Efficient Medial Voxel Extraction for Large Volumetric Models

Takashi Michikawa
The University of Tokyo
4-6-1 Komaba, Meguro-ku,
Tokyo, 153-8904, JAPAN
michi@den.rcast.u-tokyo.ac.jp

Shun Nakazaki
The University of Tokyo
4-6-1 Komaba, Meguro-ku,
Tokyo, 153-8904, JAPAN
nakazaki@den.rcast.u-tokyo.ac.jp

Hiromasa Suzuki
The University of Tokyo
4-6-1 Komaba, Meguro-ku,
Tokyo, 153-8904, JAPAN
suzuki@den.rcast.u-tokyo.ac.jp

ABSTRACT

Here we propose a method for medial voxel extraction from large volumetric models based on an out-of-core framework. The method improves upon geodesic-based approaches to enable the handling of large objects. First, distance fields are constructed from input volumes using an out-of-core algorithm. Second, medial voxels are extracted from these distance fields through multi-phase evaluation processes. Trivial medial or non-medial voxels are evaluated by the low-cost pseudo-geodesic distance method first, and the more expensive geodesic distance computation is run last. Using this strategy allows most of the voxels to be extracted in the low-cost process. This paper outlines a number of results regarding the extraction of medial voxels from large volumetric models. Our method also works in parallel, and we demonstrate that computation time becomes even shorter in multi-core environments.

Keywords:

medial surfaces, medial voxels, out-of-core algorithm, geodesic distance

1 INTRODUCTION

This paper outlines a method of creating medial voxels from large CT images. A medial voxel is a volumetric representation of medial surfaces or a set of voxels across the centerline of a volume model.

Our work is motivated by the application of the technique to digital engineering [18]. Industrial companies have recently started to utilize scanning technologies such as X-ray CT scanners and range scanners to create CAE or CAM models, which are used to accelerate the engineering process. For instance, we can achieve FEM simulation for real objects and feed the information back to CAD models.

A primary issue to be resolved is the creation of mesh models using scanned volumetric data from thin-plate objects. We can obtain meshes from solid objects using contouring algorithms [11, 8]. However, these meshes are not good for thin-plate objects because they create closed surfaces and it is hard to create FEM models from them. Instead, it is better to use medial surfaces because it is easy to create FEM meshes from such open structures.

Although medial surface extraction from polygonal models has been well documented [1, 4, 21], this cannot be used for volumetric models, and applying polygon-based models to isosurfaces created using the Marching Cubes algorithm [11] remains difficult. This is because polygon-based methods are noise-sensitive. Scanned CT images often involve noise, and the medial surfaces of such models become noisy or include many branches. In addition, the number of polygons of isosurfaces is usually large. This is why we choose a volumetric approach or compute medial surfaces from medial voxels of volumetric models. In this study, we focus on how to extract medial voxels from large volumetric models.

Medial voxels have a clear mathematical definition [15], and there are several methods of computing medial surfaces based on this definition. However, such definition-based methods create many branches for noisy volumetric models. Some voxel-based methods [16, 5, 19] are suitable for our purposes because the surfaces of engineering objects must be smooth or branchless. Such techniques evaluate each voxel using the geodesic distance of its nearest boundary points on boundary surfaces. However, these approaches are not designed for large models. Recent progress in scanning technology enables us to obtain high-resolution CT images, and industrial companies need high-resolution models to enable simulation with high accuracy. Since the size of volume models escalates the cubic order of the resolution, it is difficult for hardware devices to keep up with the memory usage required.

We propose a method for medial voxel extraction from large volumetric models based on an out-of-core framework. The method improves upon geodesic-based approaches to enable the handling of large objects. Given an input model, we first compute distance fields from the input model using an out-of-core version of the distance transform algorithm [13]. Next, we classify input volumes into medial voxels through multi-phase evaluation starting with low-cost tasks. Basically, each phase evaluates whether the geodesic distance between the two nearest boundary points is longer than a threshold. First, we introduce pseudo-geodesic distance, which is a lower bound of the geodesic distance. This can be computed from the difference vector to the nearest boundary points, and is completely local. In the latter phases, we use Dijkstra's algorithm to compute the correct geodesic distance with high cost. Then, we propose a method for computing tight bounding boxes to enable correct judgment of geodesic distance in small spaces.

The main contribution of our work lies in the design of medial surface extraction algorithms for large volumetric models. For instance, PGD-based evaluation is a completely local operation, and bounding box estimation reduces the computational costs as much as possible. These improvements directly affect performance for large objects. In particular, the out-of-core data structure used in the distance transform algorithm offers a range of benefits. First, very large-sized input models can be handled using a hard disk drive. Second, the method works in parallel; indeed, we have implemented it using multi-thread technology to allow faster results in multi-core environments.

2 RELATED WORK

Medial surfaces are the centerline surfaces of models (Figure 1 shows a simple example). The left image shows a medial surface (a set of center points in contact with two or more boundary points called nearest boundary points (NBPs)). Medial voxels are volumetric representations of medial surfaces (right).

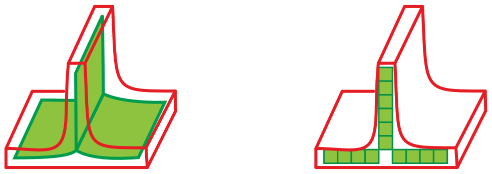


Figure 1: An example of medial surfaces(left) and medial voxels(right)

Studies on medial voxel extraction come from medial axis extraction in 2D. An example of such a survey is found in [9]. The possible methods can be classified into the thinning-based approach and the distance-based approach.

The thinning-based approach removes voxels so that topology is preserved. Sequential thinning algorithms [2, 6, 7] remove voxels step by step; since each step removes only one voxel, the object's topology is kept. However, this approach often generates bumpy surfaces because thinning algorithms check only local topology information and it is difficult to obtain smooth surfaces. On the other hand, the parallel thinning approach [10, 20, 12] removes many voxels or boundary voxels at the same time, generating relatively smooth surfaces. However, the topology management is difficult in some cases.

The distance-field-based approach resolves these issues. Prohaska and Hege proposed geodesic-based medial voxel evaluation [16] (Figure 2 shows a 2D example of this). For each voxel $\mathbf{v}$, two neighboring points $\mathbf{v}_i$ and $\mathbf{v}_j$ and their corresponding NBPs $\mathbf{N}(\mathbf{v}_i)$ and $\mathbf{N}(\mathbf{v}_j)$ are picked. It is considered that a voxel tends to be medial if the geodesic distance between $\mathbf{N}(\mathbf{v}_i)$ and $\mathbf{N}(\mathbf{v}_j)$ is longer (because the NBPs of medial voxels are located on opposite surfaces), while the distance for non-medial voxels is shorter. This technique computes medial voxels using these criteria. Since geodesics represent global information, this method is robust for noise, and hardly any unnecessary branches are generated. However, two neighboring voxels are specified for evaluation, meaning that the thickness may change according to the surface direction. While this is sufficient for visualization purposes, it is not good for surface reconstruction. Fujimori et al. extended the above algorithm in [5] to improve surface precision. Their method defines cells between neighboring voxels and finds the NBPs of these cells. Since such cells cover the voxel completely, the method is robust for direction, and the thickness is always one.

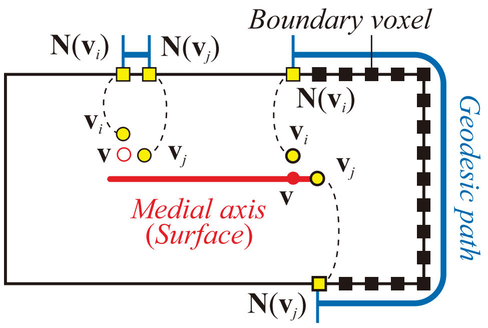


Figure 2: Geodesic-based medial voxel classification.

An alternative technique is the polygon-based approach, which obtains surface polygons of input volumes by contouring, and computes medial surfaces from these polygons. An example of this is the direct computation from polygons proposed in [1, 3, 17, 21]. However, surface polygons created from CT images have many and ill-shaped triangles, thus creating complex results. In addition, the quality of the medial sur-

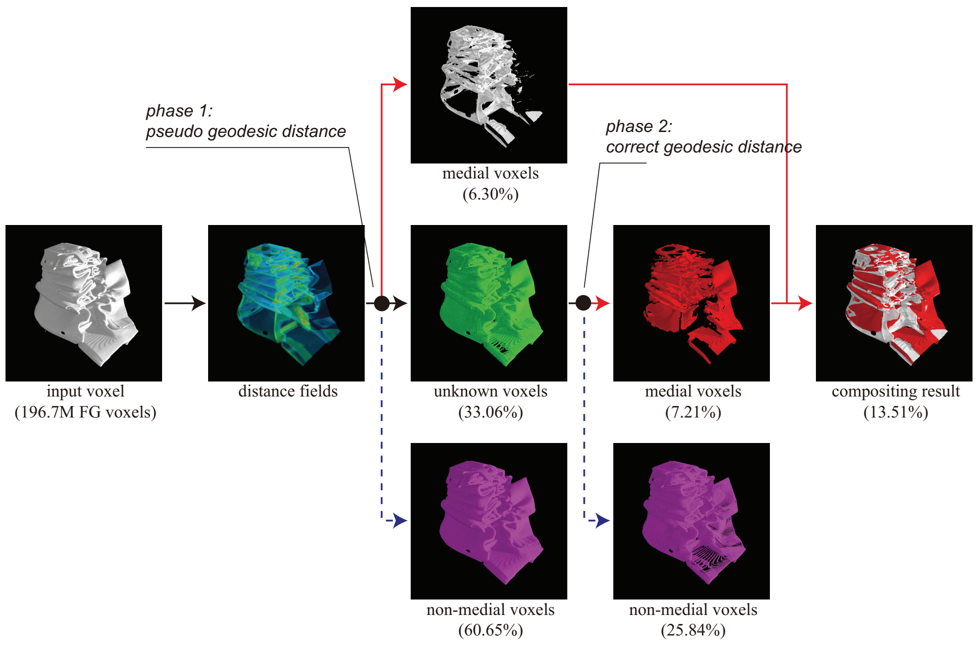


Figure 3: An overview of our algorithm.

face affects that of the surface polygons, and many branches are also generated.

3 MEDIAL VOXEL EXTRACTION FOR LARGE VOLUMES

3.1 Overview

Our method is inspired by the geodesic-based medial voxel extraction method proposed in [16, 5, 19]. A cell is evaluated as a medial voxel if the geodesic distance of the NBPs at that point is longer than the threshold, as shown in Figure 2. We suppose the input volume model is too large to fit into the memory. Figure 3 shows an overview of this algorithm. Input in this method involves a binarized volumetric model usually obtained by CT scanners in our research. The technique consists of two phases, the first of which is a distance field computation from binary images. Here, we compute not only distance values but also vector fields defined by the difference vector to the NBPs. The second phase involves medial voxel classification from distance fields. The main concept of this phase is to apply multi-phase evaluation with the aim of reducing memory usage. All voxels are first evaluated by pseudo-geodesic distance (PGD) a process in which trivial medial and non-medial voxels can be classified. Next, the remaining voxels are evaluated through a correct geodesic-based method equivalent to Dijkstra's algorithm with high cost. Then, we introduce a method of creating bounding boxes to reduce the computational costs of correct geodesic length computation as much as possible.

Notation

In this paper, italics represent scalar values, and bold text indicates vector values. For instance, $\mathbf{v}$ denotes a coordinate in volumetric space, and $d(\mathbf{v})$ denotes the distance field value at $\mathbf{v}$. $\mathbf{d}(\mathbf{v})$ denotes the distance vector or the difference vector to the nearest boundary point $\mathbf{N}(\mathbf{v})$ or $\mathbf{d}(\mathbf{v}) = \mathbf{N}(\mathbf{v}) - \mathbf{v}$.

3.2 Computing distance fields

Distance fields are first generated from binarized input models. We use out-of-core distance transforms [13] (an out-of-core framework for distance field computation) to compute these distance fields. This method decomposes an input model into sub-blocked clusters and applies distance transforms for each cluster. Inconsistency in distances between clusters can be resolved by inter-cluster propagation, and the propagated clusters are subjected to distance transform again. The advantages of this method include its ability to compute large and exact distance fields with low memory usage and its capacity to work in parallel. In our implementation, we compute the difference vector to the nearest boundary point to enable easy identification of NBPs.

3.3 Multi-phase medial voxel classification algorithm

Once the difference vector fields have been obtained, medial voxels are extracted from them.

Our strategy takes a multi-phase evaluation approach. In the first phase, obviously medial and non-medial voxels are evaluated using a rough but low-cost process. In the second phase, the remaining voxels are evaluated using a correct but expensive method. Angle-based evaluation [4] belongs to the first case, while the correct geodesic-based method [16, 5, 19] would belong to the latter case. In this paper, we introduce pseudo-geodesic distance – a combination of the above two methods – to accelerate evaluation.

In this method, we evaluate dual cells of voxels (Figure 4 shows a 2D example) as the number of neighboring voxels can be reduced from 26 to 8. For each dual voxel $\mathbf{p}$, we find the voxel pair $(\mathbf{v}_i, \mathbf{v}_j)$ for which the angle of $\mathbf{d}(\mathbf{v}_i)$ and $\mathbf{d}(\mathbf{v}_j)$ is the largest. Then, we need to check only the four cases located at the diagonal positions. This is because the angle between the distance vectors of diagonal voxels must be larger if medial surfaces exist. When the voxel pair $(\mathbf{v}_i, \mathbf{v}_j)$ is found, we can obtain the nearest boundary points $\mathbf{N}(\mathbf{v}_i)$ and $\mathbf{N}(\mathbf{v}_j)$ and apply the processes outlined below for evaluation.

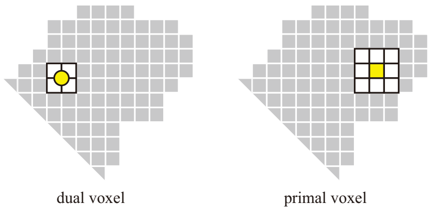


Figure 4: Dual voxels (2D). Extension to 3D is straightforward.

Threshold ε

Here we derive a threshold ε from the user-given thickness τ. If a voxel is a medial voxel, its NBPs are usually located on opposite surfaces. The minimum geodesic distance of NBPs ε is then a half circle length as follows (Eq. 1):

$$\varepsilon = \frac{\pi\tau}{2}. \tag{1}$$

Trivial non-medial voxel elimination

If two NBPs are 26-neighbors to each other, it is clear that the voxel is not a medial voxel (Figure 5), so such voxels must be eliminated first. Since we use vector distance transforms to manage the difference vectors, NBPs can be obtained from the point and distance vectors of neighboring voxels.

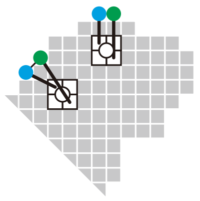


Figure 5: 26-neighbor NBPs imply that the voxel is a non-medial voxel.

Pseudo-geodesic distance

Pseudo-geodesic distance (PGD) is a simplified version of geodesic distance without Dijkstra's algorithm(Figure 6). Given a point $\mathbf{v}$ and its NBPs, PGD $\tilde{g}(\mathbf{v})$ is defined as follows (Eq. 2):

$$\tilde{g}(\mathbf{v}) = d(\mathbf{v})\theta, \tag{2}$$

where θ denotes the angle between $\mathbf{d}(\mathbf{v}_i)$ and $\mathbf{d}(\mathbf{v}_j)$. Note that this can be computed only for neighboring voxels, making it a local operation.

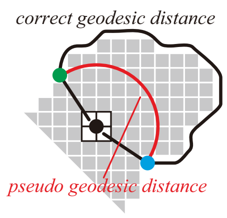


Figure 6: Pseudo-geodesic distance.

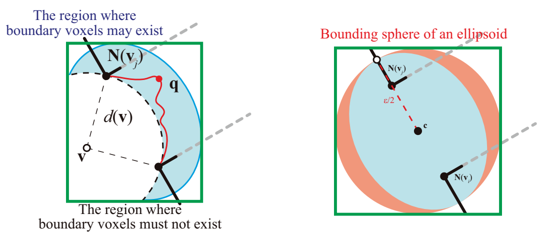
PGD can be used for medial voxel evaluation instead of CGD. If PGD is larger than a certain threshold ε, the voxel is considered medial because it represents a lower bound of CGD. A brief proof can be derived from the definition of distance fields or there is no point $\mathbf{q}$ on the correct geodesic path between $\mathbf{N}(\mathbf{v}_i)$ and $\mathbf{N}(\mathbf{v}_j)$ such that $||\mathbf{v} - \mathbf{q}|| < d(\mathbf{v})$. If CGD is shorter than PGD, point $\mathbf{q}$ must exist on the inside of the arc. This is why CGDs are always longer than PGDs.

Optimal bounding box estimation for computing correct geodesic distance

Some voxels are not classified into medial or non-medial voxels by the PGD-based evaluation outlined in the previous subsection; conventional correct geodesic evaluation algorithms are applied to the remaining voxels. Note that correct geodesic paths are not required here. We simply need to know whether the geodesic distance is longer than the threshold. Here, we introduce a tight bounding box to detect any geodesic paths shorter than the threshold ε. The basic idea is to estimate the region of the points at which the geodesic path $\mathbf{q}$ can exist. Suppose we know point $\mathbf{v}$, its distance d and the two NBPs $\mathbf{N}(\mathbf{v}_i)$ and $\mathbf{N}(\mathbf{v}_j)$; in this case, we can specify the region as follows:

- $\mathbf{q}$ can exist within an ellipsoid or $||\mathbf{N}(\mathbf{v}_i) - \mathbf{q}|| + ||\mathbf{N}(\mathbf{v}_j) - \mathbf{q}|| < \varepsilon$.
- $\mathbf{q}$ never exists on the inside of a sphere with center point $\mathbf{v}$ and radius $d(\mathbf{v})$ or $||\mathbf{v} - \mathbf{q}|| > d(\mathbf{v})$.

By compositing these criteria, we can obtain the region where the path point $\mathbf{q}$ exists. A tight bounding box is then formed, as shown in Figure 7 (a). However, the computation of the bounding box shown in Figure 7 (a) is not simple, and we use a simple bounding box of a bounding sphere for an ellipsoid with center point $\mathbf{c} = \frac{1}{2}(\mathbf{N}(\mathbf{v}_i) + \mathbf{N}(\mathbf{v}_j))$ and a radius of $\frac{\varepsilon}{2}$ as an alternative (as shown in Figure 7 (b)). Note that the bounding box of the ellipsoid (in light blue) is also simple, and a tighter bounding box can be obtained.


The region where boundary voxels may exist
N(v_j)
q
d(v)
v
The region where boundary voxels must not exist
Bounding sphere of an ellipsoid
N(v)
ε/2
c
N(v_j)

(a) Optimal box (b) Simplified box

Figure 7: Estimation of bounding boxes for correct geodesic distance computation.

Once the bounding box is obtained, Dijkstra's algorithm is run from one NBP (the source point) to the other NBP (the target point), and evaluation is performed as follows:

- If the propagation reaches the target point, $\mathbf{v}$ is evaluated as a non-medial voxel if the distance is shorter than ε, otherwise it is a medial voxel.
- If propagation is stopped before reaching the target point, $\mathbf{v}$ is evaluated as a medial voxel.

4 RESULTS AND DISCUSSION

We implemented the above algorithm on a Win32 executable. Figure 8 shows the experimental results for large CT images of engineering objects. (a) shows input voxels, and (b) shows extracted medial voxels. White voxels represent those extracted through pseudo-geodesic distance-based evaluation, while voxels in red are those extracted using correct geodesic distance-based evaluation. (c) shows the intersection of medial voxels, with medial voxels in red and non-medial voxels in white. We can see that it is possible to compute medial voxels from large scanned models.

Since the pseudo-geodesic distance guarantees the lower bound of the correct geodesic distance, a set of medial voxels obtained by this pseudo metric becomes a subset of medial voxels obtained by the correct geodesic distance. In addition, it is clear that voxels with NBPs that are in contact with each other are not medial voxels. Thus, the results are the same as those of the correct geodesic-based method.

The choice of a threshold ε or a thickness τ affects quality. The algorithm becomes noise-sensitive with small values of ε, but the computation time is faster because the bounding boxes become smaller. On the other hand, the algorithm becomes noise-robust for large values of ε, but computation is slower. In addition, the boundaries of surfaces will be shrunk.

Table 1 shows a number of related statistics. Most of the computation time required is used for medial voxel classification, and the time taken depends on the thickness. However, the classification process is independent of other voxels. Indeed, we developed this prototype for a multi-core environment. We confirmed that the speed of the two-thread mode is 1.5 times faster than that of a single thread. However, the result for many threads is somewhat slow (1.8 times faster for four-thread, twice faster for eight-thread), which can be attributed to parallelization for each cluster. This means that the speed depends on the computation time of the cluster, which is why the speed is not as fast as we had expected.

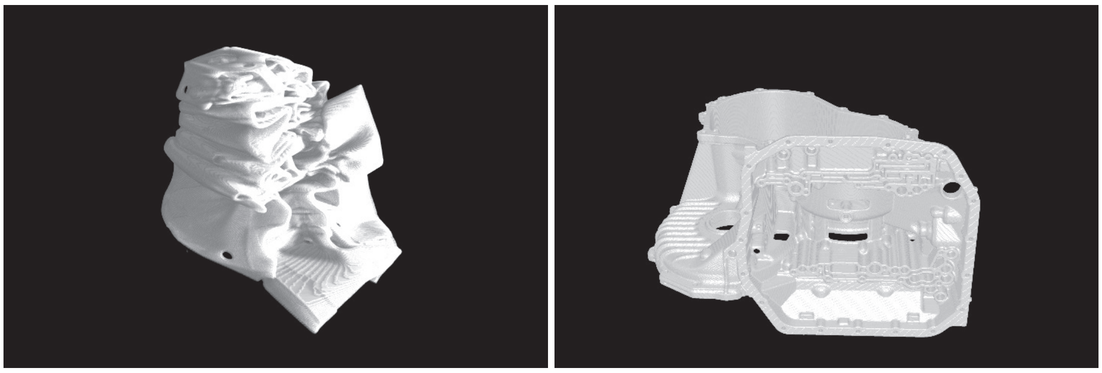
It should be noted that about 60% of foreground voxels are judged at the pseudo-based evaluation stage. This means that most voxels are locally processed, indicating that our pseudo-metric contributes to cost reduction.

Our method has a number of limitations. First, current implementation allows the user to specify only one threshold or the thickness of thin plates. If CT images involve two or more thickness values, the quality may deteriorate. For instance, a threshold that is too small will result in branches(Figure 9(a)), and values that are too large will create shrinkage of medial surfaces(Figure 9(b)). Distance fields will be used to estimate thickness, as this allows adaptive specification of a good threshold so that branches and shrinkage do not appear. The other issue is the input data used; the superposition of input models as binary images means that some grayscale information is lost.

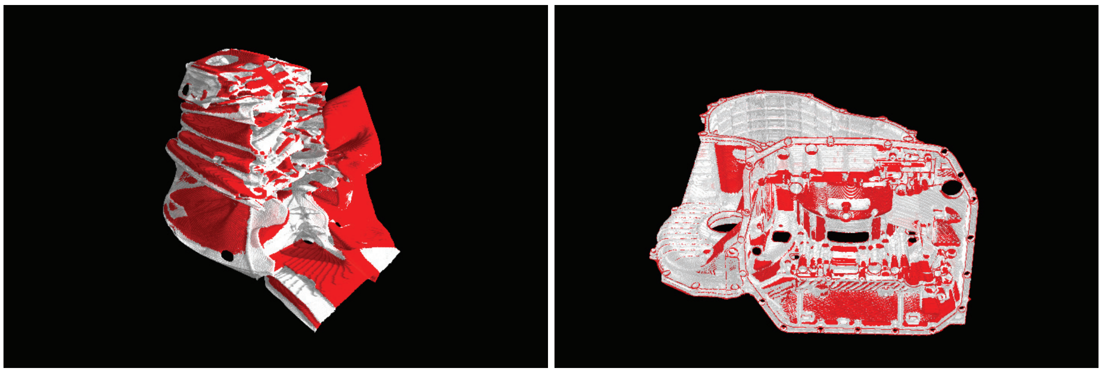
5 CONCLUSION

We have introduced a medial voxel extraction algorithm for large objects using a multi-phase evaluation strategy. Each voxel is first classified by pseudo-geodesic distance evaluation, which works locally and enables the identification of trivial medial and non-medial voxels. The remaining voxels are classified using the conventional geodesic evaluation algorithm. We also introduced a method of constructing tight bounding boxes to evaluate correct geodesic length at low cost, and applied the technique to several examples to show that large medial voxels can be computed. In addition, the method works in parallel, and faster results in multi-core environments were confirmed.

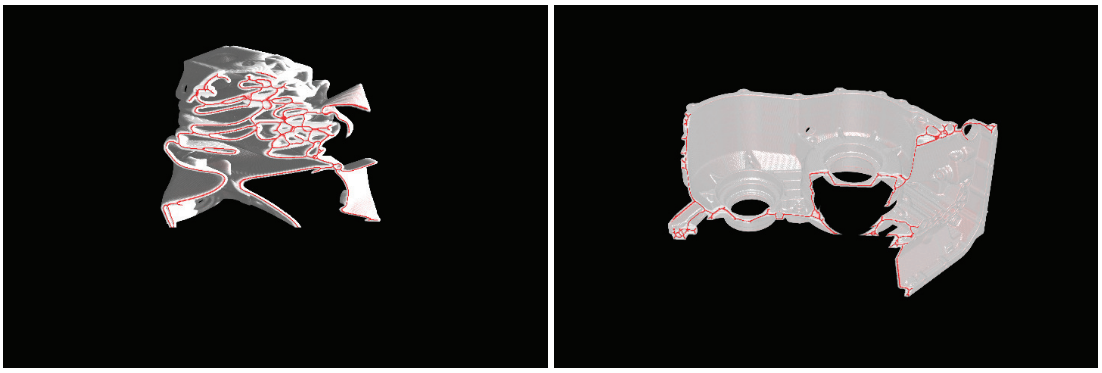
In future work, we plan to develop a method to automatically reconstruct CAX models of thin-plate large engineering objects. To this end, we aim to develop medial surface reconstruction from medial voxels. Since our algorithm still involves the formation of small branches, we would like to remove these in the surface reconstruction phase. For instance, weighted



(a) input



(b) Medial voxels (Red: medial voxels obtained by correct geodesic distance evaluation. White: medial voxels obtained by pseudo-geodesic distance evaluation)

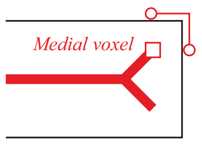


(c) Intersection (Red: medial voxels)

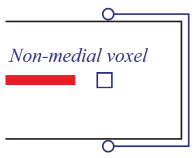
Figure 8: Experimental results for crushed side-frame (left) and cylinder-head (right) CT images

	Resolution	Parameters		Time (min.)			Persentage (%)	
Name	(#clusters)	τ	Pseudo	DF	Medial	Sum	Pseudo	Correct
Transmission cover	1,500 x 1,500 x 668	3	y	11.56	37.71	49.28	70.38	29.62
	(10 x 10 x 7)							
Crushed side frame	708 x 965 x 325	3	y	2.50	18.43	20.92	66.95	33.05
	(7x9x3)	3	n	2.48	53.80	56.28	N/A	100.00
		2	y	2.49	9.42	11.91	73.07	26.93

Table 1: Time comparison with different numbers of threads (measured on an Intel Xeon 3.16 GHz *2). DF : distance field computation time. Medial : Medial voxel evaluation time.



(a) small ε



(b) large ε

Figure 9: Choice of ε makes (a) branches or (b) shrinkage of medial surfaces.

Delaunay triangulation for polygonization of point sets [14] may be suitable for this purpose.

ACKNOWLEDGEMENT

The authors would like to thank anonymous reviewers and Yutaka Ohtake for valuable comments. Transmission cover data is courtesy of Toyota. This work is partially supported by Grant-in-Aid for Young Scientists(B) (No.20760096), Grant-in-Aid for Scientific Research(B) (No.19360070) and RCAST grant for young scientists from the University of Tokyo.

REFERENCES

[1] Nina Amenta, Sunghee Choi, and Ravi Krishna Kolluri. The power crust. In *Proceedings of the sixth ACM symposium on Solid modeling and applications*, pages 249–266, New York, NY, USA, 2001. ACM.

[2] G. Borgefors, I. Nystrom, and G. Sanniti Di Baja. Computing skeletons in three dimensions. *Pattern Recognition*, 32(7):1225–1236, 1999.

[3] Tim Culver, John Keyser, and Dinesh Manocha. Accurate computation of the medial axis of a polyhedron. In *Proceedings of the fifth ACM symposium on Solid modeling and applications*, pages 179–190, New York, NY, USA, 1999. ACM.

[4] Mark Foskey, Ming C. Lin, and Dinesh Manocha. Efficient computation of a simplified medial axis. In *Proceedings of the eighth ACM symposium on Solid modeling and applications*, pages 96–107, New York, NY, USA, 2003. ACM.

[5] Tomoyuki Fujimori, Hiromasa Suzuki, Yohei Kobayashi, and Kiwamu Kase. Contouring medial surface of thin plate structure using local marching cubes. *International Conference on Shape Modeling and Applications*, 0:297–306, 2004.

[6] Jun ichiro Toriwaki and Kensaku Mori. Distance transformation and skeletonization of 3d pictures and their applications to medical images. In *Digital and Image Geometry, LNCS 2243*, pages 412–428, 2001.

[7] Tao Ju, Matthew L. Baker, and Wah Chiu. Computing a family of skeletons of volumetric models for shape description. *Computer Aided Design*, 39(5):352–360, 2007.

[8] Tao Ju, Frank Losasso, Scott Schaefer, and Joe Warren. Dual contouring of hermite data. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 339–346, New York, NY, USA, 2002. ACM.

[9] L. Lam, S.W. Lee, and C.Y. Suen. Thinning methodologies-a comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(9):869–885, 1992.

[10] Christophe Lohou and Gilles Bertrand. A 3d 6-subiteration curve thinning algorithm based on p-simple points. *Discrete Applied Mathematics*, 151(1-3):198–228, 2005.

[11] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 163–169, New York, NY, USA, 1987.

[12] A. Manzanera, T.M. Bernard, F. Preteux, and B. Longuet. Medial faces from a concise 3d thinning algorithm. *IEEE International Conference on Computer Vision*, 1:337, 1999.

[13] Takashi Michikawa, Ken'ichiro Tsuji, Tomoyuki Fujimori, and Hiromasa Suzuki. Out-of-Core Distance Transforms. In *Proceedings of the 2007 ACM symposium on Solid and Physical Modeling*, pages 151–158. ACM Press, 2007.

[14] Yutaka Ohtake, Alexander Belyaev, and Hans-Peter Seidel. A composite approach to meshing scattered data. *Graphical Models*, 68(3):255–267, 2006.

[15] Joseph O'Rourke. *Computational geometry in C*. Cambridge University Press, 2nd edition, 2001.

[16] Steffen Prohaska and Hans-Christian Hege. Fast visualization of plane-like structures in voxel data. In *Proceedings of the conference on Visualization '02*, pages 29–36, Washington, DC, USA, 2002. IEEE Computer Society.

[17] D. J. Sheehy, C. G. Armstrong, and D. J. Robinson. Computing the medial surface of a solid from a domain delaunay triangulation. In *Proceedings of the third ACM symposium on Solid modeling and applications*, pages 201–212, New York, NY, USA, 1995. ACM.

[18] Hiromasa Suzuki. Convergence engineering based on X-ray CT scanning technologies. In *Proceeding of JSME Digital Engineering Workshop*, pages 74–77, 2005.

[19] Hiromasa Suzuki, Tomoyuki Fujimori, Takashi Michikawa, Yasuhiko Miwata, and Noriyuki Sadaoka. Skeleton surface generation from volumetric models of thin plate structures for industrial applications. In *IMA Conference on the Mathematics of Surfaces, LNCS 4647*, pages 442–464, 2007.

[20] Y.F.Tsao and K.S. Fu. A parallel thinning algorithm for 3-d pictures. *Computer Graphics and Image Processing*, pages 315–331, 1981.

[21] Shin Yoshizawa, Alexander G. Belyaev, and Hans-Peter Seidel. Free-form skeleton-driven mesh deformations. In *Proceedings of the eighth ACM symposium on Solid modeling and applications*, pages 247–253, 2003.