ACM Symposium on Solid Modeling and Applications (2004)
G. Elber, N. Patrikalakis (Editors)

3D Discrete Skeleton Generation by Wave Propagation on PR-Octree for Finite Element Mesh Sizing

W. R. Quadros+, K. Shimada+, and S. J. Owen*

+Dept. of Mechanical Engineering, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA – 15213, USA
*Sandia National Laboratory[1], P.O. Box 5800 MS 0847, Albuquerque, NM 87185-0847, USA

Abstract
This paper proposes a new algorithm to generate a disconnected, three-dimensional (3D) skeleton and an application of such a skeleton to generate a finite element (FE) mesh sizing function of a solid. The mesh sizing function controls the element size and the gradient, and it is crucial in generating a desired FE mesh. Here, a geometry-based mesh sizing function is generated using a skeleton. A discrete skeleton is generated by propagating a wave from the boundary towards the interior on an octree lattice of an input solid model. As the wave propagates, the distance from the boundary and direction of the wave front are calculated at the lattice-nodes (vertices) of the new front. An approximate Euclidean distance metric is used to calculate the distance traveled by the wave. Skeleton points are generated at the region where the opposing fronts meet. The distance at these skeleton points is used to measure both proximity between geometric entities and feature size, and is utilized to generate the mesh size at the lattice-nodes. The proposed octree-based skeleton is more accurate and efficient than traditional voxel-based skeleton and proves to be great tool for mesh sizing function generation.

Categories and Subject Descriptors (according to ACM CCS) [J.2 Physical Sciences And Engineering]- *engineering*

1. Introduction
Because mesh size is crucial in obtaining accurate analysis results, there is a great demand for automatically generating a desired FE mesh sizing function. A FE mesh is a discretization of a continuous domain, and is used in analyzing complex structures and continua in various scientific and engineering fields using a versatile and powerful numerical procedure called the finite element method (FEM). As the accuracy of the analysis results depends on the quality of the graded mesh and the quality of the graded mesh mainly depends on the sizing function, generating a desired mesh sizing function is crucial.

A geometry-adaptive graded mesh contains significantly fewer elements (~10 times less in Figure 1) while maintaining the mesh quality by having fine elements at small features; therefore reduces computation time and memory usage during analysis without sacrificing the accuracy. In the geometry-adaptive mesh, the mesh size is adapted based on the geometry of the input solid.

The mesh sizing function mainly depends on the geometric factors, such as the proximity between the geometric entities, feature size, and surface curvature. These geometric factors are used to generate an initial high quality graded mesh, which gives sufficiently accurate preliminary FE analysis results in a short duration. The accuracy of analysis can be later improved by refinement and coarsening the initial geometry-adaptive mesh based on the error maps of preliminary analysis.

In this paper a "skeleton" refers to a set of disconnected points that provides local thickness information and is generated using the concepts of medial axis transform (MAT). The MAT is defined as the locus of the center of the maximal sphere as it rolls inside a solid, along with the associated radius function [Blu67]. Even though the radius function of MAT is an accurate tool to measure proximity, it is computationally expensive to generate the continuous MAT. As FE mesh itself is a discretization of a continuous domain, here a set of accurate enough disconnected skeleton points is generated, using a less expensive, easy to implement algorithm. The skeletons have proved to be useful tool in many diverse applications including mesh generation [SNT*92, GP92, QRP*01]. This paper describes a new application of a skeleton in generating 3D finite element mesh sizing function.

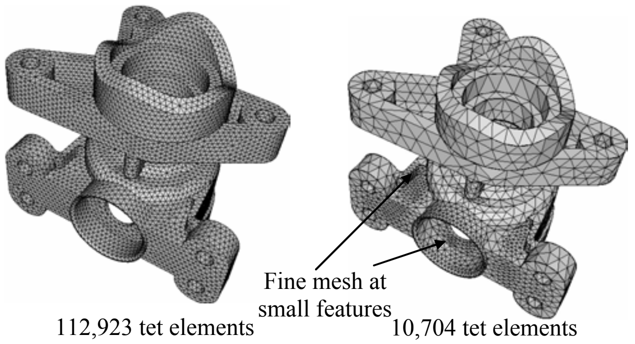


Figure 1 *Uniform and geometry-adaptive mesh*

2. Literature review
In this section, literature on both the mesh sizing function and skeleton generation is briefly discussed.

2.1 Finite element mesh sizing
As the FE meshing algorithms do not see the complexity of geometry upfront, it is worthwhile to split the meshing process into two steps: (1) Generate the mesh sizing function and (2) Generate FE mesh using the mesh sizing function. Even though

1 Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000

much research has been done in developing automatic FE mesh generation algorithms, in many of these algorithms only some aspect of mesh sizing are integrated. In the following paragraphs previous mesh sizing approaches are briefly compared with the proposed method.
Cunha et al. [CCS97] and Zhu et al. [ZBS02] estimated proximity by first representing the curves and surfaces with few sample points and then distance between all combinations is calculated. Researchers [She88] have used cell (cube) size of an quadtree and octree as the means for mesh sizing. Octree is orientation sensitive and it is difficult to control the sizing gradient. There is a very high probability of having small size cubes adjacent to a relatively large cube, resulting in abrupt gradients in sizing function. The disconnected skeleton proposed in this paper is less expensive and more accurate in measuring proximity and feature size, and generates a smooth sizing function.

Even though MAT has been used in mesh generation, no work has been done specifically in generating the 3D mesh sizing function. Researchers [SNT*92, GP92, QRP*01] have used MAT to decompose the domain into simpler sub-domains and to control the nodal spacing or element size. But these approaches do not specifically generate the mesh sizing function.
Authors [QSO03] presented an approach for generating 3D mesh sizing function using skeleton generated from a voxel model of the input solid. The voxel-based approach works well with bulky objects that contain no small features; however, for complex industrial models, the voxel needs a huge memory and is computationally expensive. To overcome these limitations, the Point-Region-Octree (PR-Octree) based method [QSO03] was developed. Recently, Tchon et al. [TKG*03] presented a similar work, but for generating anisotropic metric rather than generating sizing function using octrees and skeletons; in their approach a digital skeleton consisting of set of octree cells was generated using the concepts from digital topology. In this paper, a new, computationally efficient, easy to implement, octree-based skeleton algorithm that captures intricate features is proposed, and this skeleton has been used specifically in generating geometry-adaptive 3D FE mesh sizing function.

2.2 Skeletons

In the literature, properties of MAT have been explored in developing various algorithms for the generation of the continuous and discrete skeletons. One of the important properties used is that, the medial exists where the grass fire propagated from the boundary meet. As the proposed approach relies on this property, the algorithms developed based on this property are discussed in the following paragraphs.

Thinning algorithms [LLC92, ZW93] have been used on pixel and voxel image data in the areas of pattern recognition and image processing to generate digital skeletons. Thinning or erosion operations are performed in a layer by layer manner from the boundary to generate a set of pixels/voxels, which forms the digital skeleton. Thining algorithms tend to produce excessive erosion and have to be constrained. These skeletons do not completely represent the initial image. These voxel-based approaches consume huge memory and are not suitable for complex industrial models containing small features.

Another class of skeleton schemes is based on the distance map transformations. These algorithms [Dan80, Rag93] exploit the fact that the medial points coincide with the singularities of Euclidean distance function to the boundary. Siddiqi and Bouix [SB99] used differential equations to simulate the inward progress of the wave and medial points were generated if the mean flux entering the neighborhood of a point is positive. In these approaches, the numerical detection of singularities is a non-trivial problem, and ensuring homotopy with the original object is difficult.

The proposed approach combines the concepts of the thinning process, the generation of distance maps, and the bisector property (medial point exists at an equal distance from at least two points on the boundary). Unlike, traditional thinning where voxels are removed in a layer by layer pattern, here octree lattice-nodes (vertices) are removed incrementally from the wave front and new adjacent nodes are inserted. Like distance map approaches, distance traveled and direction of wave is calculated at each internal lattice-node. The skeleton points are generated where the opposing fronts meet, using the distance and direction stored at the opposing lattice-nodes, such that the skeleton point is approximately equidistant from the boundary.

3. Statement of the problem

The goal of this paper is to develop an algorithm to generate a discrete 3D skeleton, and to use that skeleton in generating a geometry-adaptive FE mesh sizing function. The geometry-adaptive mesh sizing function that depends on proximity, feature size, and curvature, has to meet certain requirements: the mesh size should be bounded between the minimum size (d_{min}) and the maximum size (d_{max}); the gradients should be within the predefined limit α. A more formal statement is given below.

Given a solid $S \subseteq R3$ and the bounds of mesh size dmax and dmin,

1. Generate a set of disconnected skeleton points M(S)

2.Generate a geometry-based FE mesh sizing function s on an Octree O(S) by interpolating M(S), such that,

- *Mesh size $d = s(p)$ where $p(x,y,z) \in S$ and $d_{min} \le d \le d_{max}$.*
- *s is α-Lipschitz, i.e., for any two points $p1$, $p2 \in S$*

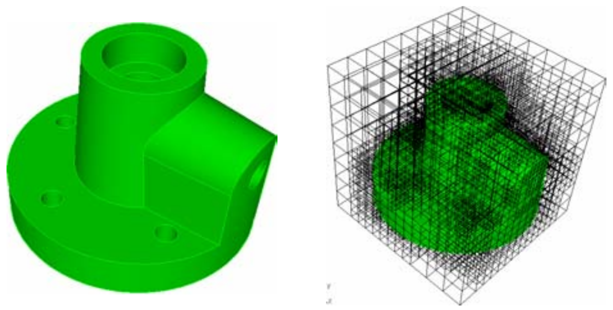
  $| s(p1) - s(p2) | \le \alpha \, || p1 - p2 ||$.

4. Overview of the algorithm

The steps involved in generating the FE mesh sizing function using a disconnected skeleton are shown in Figure 2 and are described in the following paragraphs. Figure 2(a) shows a typical industrial input CAD model. In this paper a B-rep solid model, such as ACIS sat file, is considered. Other input formats, such as faceted models, meshed models, and volume data can be handled with slight modifications. Figure 2(b) shows the PR-Octree of the solid model generated using the graphics facets of the solid (details are given in Section 5).

The next challenging step is to propagate a wave from the boundary towards the interior on the non-uniform PR-Octree lattice to generate a trimmed disconnected skeleton, which is shown in Figure 2(c) and is explained later in detail in Section 6. The complete medial formed when the wave collides (see Figure 6) is trimmed for mesh sizing application. The radius function or distance traveled by the wave at the skeleton points is shown using color scale in Figure 2(c). The skeleton points where the wave has traveled the nearest and furthest are shown in red and blue, respectively.

In the last step, the radius function of the skeleton points, which measures proximity and feature size, and the minimum principal radius of curvature, which measures surface curvature, are used to interpolate the mesh size over the PR-Octree lattice. Figure 2(d) shows the interpolated size on the PR-Octree using a color scale and Section 7 gives the details of the interpolation. The PR-Octree is an efficient means for storing the mesh sizing function and it reduces the computational cost of determining the size at a point during mesh generation.


(a) Input B-rep solid model (b) PR-Octree lattice

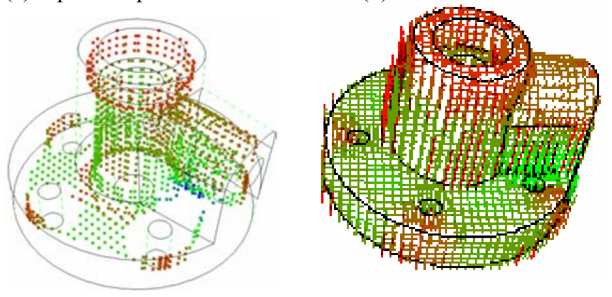

(c) Disconnected 3D skeleton (d) Interpolation on octree

Figure 2 *Overview of the algorithm*

5. Pr-Octree lattice generation

PR-Octree is selected over other types of octree (a hierarchical spatial data structure) [Sam95], because it provides a suitable lattice for the propagation of wave while generating skeleton and serves as an efficient mechanism to store the mesh sizing function. By definition, the PR-Octree contains cells which are either empty, or contain only one point. Therefore the cell (cube) size depends on the density of the input points. The vertices and centroid of the facets of the solid are given as the input points while generating PR-Octree. Smaller facet edges exist at small features and high curvature regions. Therefore the density of facet points will be higher at these regions as shown in left side image of Figure 3. Thus small-sized octree cells are generated in these regions as shown in the right side image of Figure 3. As larger facets may exist at planar regions, the PR-Octree is subdivided completely till a user specified minimum depth (min_depth), which determines the max size of the cell. Now to establish a smooth transition in the cell size, the larger cells are further subdivided until only 1-level of depth difference is maintained between the adjacent cells. Thus a suitable lattice is prepared for wave propagation in generating a disconnected skeleton.

One disadvantage of the PR-Octree is that maximum depth depends on the minimum distance between any two points; therefore maximum depth (max_depth) of the PR-Octree is taken as a user input. This reduces the memory usage and the computational cost.

5.1 Data structures for octree lattice

Finding adjacent cells in an octree usually requires traversing up till the desired parent cell and then descending down the tree in search of neighboring cell [Sam89]. This is an expensive process. To overcome this problem, each lattice-node stores eight pointers (cells[2][2][2]) to store the incident cells. Using these pointers, adjacent cells can be easily found and thus reduces the computational time while ensuring 1-level of depth difference between the adjacent cells.

The lattice-nodes of the octree are stored using a graph data structure. Each lattice-node will contain eight pointers to the adjacent lattice-nodes. The graph data structure reduces the time complexity of the wave propagation, which is discussed in section 6.2.

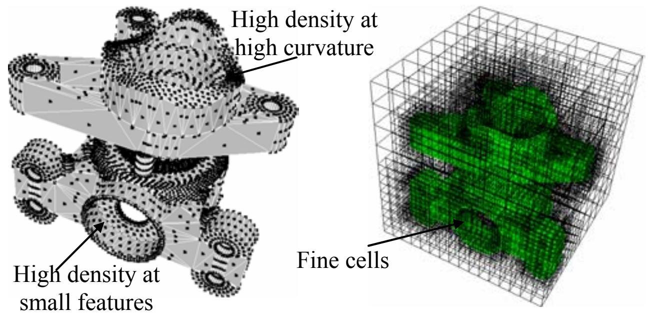


Figure 3 *Facet vertices and centroid, and PR-Octree*

6. Skeleton generation

A disconnected skeleton is generated by propagating a wave from the boundary towards the interior on the PR-Octree. The three important phases of wave propagation are initiation of the wave, propagation of the wave front, and termination of the wave. The details of these three phases are described in the following subsections.

6.1 Initiation of the wave

Because the wave must propagate from the boundary, a set of lattice-nodes representing the initial front at the boundary are generated by intersecting octree lattice with the facets. Finding the intersection point is necessary for marking the lattice-nodes and accurately calculating the distance travelled by the wave from the boundary, at the lattice-nodes of the initial front. All the lattice-nodes are initially marked UNVISITED. After the intersection, the two lattice-nodes of the lattice segments that intersect with the facets are marked. If the lattice-node is outside the solid, i.e., lattice-node is in positive half space of facet, then that lattice-node is marked VISITED_EXT. If the lattice-node is inside the solid, and it is not already marked VISITED_EXT, then it is marked VISITED_INT. After the marking of lattice-nodes, a layer of VISITED_INT lattice-nodes exist just inside the boundary, which forms the initial front and are shown with arrows in Figure 4. Rest of the internal lattice-nodes remain UNVISITED and during wave propagation these will be marked VISITED_INT.

Unlike some of the previous approaches [LLC92], here a more accurate distance metric is used instead of the Manhattan distance metric while calculating the distance travelled by the wave. At each VISITED_INT lattice-node of the initial front, the distance is set equal to the minimum length perpendicular drawn from the lattice-node to the associated facets. The distance of VISITED_EXT lattice-nodes are set to -1.0.

Surface ID (identity number) and direction of wave (reversed facet normal) are also stored at the lattice-nodes of the initial front. The normal stored is used in calculating the distance at the lattice-nodes of subsequent fronts, which is explained in next section (see Equation 1). Surface ID is used in deleting certain discrete skeleton points, which is explained in Section 6.3.

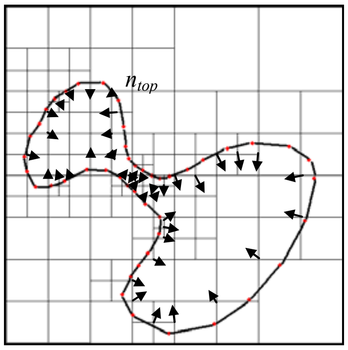


Figure 4 *Direction of wave at initial front*

6.2 Propagation of the wave front

Wave propagation on a non-uniform octree lattice is more challenging than traditional voxel-based approaches [LLC92, ZW93] and is described in Algorithm 1. The propagation starts from the initial front, which consists of a set of VISITED_INT lattice-nodes at the boundary. A heap data structure is used to store the lattice-nodes of the initial front. As the wave propagates, the top lattice-node of the heap ('n_{top}' in Figure 4) is popped out one at a time as given in Algorithm 1, and the valid adjacent lattice-nodes of the popped out lattice-node are pushed into the heap in Function Visit_Node. The top of the heap ('n_{top}' in Figure 4) is always the lattice-node where the wave has traveled the least from the boundary. The distance traveled by the wave, surface ID, and normal, at the adjacent lattice-nodes added to the front are calculated, and the new lattice-nodes are marked VISITED_INT. The wave fronts meet at the skeleton points and no additional points are pushed into the heap. Thus all the interior lattice-nodes are popped out and visited only once in an incremental manner. The wave propagation ends when the number of lattice-nodes in the current front becomes zero.

Algorithm 1: Wave propagation on PR-Octree Lattice
Input: A list of VISITED_INT lattice-nodes *B* of the initial front and octree lattice *O*.
Output: Find distance and normal of lattice-nodes, and generate skeleton points.

```
Begin
Insert all VISITED_INT lattice-nodes into a priority queue $H$.
While ($|H| \neq 0$)
    $n_{top} \leftarrow H$.Pop()
    Visit_Node($n_{top}$, $H$)
End
```

In Function Visit_Node, either wave is propagated forward by adding the adjacent node (n_{adj}) of top node (n_{top}) into the heap or wave is terminated by generating skeleton points. If n_{adj}, is not NULL, is not VISITED_EXT, and has Surface ID equal to NULL (not yet visited by wave), then distance ($\text{dist\_n}_{\text{adj}}$) at n_{adj} is calculated as given in Equation 1. If the $\text{dist\_n}_{\text{adj}}$ is less than or equal to distance at n_{top}, then n_{adj} is not added into the heap to avoid the backward or sidewise movement of the wave. Otherwise, the nadj is added into the heap, and wave is propagated forwards. The termination of the wave and generation of the skeleton points are discussed in detail in next section.

$$\text{dist\_n}_{\text{adj}} \leftarrow \text{dist}(n_{top}) + (\text{coord}(n_{adj}) - \text{coord}(n_{top})).\text{normal}(n_{top}) \quad \textbf{(1)}$$

6.3 Termination of the wave

Possible cases in which the fronts meet are examined here. In Figure 5 the path traced by the propagated wave on the lattice is shown in dashed lines. The paths are classified based on the number of lattice-nodes visited during propagation. Cases I, II, and III contain zero, one, and two or more, lattice-nodes (q in Figure 5) respectively.

The position and radius of the skeleton points in all three cases are calculated in order to keep the skeleton point approximately equidistant from the surfaces, whose ID is stored at the opposing lattice nodes. Note that the opposing lattice-node (q) will have the ID of closest surfaces, because in Function Visit_Node, the surface ID at new lattice-node is updated based on minimum distance from the facet. In Figure 5 and Equations 2 and 3, 'd_i' denotes the distance (approximated minimum distance from closest surface) at lattice-node $\mathbf{q_i}$, and $\mathbf{n_i}$ denotes the facets' normal, stored at the lattice-node $\mathbf{q_i}$. The intersection points between facet and lattice-segment is represented by $\mathbf{p_1}$ and $\mathbf{p_2}$. The position vector $\mathbf{m_q}$, and radius r_q, of the skeleton point in Cases II and III, are given by Equations 2 and 3 respectively. Case I can be converted into Case II by inserting a virtual lattice-node $\mathbf{q}$ at the mid point of the line segment connecting $\mathbf{p_1}$ and $\mathbf{p_2}$.

$$\mathbf{m}_q = \mathbf{q} + \left(\frac{d_2 - d_1}{2}\right) \cdot \mathbf{n_1}\, ; \; r_q = \left(\frac{d_1 + d_2}{2}\right) \quad \textbf{(2)}$$

$$\mathbf{m}_q = \mathbf{q}_1 + \left(\frac{d_2 - d_1 + d}{2}\right) \cdot \mathbf{n_1}\, ; \; r_q = \left(\frac{d_1 + d_2 + d}{2}\right) \quad \textbf{(3)}$$

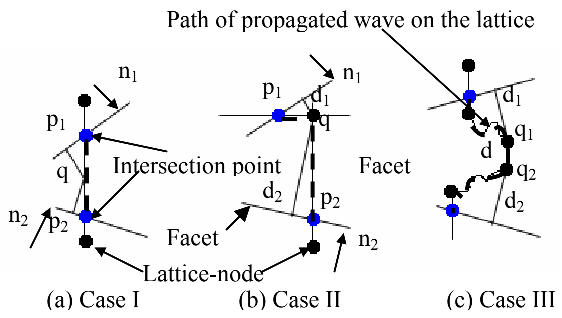


Figure 5 *Possible cases of termination of wave front*

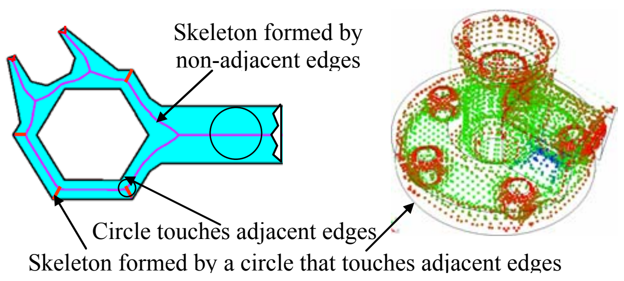

Skeleton formed by non-adjacent edges
Circle touches adjacent edges
Skeleton formed by a circle that touches adjacent edges

Figure 6 *Complete skeleton formed by adjacent and non-adjacent edges/surfaces*

For mesh sizing function generation, only a subset of skeleton points are considered. Figure 6 shows the complete 2D and 3D skeleton. As the radius of skeleton points formed by adjacent edges goes to zero at convex vertices/edges, these points will generate a fine mesh. The surface ID and the facet normal stored at the lattice-nodes of the opposing wave front are used in trimming the skeleton, which is shown in Figure 2(c).

7. Interpolation of mesh size

In this section the details of interpolating mesh size on the PR-Octree using source points is described. First the source points due to the skeleton are generated. Later additional source points based on surface curvature and user specified size can be generated. A source point is defined by: size, s; position, p[x,y,z]; scope radius, rad; and local sizing function , f. The s, p, and rad of a skeleton source point are set equal to the radius, center, and radius of corresponding skeleton point respectively; and a linear function has been used for f.

The size at a VISITED_INT lattice-node is determined by interpolating the size of the source points containing that lattice-node. The interpolation method used to calculate the size, sn at an lattice-node, n from a set of m source points of particular type (skeleton, curvature, or user specified) is given in Equation 4. Average of the normalized weights, due to inverse square distance and inverse square size, is used as the weight at every source point.

$$s_n = \sum_{i=1}^{i=m} s_i \times \left(\frac{\mathrm{Wdist}_i + \mathrm{Wsize}_i}{2} \right) \qquad \textbf{(4)}$$

where $\mathrm{Wdist}_i = \dfrac{\frac{1}{d_i^2}}{\sum_{j=1}^{j=m} \frac{1}{d_j^2}}$, $\mathrm{Wsize}_i = \dfrac{\frac{1}{s_i^2}}{\sum_{j=1}^{j=m} \frac{1}{s_j^2}}$, and d is the distance between the centre of ith source point and the lattice-node n.

The sizing function due to skeleton, curvature, etc. are combined and smoothed to meet the sizing requirements given in Section 3. Minimum of the interpolated size due to each type of source point determines the size at a lattice-node as shown in Figure 7. The maximum and minimum sizes given as user input are used to limit the sizing function. Note that individual sizing functions may not cover all the VISITED_INT and VISITED_EXT lattice-nodes. At these lattice-nodes, the average of known sizes of adjacent lattice-nodes is used. Smoothing techniques are applied iteratively to eliminate the abrupt gradient present in the minimum of all the sizing function and to satisfy α-Lipschitz criterion.

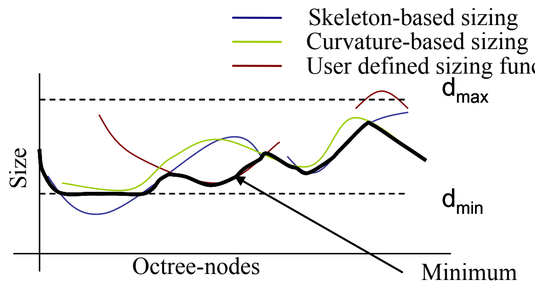
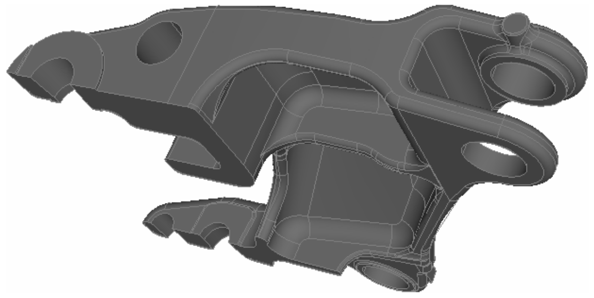
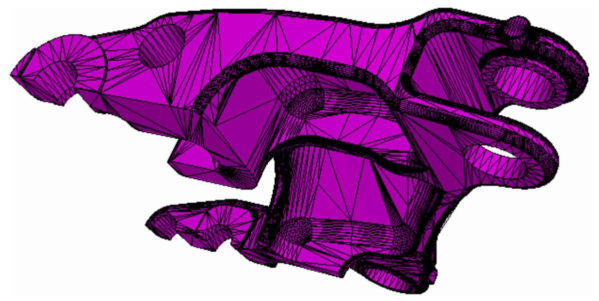
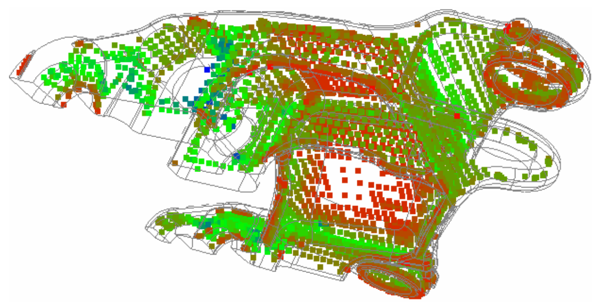


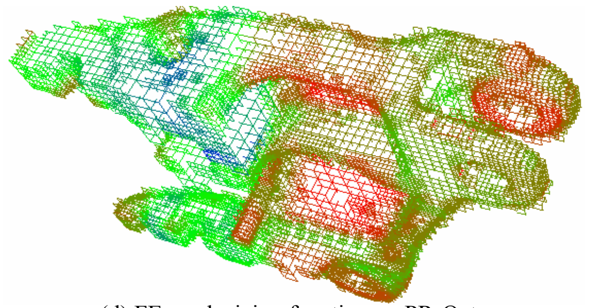
Figure 7 *Minimum of all sizing functions*

Tri-linear interpolation is used to calculate the final mesh size at a point, **p** using lattice-nodes of the cell containing **p**. One advantage of storing sizing function on an octree over background mesh [OS97] is that the query time during mesh generation is O(max_depth).


(a) Input CAD model


(b) Graphics facets


(c) Trimmed skeleton points


(d) FE mesh sizing function on PR-Octree

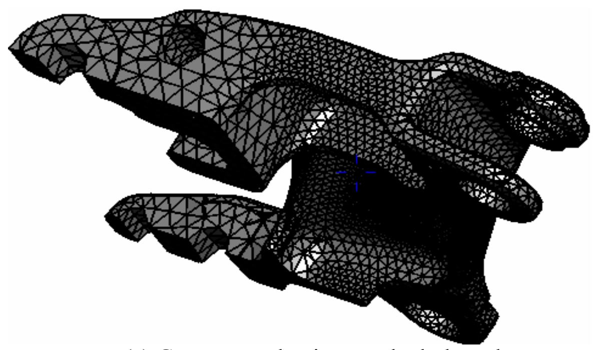

(e) Geometry-adaptive tetrahedral mesh

Figure 8 *Stages of the proposed approach*

8. Results and discussion

The algorithm described in this paper has been implemented in C++ in CUBIT, mesh-generator software of Sandia National Laboratories. The algorithm has been tested on many industrial models and one such model with the stages involved in the current approach is shown in Figure 8. The mesh sizing command takes the arguments, min_depth (int=4), max_depth (int=7), overall_scale (0.0 to 1.0 = 0.75), and interpolation_scheme (0 to 5=2). The numbers inside the parenthesis show the default values.

The min_depth and max_depth can be used to control the accuracy and density of the skeleton. From Table 1, it is clear that as the max_depth increases, the accuracy and density of the skeleton increases. Figure 9 shows one such skeleton with 8341 points generated from a PR-Octree of min_depth, 4 and max_depth, 7 on a model consisting of 5,038 facets. It is seen that octree-based skeleton captures intricate and fine features.

The skeleton-based mesh sizing function has an added advantage in layered mesh generation. In mold flow simulation of thin section solids, layers of elements are preferred along the flow direction. The proposed approach can create single and multiple layered meshes by controlling the overall_scale. Figure 9 shows the multi layer mesh generated by setting overall_scale equal to 0.3.

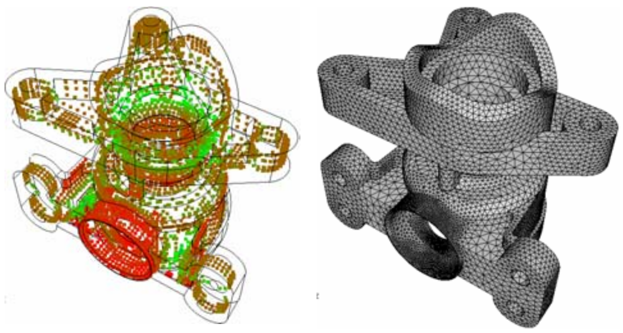


Figure 9 *Skeleton and multi layered mesh*

Table 1 *Density and accuracy of skeleton shown in Figure 9*

max_depth	Time (sec)	No. of points	Maximum radius	Minimum radius
6	2.031	4,332	9.728348	0.554324
7	3.172	8,341	9.724072	0.50
8	3.344	10,786	9.721934	0.352418

9. Conclusion

A new octree-based algorithm for the generation of a disconnected skeleton has been presented and an application of such a skeleton in generating a 3D FE mesh sizing function is also presented. The proposed approach is more accurate and efficient than the traditional voxel-based approaches in generating the skeleton of industrial models containing fine features. The skeleton-based mesh sizing function captures proximity and feature size accurately with smooth gradients in mesh size, and is a good tool for layered meshing.

References

[Blu67] H. BLUM, A Transformation for Extracting New Descriptors of Shape, *Models for the Perception of Speech and Visual Form, Cambridge, MA The MIT Press,* (1967), pp. 326-380.

[CCS97] A. CUNHA, S. A. CANANN, S. SAIGAL, Automatic Boundary Sizing For 2D and 3D Meshes, *AMD Trends in Unstructured Mesh Generation, ASME*, (July 1997) vol. 220, pp. 65-72.

[Dan80] P. E. DANIELSSON, Euclidean Distance Mapping, *Computer Graphics and Image Processing*, (1980) vol. 14, pp. 227-248.

[GP92] H. N. GURSOY, N. M. PATRIKALAKIS, An Automatic Coarse And Fine Surface Mesh Generation Scheme Based on MAT Part I: Algorithms, *Engineering With Computers*,(1992), vol. 8, pp. 121-137.

[LLC92] L. LAM, S. W. LEE, C. Y. CHEN, Thinning Methodologies - A Comprehensive Survey, *IEEE Trans. PAMI*, (1992), vol. 14, pp. 869-885.

[OS97] S. J. OWEN, S. SAIGAL, Neighborhood Based Element Sizing Control for Finite Element Surface Meshing, *Proceedings, 6th International Meshing Roundtable*, (Oct 1997), pp. 143-154.

[QRP*01]W. R. QUADROS, K. RAMASWAMI, F. B. PRINZ, B. GURUMOORTHY, Automated Geometry Adaptive Quadrilateral Mesh Generation using MAT, *Proceedings of ASME DETC*, (Sept 2001).

[QSO03] W. R. QUADROS, K. SHIMADA, S. J. OWEN, Skeleton-Based Computational Method For Generation Of 3D Finite Element Mesh Sizing Function, *7th US National Congress on Computational Mechanics - 4th Symposium on Trends in Unstructured Mesh Generation*, (July 27-31, 2003).

[Rag93] I. RAGNEMALM, The Euclidean Distance Transformation in Arbitrary Dimensions., *Pattern Recognition Letters*, (1993), vol. 14, pp. 883-888.

[Sam89] H. SAMET, Neighbor Finding in Images Represented by Octrees, Computer Vision, *Graphics, and Image Processing*, (1989), vol. 46, pp. 367-386.

[Sam95] H. SAMET, Spatial Data Structures, *in Modern Database Systems: The Object Model, Interoperability, and Beyond, W. Kim Ed. Addison-Wesley/ACM Press*, (1995) pp. 361-385.

[She88] M. S. SHEPHARD, Approaches to the Automatic Generation and Control of Finite Element Meshes, *Applied Mechanics Review*, (1988), vol. 41, pp. 169-185.

[SB99] K. SIDDIQI, S. BOUIX, The Hamilton-Jacobi Skeleton, *International Conference of Computer Vision*, (1999), pp. 828-834.

[SNT*92]V. SRINIVASAN, L. R. NACKMAN, J. M. TANG, S. N. MESHKAT, Automatic Mesh Generation using the Symmetric Axis Transformation of Polygonal Domains, *Proc. IEEE*,(Sept 1992), vol. 80(9), pp. 1485-1501.

[TKG*03]K.-F. TCHON, M. KHACHAN, F. GUIBAULT, R. CAMARERO, Constructing Anisotropic Geometric Metrics Using Octrees and Skeletons, *12th International Meshing Roundtable*, (2003), pp. 293-304, Sept 14-17.

[ZW93] Y. Y. ZHANG, P. S. P. WANG, Analytical Camparison of Thinning Algorithms, *Int. Journal of Pattern Recognition and Artificial Intelligence*, (1993) vol. 7, pp. 1227-1246.

[ZBS02] J. ZHU, T. BLACKER, R. SMITH, Background Overlay Grid Size Functions, *Proceedings of 11th International Meshing Roundtable*, (2002), pp. 65-74, Sept 15-18.