghai, China; p. 3-8.

[2] Kawaharada H, Sugihara K. Hexahedral mesh generation using subdivision. Computational Engineering. 2011; 16(2): 12-15.

[3] Schneiders R, Schindler R, Weiler F. Octree-based generation of hexahedral element meshes. In: The 5th International Meshing Roundtable; 1996 Oct 10-11; Pittsburgh, PA; p. 205-215.

[4] Loic M. A new approach to octree-based hexahedral meshing. In: The 10th International Meshing Roundtable; 2001 Oct 10-11; Newport Beach, CA; p. 209-221.

[5] Saten ML, Owen SJ, Blacker TD. Unconstrained paving and plastering: A new idea for all hexahedral mesh generation. In: The 14th International Meshing Roundtable; 2005 Sep 11-14; San Diego, CA; p. 399-416.

[6] Wada Y, Yoshimura S, Yagawa G. Automatic mesh generation method using intelligent local approach 2nd report its extension to hexahedral mesh generation. Journal of Transactions of the Japan Society of Mechanical Engineers Series. 2001; 67(655): 496-502.

[7] Blacker TD, Meyers RJ. Seams and wedges in plastering: A 3d hexahedral mesh generation algorithm. Engineering with Computers. 1993; 2(9): 83-93.

[8] Tautges TJ, Blacker T, Mitchell SA. The whisker weaving algorithm: A connectivity-based method for constructing all-hexahedral finite element meshes. International Journal for Numerical Methods in Engineering. 1996; 39: 3327-3349.

[9] Muller-Hannemann M. Hexahedral mesh generation by successive dual cycle elimination. In: The 7th International Meshing Roundtable; 1998 Oct 26-28; Dearborn, MI; p. 365-378.

[10] Sheffer A, Etzion M, Rappoport A, Bercovier M. Hexahedral mesh generation using the embedded Voronoi graph. In: The 7th International Meshing Roundtable; 1998 Oct 26-28; Dearborn, MI; p. 347-364.

[11] Jason S, Mitchell SA, Knupp P, White D. Methods for multi-sweep automation. In: The 9th International Meshing Roundtable; 2000 Oct 2-5; New Orleans, CA; p. 77-87.

[12] Xevi R, Sarrate J. An automatic and general least-squares projection procedure for sweep meshing. In: The 15th International Meshing Roundtable; 2006 Sep 17-20; Birmingham, AB; p. 487-506.

[13] Moriya S, Hiraoka H, Kawaharada H. Replication of sharp features hexahedral meshes using modified Laplacian energy minimization. In: Proceedings of the 3rd Asian Conference on Design and Digital Engineering; 2012 Dec 6-8; Hokkaido, JAPAN; Paper No. 100062.

[14] Aim@Shape Project [Internet]. Darmstadt: Germany; [cited 2012 Jul 11]. Available from: http://shapes.aimatshape.net/

[15] Polymender [Internet]. Seattle; [cited 2012 Jul 11]. Available from: http://www1.cse.wustl.edu/~taoju/code/polymender.htm

[16] Mehra R, Zhou Q, Long J, Sheffer A, Gooch A, Mitra NJ. Abstraction of man-made shapes. ACM Transactions on Graphics. 2009; 28(5): 1-10.