ELSEVIER

Computer-Aided Design 33 (2001) 221–232

COMPUTER-AIDED DESIGN

www.elsevier.com/locate/cad

Feature based hex meshing methodology: feature recognition and volume decomposition ☆

Y. Lu[a,1,*], R. Gadh[a,2], T.J. Tautges[b,3]

[a]*I-CARVE Lab, University of Wisconsin-Madison, Madison WI, USA*
[b]*Sandia National Laboratories, USA*

Abstract

Considerable progress has been made on automatic hexahedral mesh generation in recent years. A few automated meshing algorithms (e.g. mapping, submapping, sweeping) have proven to be very reliable on certain classes of geometry. While it is always worth pursuing general algorithms viable on arbitrary geometry, a combination of the well-established algorithms is ready to take on classes of complicated geometry. By partitioning the entire geometry into meshable pieces matched with appropriate meshing algorithms, the original geometry becomes meshable and may achieve better mesh quality. Each meshable portion is recognized as a meshing feature. This paper, which is a part of the feature based meshing methodology, presents the work on shape recognition and volume decomposition to automatically decompose a CAD model into hex meshable volumes. There are four phases in this approach: Feature Determination to extract decomposition features; Cutting Surfaces Generation to form the cutting surfaces; Body Decomposition to get the imprinted volumes; and Meshing Algorithm Assignment to match volumes decomposed with appropriate meshing algorithms. This paper focuses on describing feature determination and volume decomposition; the last part has been described in another paper. The feature determination procedure is based on the CLoop feature recognition algorithm that is extended to be more general. Some decomposition and meshing results are demonstrated in the final section. © 2001 Elsevier Science Ltd. All rights reserved.

Keywords: Feature recognition; FEA; Solid modeling; Hexahedral meshing; Volume decomposition

1. Introduction

The Finite Element Analysis (FEA) technique is widely used for parts prototyping and design verification. Meshing, the procedure to discretize geometry for the FEA model, tends to be time-consuming and error prone. The automation of mesh generation can immediately speed the product design cycle with faster design verification. Using hexahedral meshes for analysis is preferable to tetrahedral mesh in some applications [1]. Many researchers have been investigating algorithms to automate the hexahedral meshing procedure to get all-hexahedral elements models [2]. Although significant progress has been made over the years, the completely automated method that is demanded by designers who practice FEA is not yet available due to the general difficulty of filling hexahedral elements into 3D space.

Meshing is a process of spatial decomposition. During the procedure of meshing, a physical 3D space is decomposed into small elements with required topology and geometry constraints. Available techniques for generating hexahedral meshes can be characterized by the constraints on the region being meshed and the resulting mesh quality; typically, the more constrained an algorithm's domain of applicability, the higher the quality of the mesh generated. The algorithms that are most widely used for generating hexahedral meshes are mapping/submapping, meshing primitives, and sweeping.

The mapping/submapping algorithms [3,4] generate structured meshes by first identifying the logical "corners" of the mesh, then placing the mesh nodes in space using interpolation from the boundary mesh. These algorithms work best on "blocky" volumes, and are computationally inexpensive. However, their domain of applicability is

☆ Work funded in part by Sandia National Laboratories. Sandia is a multi-program laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94AL85000.

* Corresponding author. Present address: Mechanical Engineering Department, University of Wisconsin-Madison, 1500 Engineering Drive, Madison, WI-53706, USA. Tel.: +1-608-265-3953; fax: +1-608-262-4814.

E-mail addresses: yong@smartcad.me.wisc.edu (Y. Lu), gadh@engr.wisc.edu (R. Gadh), tjtautg@sandia.gov (T.J. Tautges).

[1] http://icarve.me.wisc.edu/~yong
[2] http://icarve.me.wisc.edu/~gadh
[3] http://endo.sandia.gov/9225/Personnel/tjtautg/

quite restricted, and in some cases applying these algorithms to arbitrary volumes requires user interaction to identify corners in the map/submap.

Meshing primitives have also been widely used, either explicitly through implementation in a meshing tool [5–7], or implicitly by the user decomposing an arbitrary region into mappable regions using well-known strategies (e.g. u-grid, c-grid, etc.). These techniques expand slightly the domain of applicability of mapping and submapping, but otherwise are very similar in mesh quality and the required level of user interaction.

Various forms of the sweeping algorithm have been proposed [8–11]; sweeping generates mesh by extruding one or more surface meshes into the third dimension. While sweeping has been used extensively to generate very complex meshes, its domain of applicability is restricted to volumes that have a logical extrusion direction. The identification of such volumes, along with the source and target surfaces of the extrusion, has been automated [12]; however, the domain of applicability is still restricted.

There are also many algorithms being researched whose goal is to generate a hexahedral mesh fully automatically for arbitrary volumes. Whisker weaving [13,14] is an advancing front algorithm that generates hex mesh connectivity in the dual space, and then places the mesh nodes in physical space. The grid based method described in Refs. [15,16]constructs a structured grid in the interior of the body and then generates elements in the boundary region by using an isomorphism technique. Various algorithms have also been proposed for generating hex-dominant meshes, where hex elements are placed near the boundaries of the solid and tetrahedra are used to fill any remaining space. Notable among these approaches are H-Morph [17] and Plastering [18]. Although substantial progress has been made on these all-hex and hex-dominant algorithms, none of them is currently robust enough to be used as the only generation method in typical models.

Thus, the current state of the art for generating all-hexahedral meshes for complicated geometries requires the decomposition of a model into smaller pieces, each of which can be meshed with one of the algorithms described earlier. This "divide and conquer" approach, although time-consuming, has been shown to be quite effective for generating large hexahedral meshes [19,20]. This leads to the question of whether the process of decomposing a model into pieces, each of which can be meshed with algorithms such as sweeping and mapping, can itself be automated.

There has been some research done on decomposition-based approaches to quad and hex meshing. Blacker proposes a knowledge system approach that automates two-dimensional quadrilateral mesh generation [21,22]. In this approach, a 2D decomposition process is developed to subdivide the geometry into "mappable" sub-regions. In 3D, the problem of decomposition and meshing becomes more complicated, with the identification of primitive regions and the construction of cutting surfaces used in the decomposition being significantly more difficult.

Armstrong et al. suggest using the medial axis transform (MAT) technique for decomposition and meshing [5,6,23] The medial surface is described by the center of an inscribed sphere with maximum radius rolling through the model. Armstrong, Price, et al. use the medial surface to guide the decomposition of the model, and the midpoint subdivision technique is used to mesh the resulting primitive volumes. The computation of a medial surface is not a trivial task and, so far, sufficiently reliable and practically effective algorithms for generating medial surfaces are not available yet. This technique also has difficulty in decomposing completely arbitrary volumes into recognizable primitives.

Sheffer et al. [24] propose to use the Embedded Voronoi Graph for guiding decomposition. Embedded Voronoi Graph is used here to approximate the Voronoi diagram and the medial surface. Although this approach shows some promise, it has not been demonstrated for more complicated models, and might also result in over-decomposition of bodies.

Shih and Sakurai present a mesh generation tool via swept volume decomposition [25]. The algorithm works on certain geometry that the paper demonstrates. However, the process of swept volume decomposition consists of swept volume generation and needs many regular boolean operations, which may make the algorithm computationally expensive.

There are also some decomposition algorithms proposed in the domains of computational geometry and feature recognition. Chazelle introduces the technique to decompose non-convex objects into convex components [26]. The disadvantage of this approach is that the shapes of volumes decomposed can be arbitrary and the algorithm doesn't recognize the rationale of finite element meshing. Woo suggests the process called ASV (Alternating Sum of Volumes) to extract a unique series expansion of the object in terms of convex components with alternating signs [27]. The disadvantages of ASV are that the computation of convex hulls is expensive and the decomposition may result in awkward shapes. Sakurai proposes a feature extraction method based on so called "maximal volumes" [28]. In this method, artificial edges are generated by intersecting surfaces and minimal cells are then formed by grouping edges and computing faces. Different combinations of these cells form maximal volumes. The extensive intersection of surfaces to generate minimal cells, and also the combination of minimal cells to form maximal volumes, are very computationally expensive. In the results demonstrated, the types of geometry this algorithm can handle are limited.

Feature recognition (FR) is the process of extracting design or manufacturing information from a solid model. Extensive research has been performed in the FR field [29–32]. This paper employs a FR technique to guide decomposition for hexahedral meshing. There are four phases in this approach: Feature Determination to extract decomposition features, Cutting Surfaces Generation to form the cutting surfaces, Volume Separation to generate separate volumes, and Meshing Algorithm Assignment to match volumes decomposed with appropriate meshing algorithms. This paper focuses on describing feature determination and volume decomposition; the last part has been described in another paper [12]. The paper concludes with some decomposition and meshing results.

2. Volume decomposition and meshing

Consider how a human expert hex-meshes a CAD model. If the geometry is too complicated to mesh automatically, it is decomposed into smaller pieces, each of which is meshable using some known automatic meshing patterns. A great deal of meshing knowledge is usually employed to help guide the decomposition process. Issues which are treated implicitly by expert users are where to place decomposition surfaces, whether the cutting leads to simpler, meshable volumes, and whether there exists a better cutting solution for getting higher mesh quality in less time. This process is often heuristic-based, with few deterministic rules available.

In this research, we seek to identify heuristic rules that can be used in the decomposition process, where heuristics are based solely on geometric characteristics of the model, independent of any particular meshing algorithm. In general, the goal of this strategy is to identify and cut off protrusion features in the model. While some general knowledge about meshing algorithms could be used to identify other possible locations for decomposition, we choose not to use that knowledge until the generic rules yield no further decompositions. Any remaining unmeshable pieces are left to the user for further, manual decomposition.

This approach has several advantages. Starting with a strategy which is independent of meshing algorithms reduces the number of special cases that are used in the decomposition process, which increases both the decomposition speed as well as the number of parts to which this strategy can be applied. Allowing the use of some meshing algorithm knowledge incorporates some of those special cases, which can be helpful for decomposing parts with specific well-known geometry. Since the goal is to decompose the part into as many meshable pieces as possible, rather than into 100% meshable pieces, the process can serve as one of many tools in the decomposition process, rather than the only tool for decomposition. From this perspective, this tool reduces the amount of geometry the user has to mesh by hand, rather than meshing a given model fully automatically. This widens the domain of applicability of this decomposition strategy, and reduces the model complexity experienced by the user.

We have Ω as the operator representing the entire decomposition procedure to obtain meshing features, which are volumes matched with automatic hexahedral meshing algorithms. B represents the original body. S is the set of M volumes left at the end of the decomposition. Then we have:

$$\Omega(B) = S \tag{1}$$

$$s_i \in S \qquad i \in [1, M] \tag{2}$$

Then

$$s_i \subseteq B, \qquad i \in [1, M]$$

$$s_i \cap s_j = \phi \qquad i,j \in [1, M] \tag{3}$$

$$B = i = 1 \sum_{i=1}^{M} S_i$$

where $\sum$ is the operator of exclusive addition.

If Ψ is the operator of the meshing algorithm assignment, then

$$\Psi(S) = T \tag{4}$$

$$t_i \in T \qquad i \in [1, N] \tag{5}$$

where T is the set of volumes having a valid meshing scheme assignment.

It is ideal if the automatic decomposition process yields

$$B = \sum_{i=1}^{N} t_i, \qquad t_i \in T \tag{6}$$

or, equivalently,

$$T = S \tag{7}$$

That means all the volumes decomposed can be matched with an automatic hex meshing algorithm. Certainly, Eqs. (6) and (7) will always be true if we relax Ω to include manual decomposition.

To measure the success of decomposition for meshing, we introduce ϕ as the automatic meshing ratio. ϕ can be defined based on either the volume ratio (ϕ_v) or the number ratio (ϕ_n), where

$$\phi_v = V(T)/V(B) = \sum_{i=1}^{N} V(t_i)/V(B) \qquad \phi_n = N/M \tag{8}$$

where V is the operator to calculate the volume. Both measures are useful for measuring the success of decomposition. The difficulty of meshing the remaining volumes, i.e. S−T, can depend on either or both, depending on the geometry.

Besides improving the meshability of a model, the quality of mesh can be enhanced and the (computational) meshing time can be reduced. The complicated meshing algorithms that treat more versatile geometry tend to be more computationally expensive than simpler ones. For example, the meshing speed of sweeping is approximately one tenth the speed of mapping or submapping [29,34]. Through decomposition, the geometry and topology of the volumes usually

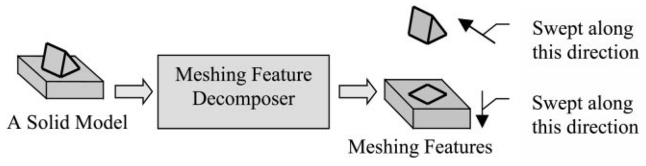

A Solid Model
Meshing Feature Decomposer
Meshing Features
Swept along this direction
Swept along this direction

Fig. 1. Meshing feature decomposer..

becomes simpler, thus it is possible for more of them to be meshed with mapping/sub-mapping, which are less expensive and produce better quality mesh. Decomposition allows the use of the fastest possible algorithm given the geometric constraints of the pieces resulting from the decomposition. The same principle will hold once fully automatic hex meshing algorithms are part of Ψ.

3. Feature recognition approach

Generally, the meshing feature decomposer described in this paper takes a solid model as the input and outputs a set of volumes, some of which are recognized meshing features (i.e. sub-volumes matched with appropriate meshing algorithms). Among those are sweeping features, mapping/sub-mapping features, primitive features, etc. The procedure is shown in Fig. 1.

The features defined for the design and manufacturing domains may not be appropriate for meshing. “Bosses”, “ribs”, “slots” and “key-ways” are defined for design and manufacturing functionalities. In meshing, geometries are regarded as the same feature as long as the same algorithm can be used to mesh them. Only an exclusive addition operation is well recognized and implemented for set operation to join individual lumps of meshes. Negative volumes or a set of volumes with overlapping regions have no direct value to meshing. Since only positive features are meaningful in the meshing domain, FR-based techniques applied to meshing concentrate on protrusion features, which tend to be bounded at least partly by concave transition zones.

Among the many feature recognition techniques, one based on CLoops [32–37] is chosen as the foundation of the feature-based decomposition technique employed in this research. After identifying possible features, cutting surfaces are generated based on the CLoops. These cutting surfaces are used to decompose the model, thereby slicing off the identified features. The process is applied recursively to the resulting pieces until no further decompositions are made. This section focuses on describing the extended CLoop FR technique.

3.1. CLoop definition

Based on the dihedral angle along the edge, an edge can be classified as Concave Edge (if the dihedral angle $>180°$), Convex Edge (if the dihedral angle $<180°$), Neutral Edge (if the dihedral angle $= 180°$) and Hybrid Edge (if the dihedral angle varies around 180° along the edge) [36,37]. Gadh and

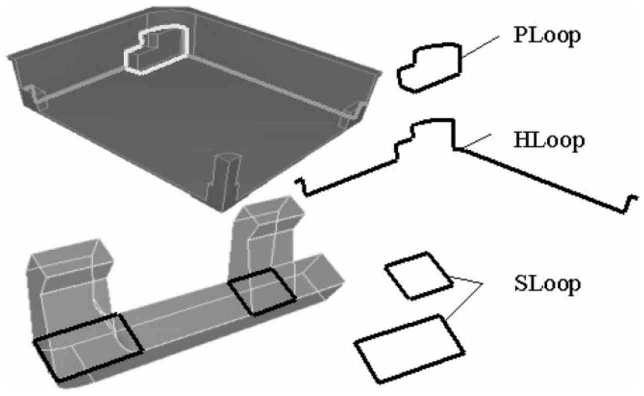


Fig. 2. Ploop, HLoop and Sloop.

Prinz suggested an abstraction of CLoop for feature recognition, where CLoop was introduced as a closed set of linked edges with the same convexity [32]. The CLoops, combined with rules, are used to extract both protrusive features like ribs, bosses, etc. and depressive features like holes and slots. For the purpose of decomposition, Liu and Gadh extended the definition of CLoop to be an open or closed link of edges by introducing PLoop and HLoop [35,36]. The concept of PLoop and HLoop is relaxed further and SLoop, a type of CLoop with mixed convexity, is introduced to accommodate more classes of features such as fillet shapes [37].

Based on the convexity of edges, a CLoop is classified as [36,37]:

- *Pure CLoop.* A Pure CLoop (PLoop for simplicity) is a closed link of edges of the same convexity. It can be further classified as Pure Concave CLoop, Pure Convex CLoop, or Pure Neutral CLoop.
- *Pseudo CLoop.* A Pseudo CLoop (SLoop for simplicity) is a closed link of edges of mixed convexity. The edges in the CLoop can be neutral, concave, convex or hybrid.
- *Hybrid CLoop.* A Hybrid CLoop (HLoop for simplicity) is an *open* link of edges of the same or mixed convexity.

Fig. 2 illustrates the three different kinds of CLoop. The inclusion of SLoop and extension of PLoop and HLoop allow for the definition and extraction of more kinds of decomposition features that cannot be determined by the previous definitions of PLoop and HLoop. In practice, only a small set of SLoop or HLoop is good for decomposition. We set a threshold of the dihedral angle for those edges that are convex or hybrid to be qualified as edges in a SLoop or HLoop. Usually, convex or hybrid edges in a SLoop or a HLoop are restrained to be edges with dihedral angle close to 180°. In addition, a SLoop or HLoop is required to have one concave or neutral concave edge inside the link at least.

CLoops have been used to define four classes of shapes: Protrusion, Blind Depression, Through Depression and Bridge [36]. Meshing features are protrusive volumes, therefore, shapes such as protrusion and bridge are recognized for decomposition. Negative features are not decomposed for meshing since it is difficult to implement general set operation on negative meshing primitives.

3.2. CLoop determination

The extraction of PLoops, HLoops, and SLoops from a BREP solid is a graph searching procedure. Both PLoops and SLoops are closed links of edges, while HLoops are open links of edges. Specially, the algorithms used to identify PLoops and SLoops are very similar except for the different requirement on convexity.

We define G(e) as the edge graph of model B. A CLoop is formed by linking edges sharing the required convexity in the Graph G.

3.2.1. PLoop and SLoop identification

The algorithm used to identify PLoops in a BREP model is shown in Table 1. Clearly, this is a depth-first searching algorithm. In essence, the procedure to determine SLoop is the same as PLoop. The only difference is that there is a different requirement on edge convexity in the loop. A SLoop is formed by edges with varying convexity.

SLoops can define some features that are impossible to define using CLoops and HLoops. For example, they can be used to identify a fillet-type feature. As shown in Fig. 2, two SLoops are extracted, one of which bounds a fillet-type shape.

Table 1
PLoop finding algorithm

```
Every edge can be marked with a flag "unmarked", "in progress" or "finished"
L is a list of edges
λ is a list of edges which represents a CLoop
Φ holds a list of CLoops in the model M

Begin with an edge graph G(e)
    Start with initializing all nodes of concave edges by marking them
    with flag "unmarked"
    Initialize an empty list Φ
    Initialize an empty ordered list L
    For every node of a concave edge e that is "unmarked"
        Do depth-first search by calling DFS (e)
        Dispose of the list L
    End For
End

Function DFS (n)
    Mark n "in progress"
    Put n in list L
    For every successive node of concave edge m of n (successive nodes
    are all the adjacent nodes of node n except the node preceding n in the
    traversal path)
    Begin
        If m is "unmarked", then do DFS (m)
        Else if m is marked with "in progress" then
            Initialize an empty list λ
            Copy nodes in L from m through n into list λ
            Put λ into list Φ
        Else if m is marked with "finished" then
            Do nothing
        End if
    End
    Mark n with "finished" flag
    Remove n from the list L
Function end
List Φ holds all PLoop in the model M
```

3.2.2. HLoop identification

A HLoop is an open link of edges and needs to be completed to form a closed cutting loop for decomposition. The original edges in a HLoop must be combined with neutral edges, and possibly with some edges from other CLoops, to form a new PLoop. There are three steps in HLoop identification: finding the open link of edges (i.e. the HLoop), traversing and generating neutral edges on the lateral faces between the two ends of the HLoop, and combining the edges to form a new SLoop.

The first step of getting an open link of edges for HLoop has similarity to the determination of PLoop and SLoop. For PLoop and SLoop, a CLoop is formed when the traversal path ends up with a cycle. For HLoop, it is formed when a non-cyclic traversal path ends at a vertex where no more successive edges with required convexity can be added into that path. Usually, neutral edges generation and cutting surface fitting for HLoop is done at the same time. For convenience, the determination of HLoop will be completed in the later section, where cutting surface generation for HLoop is detailed.

3.3. CLoop pool

All instances of PLoop, SLoop and HLoop constitute a CLoop pool. Fig. 3 shows the CLoop pool for an example model (note that some HLoops are not displayed in this example). These CLoops can be classified according to their spatial relationship; four possible relationships are observed:

Joint: CLoops (excluding HLoops) share edges or vertices.

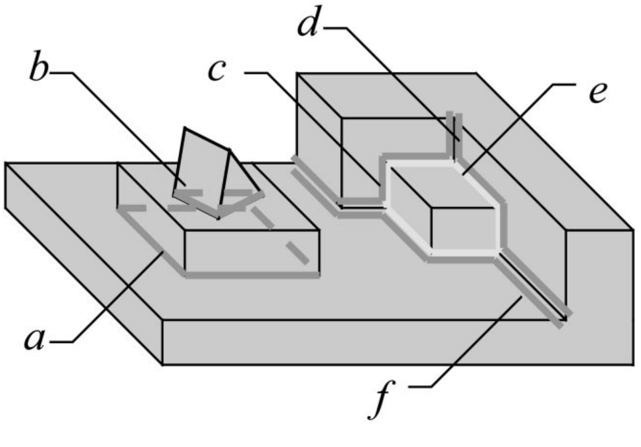


Fig. 3. CLoop pool (partial).

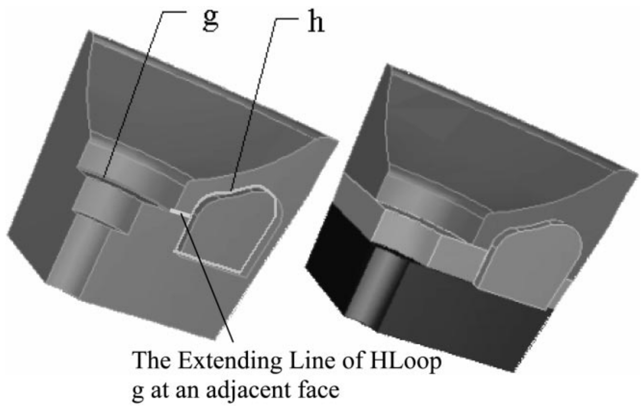


Fig. 4. Intersection between a PLoop and a HLoop.

For example, in Fig. 3, CLoops c, d, e and f are "joint" to one another.

Intersected: It is between a HLoop and another CLoop (PLoop, HLoop or SLoop). The involved HLoop might be extended to check the relationship. An "intersected" relationship is identified when a HLoop or its extension is intersected with another CLoop. Fig. 4 shows an example of intersecting between a HLoop and a PLoop (the HLoop is extended so as to check the relationship).

Coplanar: Two CLoops share the same set of extending surfaces (see Fig. 5(a) for "coplanar PLoops" and (b) for "coplanar" HLoops).

Disjoint: Two CLoops don't have any of the relationships above. In the example of Fig. 3, CLoop a and b are disjoint and either of them is disjoint to any one of c, d, e and f.

Fig. 3 shows that CLoops are clustered. CLoops c, d e and f form a cluster with four members. CLoops a and b each form a cluster with a single member. CLoops in different clusters may be grouped together for constructing a set of cutting patches, while CLoops in one cluster may be sorted to generate cutting surfaces separately. The concept of a separator that is detailed in the following section is introduced to facilitate the grouping and sequencing of CLoops.

4. Constructing cutting surfaces

4.1. Separators

PLoop, SLoop and HLoop are retrieved from the edge graph and a CLoop pool is built. CLoops in this pool are then sorted and analyzed to form the pool of cuttable separators. A separator is one or more loops which can bound a cutting surface (a cutting surface may be made up of a set of adjacent cutting patches). Separators are formed from one or more CLoops; the spatial relationship of CLoops is an important factor in choosing which CLoop(s) will form the next separator. There are typically many possible combinations of CLoops to form separators, and the choice of separators strongly influences the quality of the decomposition. The following heuristic rules for generating separators have been investigated:

- "Coplanar" CLoops possibly form a single separator. Fig. 5 gives two examples of them, one for PLoops (Fig. 5a), the other for HLoops (Fig. 5b).
- Among "joint" CLoops, only one of them is chosen as a separator in a single cutting stage. CLoop c, d, e and f are joint to one another, as shown in Fig. 3. For the example in Fig. 6, CLoop f is chosen to be a separator at Stage I. Other CLoops like c, d and e are made cuttable during the following stages.
- Among "intersected" CLoops, a PLoop or SLoop is chosen over the involved HLoop as a separator in a single cutting stage in order to have fewer partitions. Fig. 4 gives the decomposition result by choosing PLoop h over HLoop g as the separator.
- A disjoint CLoop may serve as a separator. PLoop a and PLoop b of the example in Fig. 3 above can each form a separator. Their decomposition result is shown in Fig. 6.

All the available separators form a pool of choice at each cutting stage. For non-binary decomposition, each separator can be used simultaneously to generate a cutting surface,

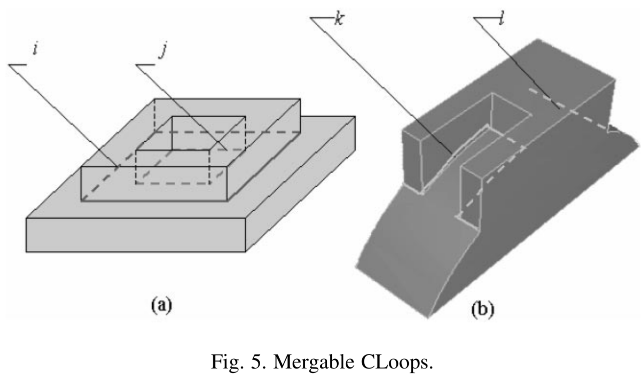


Fig. 5. Mergable CLoops.

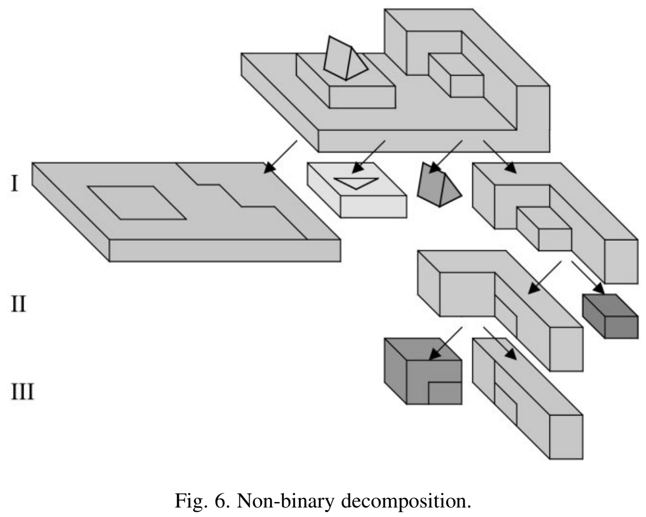


Fig. 6. Non-binary decomposition.

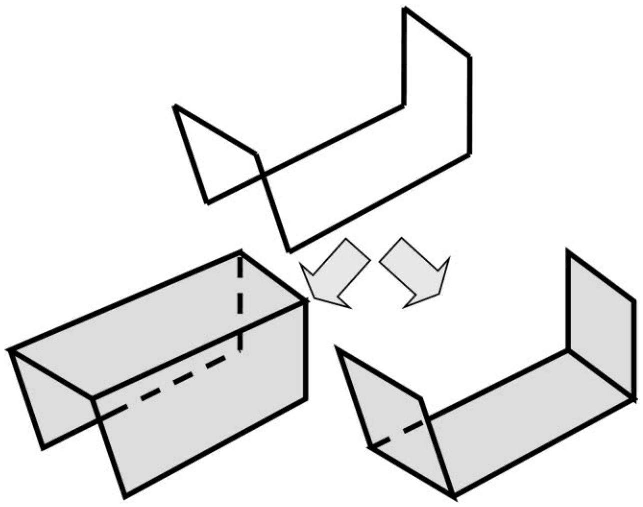


Fig. 7. Two different sets of fitting surfaces for a PLoop.

and the model is cut with many surfaces at one step. More than two volumes are obtained per step from non-binary decomposition. For binary decomposition, only one separator is used at each decomposition step. The old "un-cuttable" separators may become "cuttable" on new volumes and new separators could be uncovered from them. The new ones together with old ones form the pool where separators for the next cutting are chosen. There can be many criteria for selection of the next separator to construct a cutting surface with. For example, the simplest CLoop serves as the separator. Or, choose the longest CLoop as the separator instead.

The rules listed above, as well as the priorities used to choose between them at any stage, are heuristic in nature. Further investigation is ongoing. Besides these rules, the use of meshing knowledge to judge the prioritization of CLoops is being investigated.

4.2. Cutting surface formation

Cutting surfaces are constructed by fitting a surface over a separator. In many cases, there can be several good candidate surfaces to fit over a given set of edges. Fig. 7 shows two reasonable fitting results for a single CLoop. Different cuttings can result in significantly different geometry and can have a heavy impact on meshing. For example, in Fig. 8, the decomposition of (b) results in swept volumes with constant cross sections, while the decomposition of (a) leads to two more complicated volumes with Bspline patches, where the mesh quality deteriorates and usually more time is needed to mesh them.

The proper choice of cutting surface is not determined solely on the basis of the feature being cut from the remaining part. In the decomposition shown in Fig. 9(a), the lower portion of the part can be swept and will yield a good quality mesh; however, this decomposition leaves an acute angle on the upper portion, which will result in poor quality mesh in

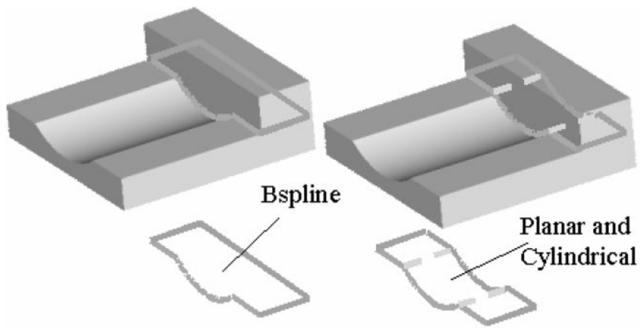


Fig. 8. A case favorable for extending.

that region. The decomposition shown in Fig. 9(b) is much more acceptable. The upper portion will be swept with much better mesh quality, at a modest penalty to mesh quality in the lower portion of the model.

Various techniques are used for fitting surfaces over different types of CLoops in this research; these techniques are listed in Table 2. This paper focuses on presenting progress made on "extending" algorithms: "natural extending algorithm" for HLoop and "natural fitting algorithm" for PLoop and SLoop. As the name suggests, these algorithms create cutting surfaces by extending surfaces already in the model. Avoiding the introduction of extra geometry into the model by using native geometry results in more natural-looking features, and leads to higher efficiency during the

Table 2
Types of surface fitting algorithms used for various types of CLoops.

Type of CLoop	Surface fitting algorithm
PLoop	Natural fitting, best fitting
SLoop	Natural fitting, best fitting
HLoop	Natural extending, best fitting

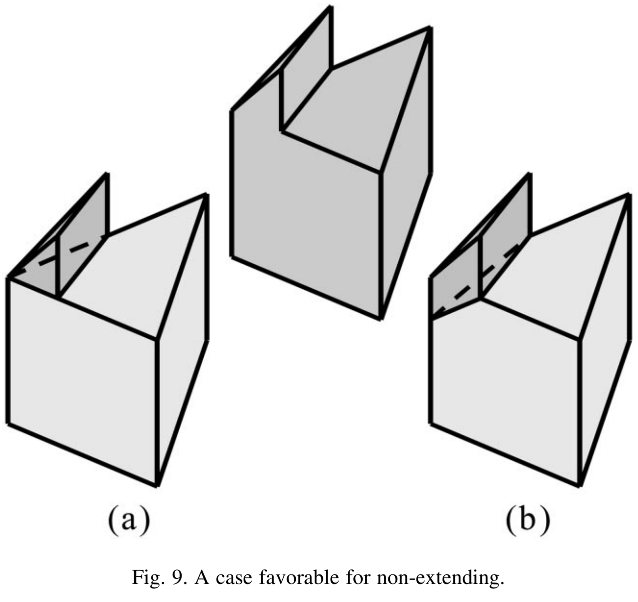


Fig. 9. A case favorable for non-extending.

decomposition process; unnecessary surface merging is also avoided. “Best fitting” is a non-extending algorithm and is addressed briefly in this paper as well.

A “naive” extending strategy can be easily envisioned: i, extending and intersecting a face adjacent to each edge in the CLoop with other faces in the model to generate artificial edges; ii, trimming the artificial edges by intersecting the artificial edges themselves; iii, sorting the trimmed edges and discarding useless edges; iv, forming loops by combining relevant artificial edges with original edges in the CLoop; v, covering the loops with correspondent surface geometries of the extending faces. Only intersections that lead to artificial edges useful for building the cutting surface are regarded as necessary intersections. The “naÏve” approach may lead to massive “useless” intersections. The two extending algorithms developed in this research are aimed at addressing this issue. “Useful” intersections are traced and performed, incrementally and locally, and “over-cuttings” can be largely avoided too.

4.2.1. Natural extending

Natural extending is used to construct cutting surfaces for separators composed of one or more HLoops. Fitting a surface over these types of separators is more complicated than fitting PLoop- and SLoop-based separators because neutral edges must be formed to close the HLoop before a cutting surface can be formed.

In a previous effort, only planar surfaces were used for generating the remaining edges and closing HLoops; this method is referred to here as a simple extending algorithm. In many cases, the cutting surfaces generated by the simple extending algorithm did not yield good results for decomposition and meshing. In those cases, a natural extending algorithm often produces better results [38]. Fig. 8(b) shows an example of decomposition generated by the natural extending algorithms.

The natural extending algorithm generates artificial edges by traversing surfaces between two ends of a HLoop. Along a HLoop, every extending face of an edge in the HLoop is extended one by one and then intersected with lateral faces (faces intersected with extending faces) continuously. In the meantime, they are self-intersected with one another. The procedure is incremental, once for one extending face. A succeeding extending face (along the HLoop) takes its turn when the projection of the intersecting edge between this extending face and the preceding extending face hits the current lateral face. As a result, a lateral face is intersected with more than one extending faces only when a succeeding extending face takes the role of current extending face and these extending faces must be adjacent (along the HLoop). Curves are computed from these intersections and then trimmed to yield artificial edges. Group the artificial edges with corresponding original edges in the HLoop to form face loops. Cutting patches are then generated by covering the loops with the geometry of the corresponding extending faces.

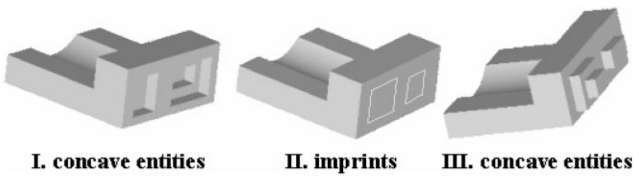


Fig. 10. Cases of “non-simple” lateral faces.

A lateral face can have multiple loops. If the intersection between a lateral face and an extending face is only involved with one face loop, the intersection is “simple” and the lateral face is a “simple” face with respect to the extending face. If all the lateral faces are “simple”, we have only one traversal thread throughout the whole procedure. A “non-simple” intersection has multiple intersections thus has multiple beginning points for the following traversal. Fig. 10 shows several cases of lateral faces that can be “non-simple”. The traversal path will be split into multiple traversal threads when entering a “non-simple” lateral face and merged thereafter when leaving the lateral face. Edges are trimmed and sorted during traversal path splitting and merging. Fig. 11 illustrates the process of splitting and merging on a “non-simple” lateral face.

4.2.2. Natural fitting algorithm

Fig. 12 shows a PLoop with six edges. Although there could be a single surface that bounds all these edges, it would not be an appropriate cutting surface for this model. In this case, the appropriate cutting surface consists of several patches and each is formed using one of the base surfaces bounding the PLoop and two edges in the PLoop. The formation of these types of cutting surfaces is referred to as the natural fitting algorithm [38], since it involves fitting one or more natural (i.e. pre-existing) surfaces over a PLoop.

The natural fitting algorithm is illustrated briefly here: for every edge *e* in the PLoop, there is a correspondent extending surface *s*. Choose an edge e_i from the PLoop. Its extending surface is s_i. The extending surface of the edge e_{i-1} that

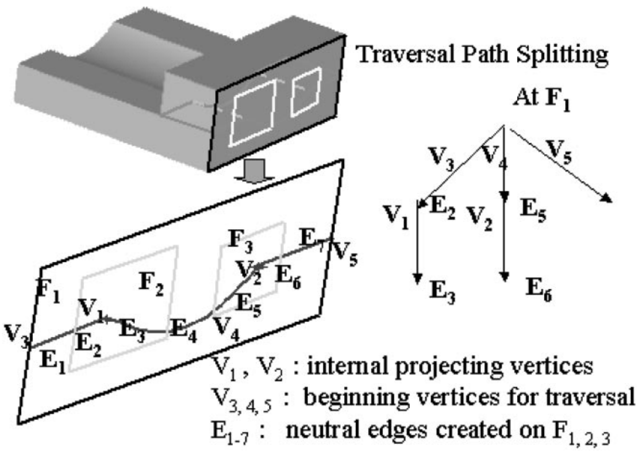


Fig. 11. Traversal path splitting and merging.

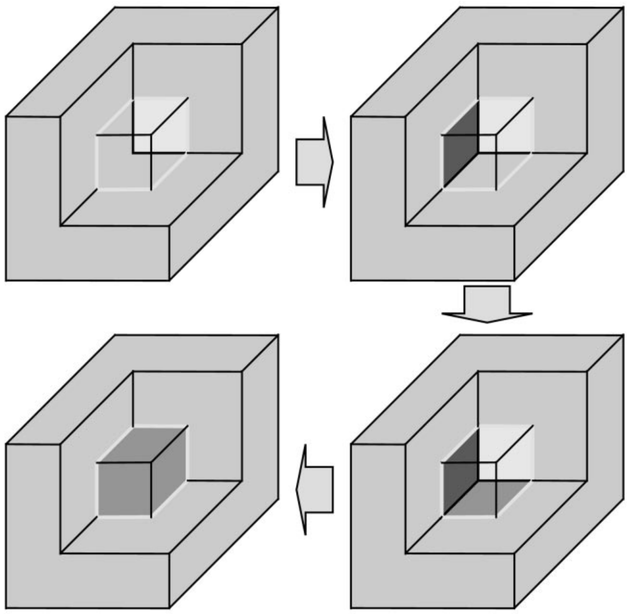


Fig. 12. Sequence of cutting patches generation with natural fitting algorithm.

immediately precedes e_i is s_{i-1} and the extending surface of the edge e_{i+1} that immediately succeeds e_i is s_{i+1}. Curves are created by intersecting s_i with s_{i-1} and s_{i+1} and then trimmed accordingly to generate artificial edges, which form a new face loop coupled with relevant edges already in the PLoop. A cutting patch is then created to cover the loop; this cutting patch shares the same geometry with the extending surface s_i. Remove the edges used to create the cutting patch from the PLoop and add the artificial edges just created into the PLoop. The above operation of creating cutting patch is resumed on the modified PLoop. The procedure is recursive until all edges in the PLoop have been consumed for new patches. In the operation of generating cutting patches, three cases are explored: (i), the neighboring extending surfaces of the current extending edge are not equivalent; (ii), the neighboring extending surfaces of the current extending edge are equivalent; (iii), the cutting surfaces of the involved extending edges are all equivalent. For the example of case (i), usually, two curves are generated by intersecting the two neighboring surface pairs. They are trimmed by self-intersecting and two relevant vertices of the extending edges.

Fig. 12 gives an example that uses the natural fitting algorithm for cutting surface generation. It shows the exact sequence of cutting patches generation. Three planar patches are formed and the cube at the corner is successfully cut off. The two sub-volumes are kept prismatic and can be easily meshed by the sweeping or the sub-mapping algorithms.

Fig. 13 shows another example of using the natural fitting algorithm. Two planar patches and one cylindrical patch are formed. The corner object is cut off smoothly, and the decomposition result is very intuitive.

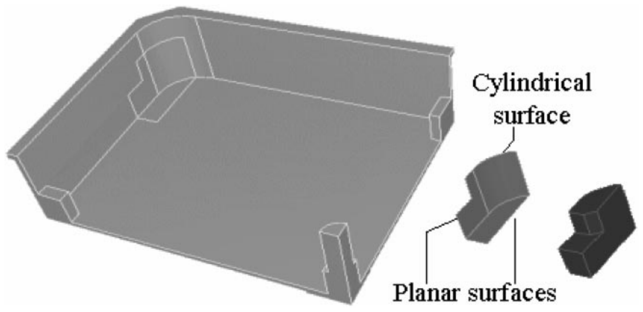


Fig. 13. Another example for natural fitting.

4.2.3. *Best fitting algorithm*

In some cases, neither the natural extending nor the natural fitting algorithms are appropriate for fitting a surface over a separator; Fig. 9 shows an example where extending algorithms are not preferable. In this case, we use a "best fit" surface to cover the separator. Among the various techniques for covering separators, Best fitting is the least reliable, as it requires the fitting of a surface over an arbitrary link of bounding edges and significantly more computation for edge and surface generation. In general, this can be difficult. Therefore, when choosing a method of those listed above to use when covering a separator, best fitting is chosen only if the other techniques fail to produce appropriate cutting surfaces. A trial version of a "best fit" algorithm is developed but the advanced investigation is still underway.

5. Implementation and results

The algorithms described above have been implemented in the CUBIT mesh generation toolkit [29]. CUBIT provides tools for importing, modifying and saving geometry in the ACIS format [39], as well as tools for generating hexahedral and tetrahedral meshes.

The geometry in Fig. 14 was decomposed then hex meshed automatically. No user interaction was required for further decomposition or for specifying meshing algorithms. Thus, both ϕ_v and ϕ_n are 100%. The scaled jacobian quality metric for this mesh [40] varies between 0.51 and 1.0, which is well within acceptable bounds. In this case, our method decomposes an unmeshable solid into a set of meshable volumes fully automatically.

Another example of how automatic decomposition is applied to the hex meshing process is shown in Fig. 15. Here, the original part is cut into 17 pieces automatically.

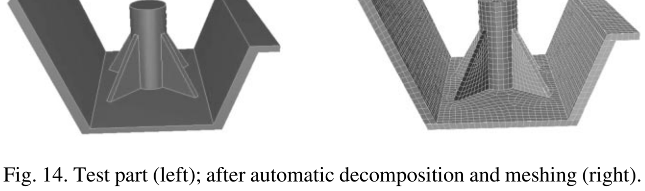


Fig. 14. Test part (left); after automatic decomposition and meshing (right).

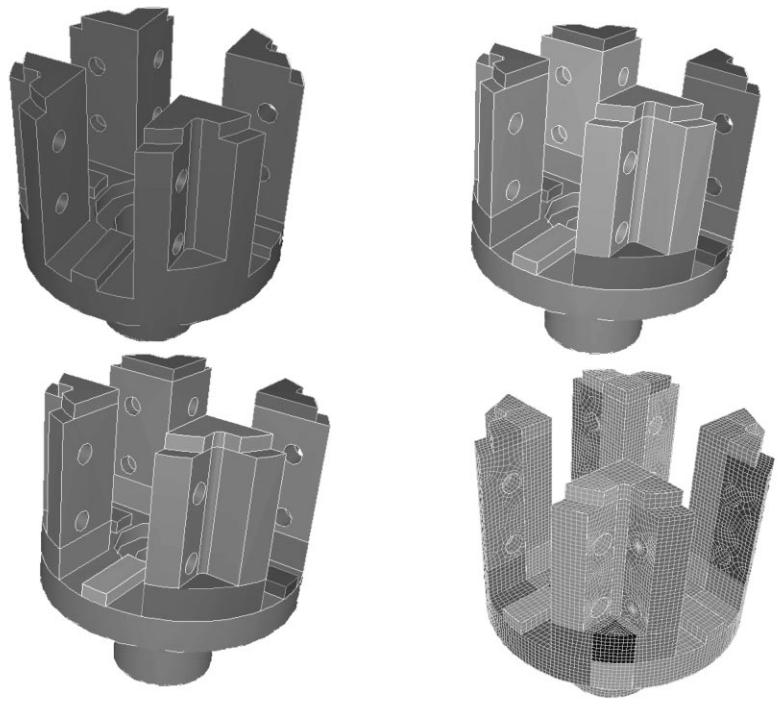


Fig. 15. Example of decomposition and meshing. Original geometry (upper left); results of automatic decomposition (upper right); volumes requiring manual decomposition (lower left); final mesh (lower right).

After identifying the unmeshable pieces, using the algorithm described in [12], 14 more cuts are performed manually; the entire model can then be meshed automatically. The resulting model and mesh are also shown in Fig. 15. In this example, ϕ_v and ϕ_n are 0.19 and 0.71, respectively. The scaled jacobian for this mesh varies between.022 and 1.0, which again is within acceptable limits.

Note that in this example, emphasis is placed on obtaining a mesh quickly rather than minimizing the number of volumes; if the latter were important, the five manual cuts could be replaced with fewer cuts, and the number of resulting volumes would be smaller. This example shows that automatic decomposition can be used even when it does not fully decompose the model into meshable pieces; rather, it performs the decompositions it can, leaving further decomposition for the user. In this context, this tool reduces the time to mesh by performing some decomposition automatically that would otherwise be done by the user manually.

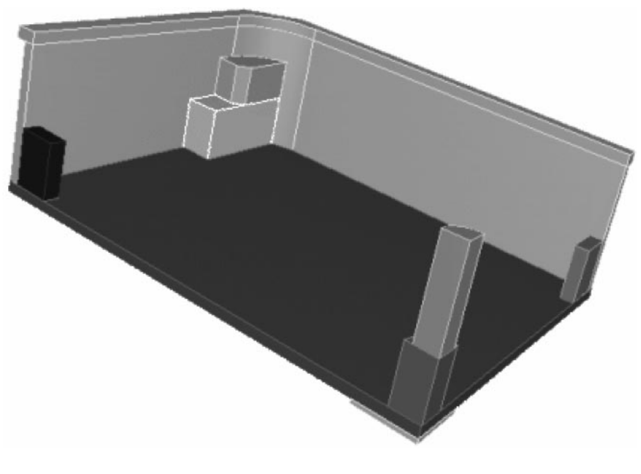


Fig. 16. A test from Sandia National Laboratories.

Figs. 16 and 17 show other parts decomposed automatically. For the part shown in Fig. 16, ϕ_v and ϕ_n are 0.48 and 0.89, respectively. For the part shown in Fig. 17, ϕ_v and ϕ_n are 0.08 and 0.56, respectively.

6. Conclusion

This paper presents the work on shape recognition and volume decomposition to automatically decompose a CAD 0model into hex meshable volumes. There are four phases in this approach: Feature Determination to extract a decomposition feature, Cutting Surfaces Generation to form the cutting surfaces, Body Decomposition to get the imprinted volumes, and Meshing Algorithm Assignment to match appropriate meshing algorithms to the volumes decomposed.

This paper employs Feature Recognition (FR) technique to guide the decomposition in an intelligent way. Some heuristic rules have been introduced to mimic the thinking of human beings when handling complicated geometry for meshing. Although there is still a lot of work to do, the methodology proves to be effective and the results are encouraging.

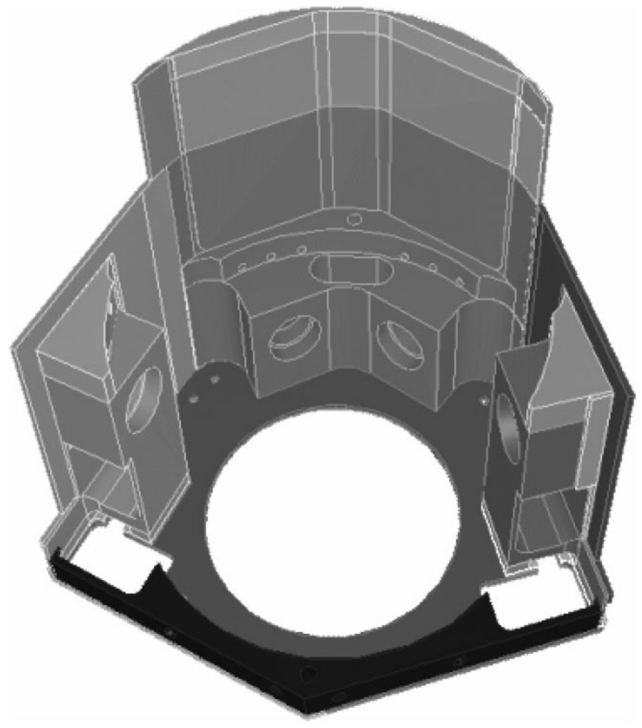


Fig. 17. A test part from Sandia National Laboratories.

There are some issues that are under further investigation. The major consideration is to add more knowledge into the system such as auto-meshing patterns to guide the decomposition so that the decomposition is more thorough and the result is more intuitive to meshing.

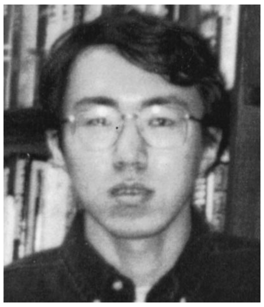
Acknowledgements

The authors gratefully acknowledge the financial support received from Sandia National Laboratories (sponsored by the US Department of Energy under Contract No. DE-AC04-94AL85000), National Science Foundation GOALI (DMI 9708531), National Science Foundation Engineering Research Equipment (DMI 9622665) and National Science Foundation Early Career Development (DMI 9501760). The rendering of decomposition results is performed by CUBIT.

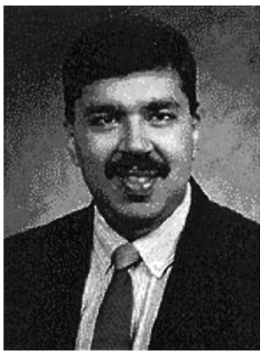
References

[1] Benzley SE, Perry E, Merkley K, Clark B, Greg S. A comparison of all hexahedral and all tetrahedral finite element meshes for elastic and elasto-plastic analysis, Proceedings of the 4th International Meshing Roundtable, Sandia National Laboratories, 1995. p. 179–91.
[2] Owen SJ. A survey of unstructured mesh generation technology, Proceedings of the 7th International Meshing Roundtable, Sandia National Laboratories, 1998. p. 239–67.
[3] Cook WA, Oaks WR. Mapping methods for generating three-dimensional meshing. Computers in Mechanical Engineering 1983;1:67–72.
[4] White DW, Mingwu L, Benzley SE. Automated hexahedral mesh generation by virtual decomposition, Proceedings of the 4th International Meshing Roundtable, Sandia National Laboratories, 1995. p. 165–76.
[5] Price MA, Armstrong CG, Sabin MA. Hexahedral mesh generation by medial surface subdivision: part I. Solids with convex edges. Int J Num Meth in Eng 1995;38(19):3335–59.
[6] Price MA, Armstrong CG. Hexahedral mesh generation by medial axis subdivision: part II. Solids with flat and concave edges. International Journal of Numerical Methods in Engineering 1997;40(1):111–36.
[7] Hohmeyer ME, Christopher W. Fully-automatic object-based generation of hexahedral meshes, Proceedings of the 4th International Meshing Roundtable, Sandia National Laboratories, 1995. p. 129–38.
[8] Blacker TD. The Cooper Tool, Proceedings of the 5th International Meshing Roundtable, Sandia National Laboratories, 1996. p. 13–29.
[9] Knupp PM. Applications of mesh smoothing: copy, morph, and sweep on unstructured quadrilateral meshes. Int J Numer Meth Eng 1999;45:37–45.
[10] Staten ML, Canann SA, Owen SJ. BMsweep: locating interior nodes during sweeping. Proceedings 7th International Meshing Roundtable, Sandia National Laboratories, Albuquerque, New Mexico, October 1998.
[11] Mingwu L, Benzley SE, White DR. Automated hexahedral mesh generation by generalized multiple source to multiple target sweeping. Int J Numer Meth Engr 2000;49:261–75.
[12] White DR, Tautges TJ, Tautges J. Automatic Scheme Selection for Toolkit Hex Meshing. Int J Numer Meth Engr 2000;49:127–44.
[13] Tautges TJ, Blacker TD, Mitchell S. The whisker weaving algorithm: a connectivity-based method for constructing all-hexahedral finite element Meshes. Int J Num Methods in Eng 1996;39:3327–49.
[14] Folwell NT, Mitchell SA. Reliable whisker weaving via curve contraction. Proceedings of the 7th International Meshing Roundtable. Sandia National Laboratories, Albuquerque, New Mexico, October 1998.
[15] Schneiders R. Automatic generation of hexahedral finite element meshes, Proceedings of the 4th International Meshing Roundtable, Sandia National Laboratories, 1995. p. 103–14.
[16] Schneiders RA. A grid-based algorithm for the generation of hexahedral element meshes. Engineering with Computer 1996;12(3-4):168–77.
[17] Owen SJ. H-Morph: an indirect approach to advancing front hex meshing. Int J Numer Meth Engr 2000.
[18] Meyers RJ, Tautges RJ, Tuchinsky PM. The 'hex-tet' hex-dominant meshing algorithm as implemented in CUBIT. Proceedings of the 7th International Meshing Roundtable. Sandia National Laboratories, Albuquerque, New Mexico, October 1998.
[19] Mitchell SA, Clark BW, Ostensen RW, Tautges TJ, White DR. Personal experience modeling the neutron generator power supply, Sandia National Laboratories, September 1997.
[20] Knupp P, Melander DJ, Mitchell SA, Tautges TJ, White DR. Personal experience modeling the neutron generator tube, July 1998.
[21] Blacker TD, Stephensoon MB. 1989. Using conjoint meshing primitives to generate quadrilateral and hexahedral elements in irregular regions, Proceedings ASME, Computers in Engineering Conference.
[22] Blacker TD, et al. Automated quadrilateral mesh generation: a knowledge system approach. ASME Paper No. 88-WA/CIE-4 1988.
[23] Armstrong CG, Robinson DJ, McKeag RM, Li TS, Bridgett SJ, Donaghy RJ, McGleenan CA. Medials for meshing and more. Proceedings of the 4th International Meshing Roundtable, Sandia National Laboratories, October 1995.
[24] Sheffer A, Etzion M, Rappoport A, Bercovier M. Hexahedral mesh generation using the embedded Voronoi graph, Proceedings of the 7th International Meshing Roundtable, Sandia National Laboratories, 1998. p. 347–64.
[25] Bih-Yaw Shih, Hiroshi Sakurai. Automated hexahedral mesh generation by swept volume decomposition and recomposition, Proceedings of the 5th International Meshing Roundtable, Sandia National Laboratories, October 1996.

[26] Chazelle BM. Convex Decomposition of Polyhedra, Symposium on Theory of Computing, Milwaukee 1981:70–79.
[27] Woo TC. Feature extraction by volume decomposition, Proceedings of Conference on CAD/CAM in Mechanical Engineering, March 24-26Cambridge, MA: MIT, 1982. p. 39–45.
[28] Sakurai H, Dave P. Volume decomposition and feature recognition, Part II: curved objects. Computer-Aided Design 1996;28(6/7):519–37.
[29] Kyprianou L. Shape classification in computer aided design, PhD Thesis, University of Cambridge, 1980
[30] Subrahmanyam S, Wozny M. An overview of automatic feature recognition techniques for computer-aided process planning. Computers in Industry 1995;26:1–21.
[31] Sonthi R, Kunjur G, Gadh R. Shape feature determination using the curvature region representation, Proceedings of Fourth Symposium on Solid Modeling and Applications, Sponsored by ACM SIGGRAPH, Atlanta, Georgia, May 14–16, 1997, pp. 285–96.
[32] Gadh R, Prinz FB. Reduction of geometric forms using the differential depth filter. Computer-Aided Design 1992;24(11):583–98.
[33] Blacker TD et al., CUBIT mesh generation environment, volume 1: user's manual, SAND94-1100, Sandia National Laboratories, Albuquerque, New Mexico, May 1994.
[34] Tautges TJ, Shang-sheng L, Lu Y, Gadh R ;Feature recognition applications in mesh generation, AMD–Vol. 220, Trends in unstructured mesh generation, ASME 1997.
[35] Liu S-S, Gadh R. Finite element hexahedral mesh generation via decomposition of swept and convex basic LOgical Bulk (BLOB) shapes. Journal of Manufacturing Science and Engineering, ASME Transactions 1998;120(4):728–35.
[36] Liu S-S, Dr of Philosophy, Mechanical Engineering, The definition and extraction of shape abstractions for automatic finite element hexahedral mesh generation, 1997.
[37] Lu Y, Gadh R, Tautges TJ. Feature decomposition for hexahedral meshing. "Feature decomposition for hexahedral meshing", Proceedings 25th Design Automation Conference, Proceedings 25th Design Automation Conferences, 1999 ASME International Design Engineering Technical Conferences and Computers in Engineering Conference, Las Vegas, NV, September 12–16. 1999
[38] Lu Y, Gadh R, Tautges TJ. Generating cutting surfaces for solid model decomposition: bounding a surface over a closed or open chain of edges. Proceedings 20th Computers and Information in Engineering Conference, ASME International Design Engineering Technical Conferences and Computers in Engineering Conference, Baltimore, MD, September 10–13. 2000
[39] Spatial Technology Inc., ACIS Geometric Modeler Application Guide, 1996
[40] Knupp P. Achieving finite element mesh quality via optimization of the jacobian matrix norm and associated quantities. Part II — A framework for volume mesh optimization. Int J Numer Meth Engr 2000.


Yong Lu is a doctoral candidate at the Department of Mechanical Engineering in the University of Wisconsin-Madison. He received his BS and MS in Tsinghua University at Beijing, China. His primary interests are in Computer Aided Design, Geometric Modeling and Feature Recognition.

Timothy J. Tautges is a Principal Member Technical Staff in the Parallel Computing Sciences department at Sandia National Laboratories and an Adjunct Professor of Engineering Physics at the University of Wisconsin-Madison. He received his Ph.D in Nuclear Engineering and Engineering Physics from the University of Wisconsin-Madison in 1990. His primary interests are in hexahedral mesh generation, geometry tools related to mesh generation, and parallel computing.


Dr Rajit Gadh, is currently a Professor at the University of Wisconsin-Madison and Director of the eMedia Center for B2B eCommerce as well as Director of the CAD-IT consortium. He received his Ph.D. from Carnegie Mellon University, MS from Cornell University, BS from IIT Kanpur. Dr Gadh has over eighty publications and technical articles in various journals, conference proceedings and magazines and has taught for a year at the University of California-Berkeley. Dr Gadh has received research and educational grants from the following companies: Ford, Caterpillar, Lucent Technology, Pratt & Whitney, CMI Tech., Texas Instruments, HP and Alcoa. He was the recipient of the NSF-CAREER award, NSF Research Initiation Award, and SAE Ralph R. Teetor Award, and has received several grants from NSF, DOE-Sandia and EPA.