INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng 2004; **61**:209–237 (DOI: 10.1002/nme.1063)

LayTracks: a new approach to automated geometry adaptive quadrilateral mesh generation using medial axis transform

W. R. Quadros[1,‡], K. Ramaswami[2], F. B. Prinz[2] and B. Gurumoorthy[1,*,†]

[1]*Department of Mechanical Engineering, Indian Institute of Science, Bangalore 560 012, India*
[2]*Department of Mechanical Engineering, Stanford University, U.S.A.*

SUMMARY

A new mesh generation algorithm called 'LayTracks', to automatically generate an all quad mesh that is adapted to the variation of geometric feature size in the domain is described. LayTracks combines the merits of two popular direct techniques for quadrilateral mesh generation—quad meshing by decomposition and advancing front quad meshing. While the MAT has been used for the domain decomposition before, this is the first attempt to use the MAT, for the robust subdivision of a complex domain into a well defined sub-domain called 'Tracks', for terminating the advancing front of the mesh elements without complex interference checks and to use radius function for providing sizing function for adaptive meshing. The process of subdivision of a domain is analogous to, formation of railway tracks by laying rails on the ground. Each rail starts from a node on the boundary and propagates towards the medial axis (MA) and then from the MA towards the boundary. Quadrilateral elements are then obtained by placing nodes on these rails and connecting them inside each track, formed by adjacent rails. The algorithm has been implemented and tested on some typical geometries and the quality of the output mesh obtained are presented. Extension of this technique to all hexahedral meshing is discussed. Copyright © 2004 John Wiley & Sons, Ltd.

KEY WORDS: lay tracks; MAT; all quadrilateral adaptive mesh

1. INTRODUCTION

In finite element method (FEM) complex geometry to be analysed is discretized as an assemblage of simple elements. Discretization of a domain into a set of finite elements is a geometrically based process and is referred to as mesh generation. Simplicial triangular elements have linear shape function. As the shape functions of quadrilateral (quad) elements are bilinear, they yield better results [1].

*Correspondence to: B. Gurumoorthy, Department of Mechanical Engineering, Indian Institute of Science, Bangalore 560 012, India.
†E-mail: bgm@mecheng.iisc.ernet.in
‡Currently at Department of Mechanical Engineering, Carnegie Mellon University

Received 10 December 2002
Revised 19 February 2003
Accepted 2 September 2003
Published online 15 July 2004

Quad meshing algorithms are grouped into two main categories: direct approach and indirect approach [2]. In the indirect approach, triangular meshing of a domain is followed by various algorithms to convert the triangles into quadrilaterals [3–5]. LayTracks belongs to the direct approach, where quadrilaterals are directly placed on a surface, thus avoiding the initial triangulation. The techniques proposed for direct quad meshing can be further grouped into two broad categories—decomposition based techniques and advancing front based techniques.

In the advancing front method, nodes are first paved on the boundary, then the elements' front advances inwards [6, 7]. Decomposition based techniques work by subdividing the region into simpler (usually convex) shapes and using templates to divide the simpler regions into quad meshes [8–12].

MAT for triangular mesh generation has been proposed by Gursoy and Patrikalakis [13–17]. Gursoy and Patrikalakis used MAT to detect constrictions, extract holes and decompose complex shape into a set of topologically simple three- or four-sided sub-domains. Nodes are then generated on the boundary and interior of each simple sub-domain to generate a triangular mesh. Mesh smoothing and local mesh refinement are suggested to improve the quality of the mesh.

Tam and Armstrong [18] used the medial axis to generate quadrilateral mesh. The MAT was used to subdivide the object into regions that are convex. Templates are then used to mesh the simpler regions. These techniques use the medial axis or surface to primarily decompose the domain into simpler regions that can be meshed using templates. While commercial mesh generators based on the MA/Medial Surface [19, 20] are available, this approach has two important problems. The first is the difficulty in predicting all possible topologies that may be obtained by the decomposition and providing templates for meshing all such regions. The second problem is that an additional step (based on linear programming) [18] has to be used to ensure that the element divisions in adjacent regions match.

Srinivasan *et al.* [21] used Voronoi diagram [22] to generate a triangular mesh and later a greedy algorithm has been used to convert the triangular mesh into a mix of quadrilateral and triangular mesh.

Sampl [23] described a 3D hex-dominant meshing algorithm that uses the medial axis transform for subdivision. The medial surface obtained is first meshed into a quad-dominant mesh, containing triangles and quadrilaterals, using advancing front based algorithm (Paving). The mesh on the medial surface is then extruded on both sides of the medial surface till it intersects the boundary of the object to obtain 3D mesh. As the mesh is formed on the medial surface and extruded to the boundary, there is no control on the quality of mesh at the boundary. Also, the output mesh contains non-hex elements at the concave edges and vertices. Reducing the 3D meshing problem into 2D quad meshing, followed by extrusion/sweeping is very similar to the idea developed by the authors [24]. There are two main differences. The first is that, in 3D version of LayTracks, quad meshing is obtained on the boundary of the domain and is then propagated towards the medial axis. This gives better control on the quality of the elements, as radii of maximal circles are orthogonal to the boundary (Figure 2). The second difference is that, the 3D version of LayTracks, uses the 2D quad meshing algorithm described in this paper, for meshing the connected boundary surfaces of the solid, thus can generate an all-quad mesh on the boundary. Merging of non-quad elements at concave vertices, described in Figure 16 can be extended to 3D to overcome the non-hex elements present in the Sampl's method.

1.1. Geometry adaptive meshing

Adaptive meshing refers to the control of the element size and there by the density of a mesh by adapting to geometric features in the model or physics of the problem. Element sizes are controlled based on the geometric features in the model, or based on a previous analysis solution. The objectives of adaptive meshing based on the analysis solution are—minimizing discretization errors (in small-deformation problems), optimizing mesh quality (in large-deformation problems). The objective underlying adaptive meshing based on geometric features in the model is to generate a fine mesh only in the regions, where the geometric features are small (but significant to analysis) and therefore greater accuracy is desired, and leaving a coarse mesh in the rest of the domain. Sharp corners, notches and holes are examples of such small features. In the remainder of this paper, the term adaptive meshing refers to, the process where the element size is controlled to adapt to only the geometric features in the part.

Mesh refinement algorithms [2, 25] are used to locally reduce element size in a mesh to adapt the mesh to either distorted geometry or to improve discretization error. In the literature, most efforts on adaptive meshing need an initial mesh and a means to deduce the mesh size at each node in the mesh based on results of analysis on the initial mesh [26]. Borouchaki *et al.* [27] used the term control space to refer to the background mesh and the discrete field of metrics associated with the nodes of the background mesh. Frey and Marechal [28] proposed the use of quadtree decomposition of a planar domain as a control space to specify desired mesh size in the domain. Khawaja *et al.* [29] used a octree encoding of the domain to control the spacing of nodes and size of elements on a surface mesh to generate a mesh that adapts to varying feature sizes in the domain. Commercial packages need user interaction, either to cluster nodes around regions of interest or to specify the element size at critical locations such as concave corners.

Radius function of the MAT is a measure of proximity between the geometric entities and local feature size [13, 15, 16, 21]. Gursoy and Patrikalakis [13, 15, 16] used MAT to detect constrictions, extract holes, and to generate adaptive triangular mesh. Srinivasan *et al.* [21] used radius function of the MAT to generate a nodal spacing function. Nodal spacing function was used to place nodes at the boundary and interior of the sub-domains identified using the MAT. These nodes are then connected using constrained Delaunay triangulation to generate an adaptive triangular mesh.

In the following sections, we describe a new algorithm, LayTracks, that belongs to the category of direct methods. While the MA has been used in meshing algorithms before for decomposition [13, 15, 16, 18, 21, 30, 31], the potential of MAT for the decomposition of the domain into well defined sub-domains called 'Tracks' and for termination of the advancing mesh (inherent in the mapping between the boundary and the MAT) has not been exploited thus far. LayTracks is the first approach to use the MAT to combine the merits of decomposition and advancing front techniques to obtain an all-quad mesh on a set of connected planar surfaces. LayTracks uses the radius function of the MAT, to automatically generate an all-quad geometry adaptive mesh on an arbitrary planar surface. The mesh satisfies the characteristics of a good mesh such as, boundary sensitivity and fewer irregular nodes [6]. The algorithm is able to generate uniform quad mesh on concave polygons, multiply connected polygons and non-manifold set of connected planar surfaces in 3D space. Adaptive all-quad mesh is generated on single surface. Metrics for the quality of the output quad elements such as skew angle, aspect ratio, and taper [32] are tabulated. It will be shown that, the proposed algorithm can

be extended to all hexahedral meshing. We describe the medial axis transform next as that is central to the LayTracks algorithm.

2. MEDIAL AXIS TRANSFORM

Medial axis transform was first proposed by Blum [33, 34] to describe biological shapes. The medial axis (MA) is defined as the loci of centres of locally maximal balls inside an object. The loci with the radii of the associated locally maximal balls is defined as medial axis transform (MAT) of the object. The boundary and the corresponding MA of an 2D object is shown in Figure 1. The points where three or more 1D entities of the MA meet are called branch points. Associated with the MA is a radius function R, which defines for each point on the axis its shortest distance to the boundary of the object.

LayTracks algorithm is based on the uniqueness and continuity of the MAT as given by the following lemma [35].

Lemma 1.0 (*Uniqueness and Continuity of Mapping to MAT*)
Let $\mathscr{A}$ be an n-dimensional compact sub-manifold of $\mathbf{R}^n$ and let MA($\mathscr{A}$) be its medial axis. Let $\mathscr{P}$ be an open subset of $\partial\mathscr{A}$ which is G^1- and piecewise C^2-continuous. Then for every point $p \in \mathscr{P}$ there is one and only one maximal ball touching p. Furthermore, the function $M : \mathscr{P} \longrightarrow \text{MA}(\mathscr{A})$, which maps each point $p \in \mathscr{P}$ to the centre of its maximal ball, is continuous.

This lemma dictates that given a bounded and closed manifold, for every point P on the boundary of this manifold there exists a unique projection Q on the medial axis that is the centre of the maximal ball touching P. Q is referred to as the projection/mapping of P on the MA. From the above lemma, it follows that, the projection/mapping of points on the boundary of a closed and bounded manifold is unique and continuous. Although the above lemma assumes that the boundary of the manifold should be G^1-continuous (its tangent direction is continuous) and piecewise C^2-continuous (the second derivatives are continuous), it can be generalized

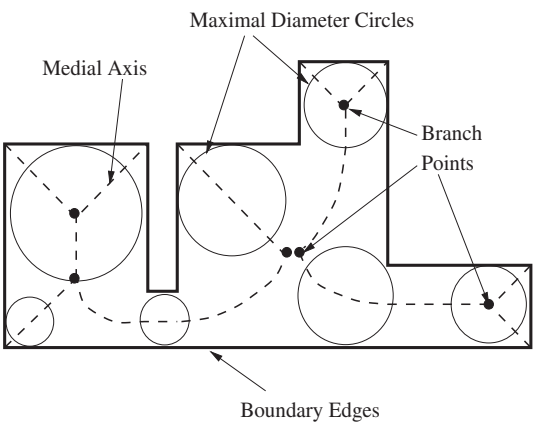

Figure 1. Boundary and its Medial Axis (MA).

to any shape with tangent discontinuities by replacing sharp corners with infinitesimal trimmed circles/spheres (Figure 13).

3. OVERVIEW OF THE ALGORITHM

The algorithm works similar to the process of laying rails on the ground to form tracks (and hence the name LayTracks). Consider a part of the object along with its MA, which is shown in Figure 2. The projection of the points p and p' lying on the boundary onto the MA are M_p and $M_{p'}$, respectively. Similarly, the projection of medial points M_p and $M_{p'}$ on the other boundary segment are p_1 and p'_1, respectively. The mapping from a MA segment to the two boundary segments is called '2-way mapping'. Laying of rails using this 2-way mapping is a straightforward process and is done by connecting each point p on the domain boundary to its projection M_p and by connecting M_p to its projection p_1 on other boundary segment.

Laying rails using the 2-way mapping, first subdivides the domain into regions called corridors. Corridors are formed by laying of rails from source nodes on the boundary, which are projection of branch points on the boundary. Rails are then laid inside the corridors to split each corridor into tracks. Unlike other decomposition methods, LayTracks which uses 2-way mapping is more generic in its subdivision. It always subdivides any complex input surface into tracks, so there is no need of templates and the problem of ensuring conformality of nodes between the tracks/subdomains does not arise. Note that as shown in the Figure 2 the tracks change direction at the MA to maintain an orthogonal orientation at the boundary. Laying of rails in this fashion to form connected set of tracks continues till the entire region is covered.

The main task in laying rails to form tracks, is to place appropriately the points p on the boundary of the surface. Rails are laid at uniform spacing inside a corridor to generate uniform, isotropic quad elements (element size $(w \times h)$ of good aspect ratio $(w/h \approx 1)$) on a set of connected surfaces. This can be viewed as a rolling disk, whose radius changes as

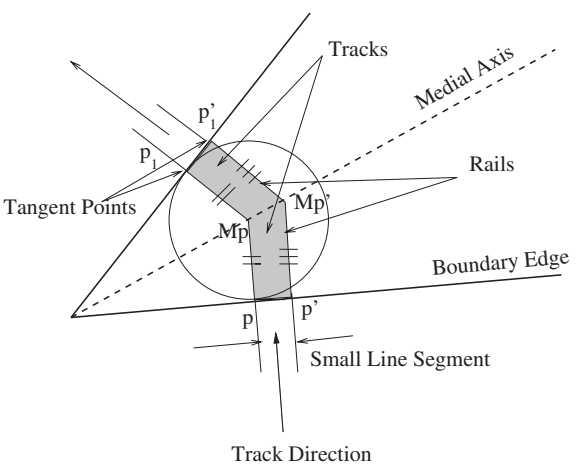

Figure 2. Laying of rails.

it rolls and the circumferential distance covered is maintained constant. The circumferential distance covered is the spacing between the two adjacent rails. This implies that the angular rotation varies to maintain constant circumferential width. In contrast, if the angular rotation of the disk with varying radius is maintained constant, then varying spacing between the rails is achieved resulting in an adaptive mesh (Figure 12). The radius function thus automatically provides the size information of the geometric features and controls the elements' size accordingly.

Quadrilateral elements are then obtained by placing sleepers/ties (nodes) at the appropriate spacing along the rails from the boundary towards the interior till the medial axis. This approach is similar to advancing front method [6] which results in nice elements at the boundary, whose contours in general follow the boundary of the domain. The expensive interference checks required at each advancing step is eliminated here. This is because the advancing mesh automatically terminates at the medial axis. It may be noted that the sequence here is opposite to that in railroad engineering where the sleepers are laid before the rails. Merging a pair of opposite triangles at the medial axis segment and merging two triangles at the concave corners results in all-quad mesh.

4. DETAILS OF THE ALGORITHM

Input to the quad mesh algorithm is a boundary representation (B-Rep) of a set of connected planar surfaces (patches) in 3D space and the MAT for each surface. Such domains are typical of the requirement for mesh generation for numerical simulation of injection moulding process and analysis of convective heat transfer. The input is also stored in the form of a dual graph, with the patches as nodes to capture the connectivity between the patches and is used while traversing across the patches. For adaptive quad meshing, the input contains a single surface with its MA. The main steps involved in the decomposition of the input domain and advancing of quad elements using MAT to generate all-quad mesh is given in Algorithm I.

Algorithm I
Input: Read a set of input connected planar surfaces along with their MA.
Output: Quadrilateral mesh.

begin

Step 1: Subdivision of the domain into corridors

- Identify all branch points in the MA of the surface patches.
- Find projection of branch points on the boundary and propagate rails across patches.

Step 2: Removing narrow corridors and forming tracks

- Identify all narrow corridors.
- Check if narrow corridor can be removed.
- Remove one of the rails of all possible narrow corridors (See Figures 6 and 7).
- Place rails inside corridors to form tracks.

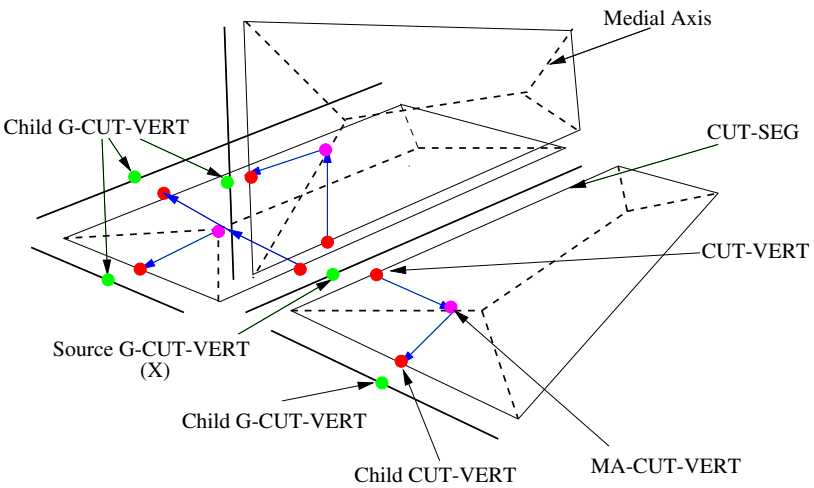

Figure 3. Propagation of rails to subdivide the boundary into CUT-SEGs.

Step 3: Generation of all-quad mesh in each track

- Place nodes on the rails.
- Place additional nodes on the MA (if needed).
- Build quad elements in each track and merge at the MA and concave vertices.

end

Steps 1, 2, and 3 are discussed in detail in Sections 4.1–4.3, respectively.

4.1. Subdivision of domain into corridors

As described in the overview, the critical part of subdividing the domain is in the placement of points on the boundary of the patches through which the rails will pass. If each boundary segment is simply divided into segments of equal length to obtain these points then it may so happen that, two adjacent rails (that would eventually form a track) will pass through different medial axis segments resulting in elements with undesirable shape. In order to eliminate this problem, the projection of all the branch points on the boundary form a set of candidate starting points (*Source* G-CUT-VERTs marked 'X' in Figure 3), for the propagation of rails. Rails split the edges at points called CUT-VERTs and cuts the MA at points called as MA-CUT-VERTs. A distinction is maintained between the CUT-VERT on the edge of each patch and the same CUT-VERT on the edge of the domain, which is denoted as G-CUT-VERT. Connection between the two is maintained in the data structure and is used to ensure connectivity between segments of a rail.

In the Figure 3, the patches are shown exploded with the edges shared between two patches shown separately in bold. The figure shows the propagation of a rail from a *Source* G-CUT-VERT to four *Child* CUT-VERTs in the three patches and is discussed below.

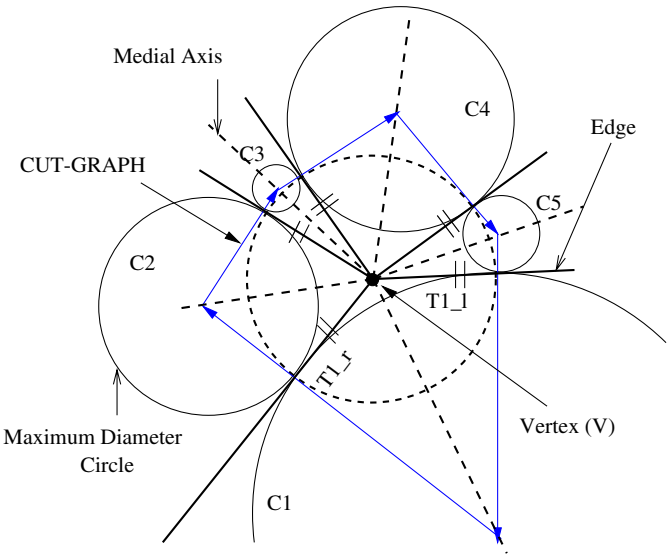

Figure 4. Rail forming loop in 2D space.

Algorithm II
Input: *Source* G-CUT-VERT
Output: Finding *Child* G-CUT-VERT by propagating rails across adjacent patches.

begin

for (every unvisited CUT-VERT, which is an image of *Source* G-CUT-VERT)

- split the edge at CUT-VERT
- mark the CUT-VERT as visited
- lay a rail between the CUT-VERT and its projection on MA (MA-CUT-VERT) (if the CUT-VERT is concave and no rails incident on it, the extreme MA vertex corresponding to the first dummy segment, in the direction of traversal of the boundary is selected as the MA-CUT-VERT ('M' in Figure 13))
- find the projection of the MA-CUT-VERT on the other edge and continue the rail between the MA-CUT-VERT and its projection (*Child* CUT-VERT)
- insert the image of *Child* CUT-VERT (*Child* G-CUT-VERT) as a child of the *Source* G-CUT-VERT

endfor

end

Rails propagation in breadth first traversal, starting from a *Source* G-CUT-VERT, with the *Child* G-CUT-VERTs becoming the source for the next recursion will result in the subdivision of the entire domain into corridors. The edges get divided into segments called CUT-SEGs, which represent the corridors.

The propagation of the rail will terminate at one of the following—an edge of the domain that is incident on only one patch, a concave vertex or at the SOURCE-CUT-VERT. Figure 4 shows that, a rail starting from a *Source* G-CUT-VERT will form a closed loop, if it does not terminate at either a boundary edge of the domain or a concave vertex. C_i are the maximal circles corresponding to the CUT-VERTs introduced on each edge in the i^th patch, by propagating one rail (marked R in the figure). The two edges of each patch shown incident at common vertex 'V', are tangents to the corresponding max circle. Let Tl_i denote the left tangent from V to C_i and Tr_i denote the right tangent from V to C_i. From geometry, it can be shown that $\mathrm{Tl}_i = \mathrm{Tr}_i$ (length of the line segments joining a point outside the circle with tangent points of the circle are equal). But, $\mathrm{Tr}_i = \mathrm{Tl}_{i+1}$ as they are identical segments (connecting V and the same CUT-VERT). Therefore, the rail R, as it propagates must always be tangential to a circle of radius Tl_i or Tr_i and hence must form a closed loop. This result holds good even when the individual patches are in 3D space.

The set of connected rails obtained above is mathematically represented as a simple bipartite graph G called CUT-GRAPH. CUT-GRAPH consists of a vertex set $V(G)$ and edge set $E(G)$, such that every edge $e \in E(G)$ has end vertices belonging to $V(G)$—one incident on the patch boundary and the other on the corresponding MA. This representation enables traversing along a rail or corridor across different surface patches in the domain.

4.2. Removing narrow corridors and forming tracks

It is possible that, the branch points in the MA of a surface are very close to each other, or the rails laid through the branch points of different patches come close. In such cases the rails result in narrow corridors (see Figure 6). These narrow corridors need to be removed, as otherwise they will result in quad elements with poor aspect ratio. After all the narrow corridors are removed, rails are laid inside the corridors to form a set of connected tracks. Sections 4.2.1–4.2.4 discuss this process in detail.

4.2.1. Detection of narrow corridors. The presence of narrow corridors is detected by visiting all the CUT-VERTs whose MA-CUT-VERT is a branch point. Visiting only these CUT-VERTs is sufficient as the rails that form a narrow corridor must pass through branch points. If the width of a corridor, which is incident on one of these CUT-VERTs is less than a specified value then the corridor is flagged as a narrow corridor. In the case of uniform quad mesh generation on multiple surfaces, corridors are flagged as narrow based on the desired quad element size. In adaptive meshing, the radius function at the branch point is used to flag the narrow corridors.

4.2.2. Preprocessing narrow corridor. As the narrow corridors propagate through several patches, it is necessary to check if a rail can be removed. For instance, a narrow corridor whose rail passes through a convex vertex implies that there is a patch where a branch point is very close to the vertex indicating a small/fine feature. Such regions would require a fine mesh. Therefore removal of narrow corridor is not advisable in this case.

In another case, there is a possibility that no rail is incident at a concave vertex at the end of deletion of rails (see Section 4.2.3). Therefore, the number of rails meeting at the concave vertex has to be tracked through the removal process.

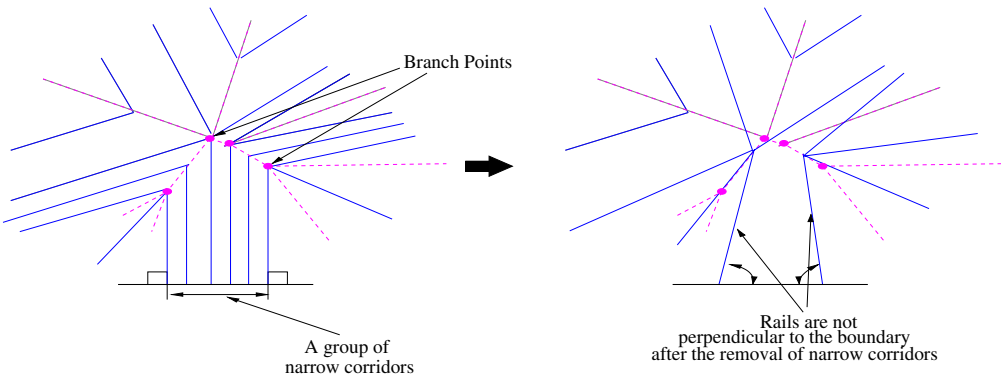

Figure 5. Unconditional removal of a set of adjacent narrow corridors.

Table I. Scenarios in narrow corridors removal.

Cases	Rail 'A' incident on branch point	Rail 'B' incident on branch point
Case 1	No	No
Case 2	No	Yes
Case 3	Yes	No
Case 4	Yes	Yes

There can be a group of adjacent narrow corridors. Merging of all these will result in a wider corridor. However, the end rails may not be perpendicular to the boundary (see Figure 5 and Section 4.2.3). Therefore laying of rails in these wide corridors to subdivide it into tracks may result in intersection of rails. In order to avoid this, we have to delete the rails from a group of narrow corridors till the minimum acceptable width is attained and the deleted CUT-SEGs should be marked. This is discussed in the next section.

Each CUT-VERT which represents a rail has two flags *never_del_l* and *never_del_r* to track if a corridor incident on either side of the rail at the CUT-VERT can be deleted. If it is found that the removal of one set of rails from the narrow corridor is not possible because of above mentioned reasons, then the flag *never_del_r* pointed by left hand CUT-VERT or *never_del_l* pointed by right hand CUT-VERT is set so that the rail is not deleted during removal of narrow corridors.

4.2.3. Deletion of a set of rails from narrow corridors. After examining a narrow corridor for the cases mentioned above, if it is found that it is safe to remove the rails, then the narrow corridor is corrected. Rails *A* and *B* can form a narrow corridor, only in four different combinations as given in Table I and these combinations are shown in Figure 6. A segment of the narrow corridor inside a single surface is corrected one at a time by removing rail *A*. The correction then proceeds along the corridor across all the patches it lies in till the entire narrow corridor is corrected (see Figure 7).

If 'A' does not pass through a branch point, then we have 2 cases as shown in Figures 8 and 9 depending upon the status of 'B'. In such cases we remove 'A' and the flag *deleted* is set to

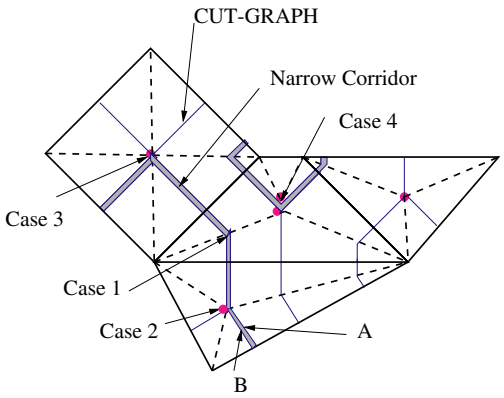

Figure 6. Four cases of narrow corridor.

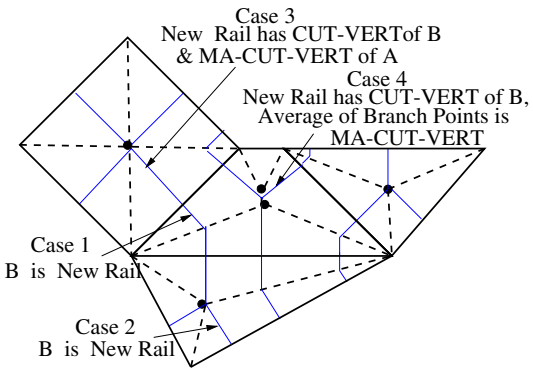

Figure 7. Removal of narrow corridor.

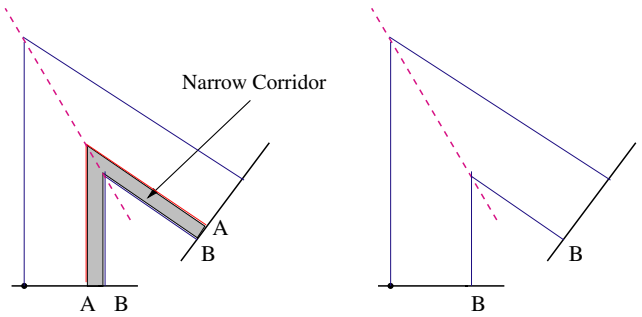

Figure 8. Case 1: MA-CUT-VERT of A and B are not branch points.

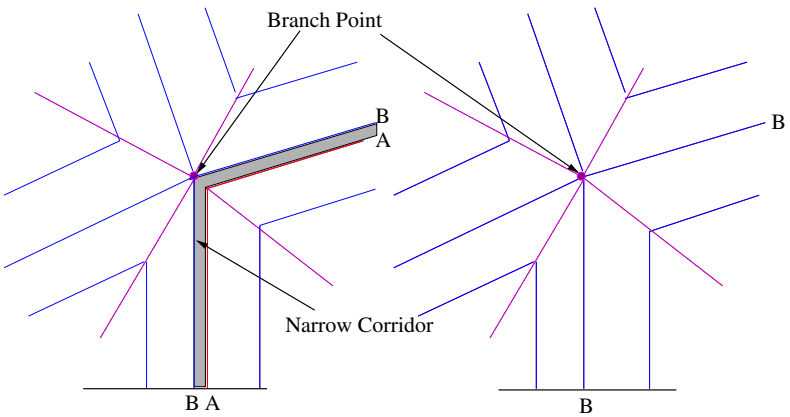

Figure 9. Case 2: MA-CUT-VERT of B is branch point, A is not.

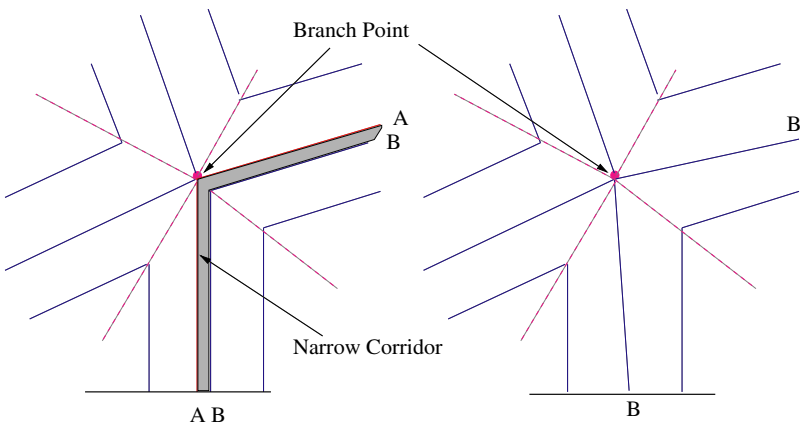

Figure 10. Case 3: MA-CUT-VERT of A is branch point, B is not.

1 in both the CUT-VERT that represents rail 'A' and the CUT-SEG between the rails 'A' and 'B'. The flag *deleted* indicates that during the formation of tracks (discussed in Section 4.2.4) rails should not be placed either inside the CUT-SEG that represented the narrow corridor or at the CUT-VERT that represented rail 'A'. This ensures that rails do not intersect during the formation of tracks (see Section 4.2.4).

On the other hand, if 'A' passes through a branch point, we have 2 more cases depending upon the status of 'B' as shown in Figures 10 and 11. In such cases in addition to removing rail 'A' and setting flag *deleted* to 1 in both the CUT-VERT representing rail 'A' and the CUT-SEG between the rails 'A' and 'B', the MA-CUT-VERTs of both rails are merged.

Gursoy and Patrikalakis [16], and Srinivasan *et al.* [21] have proposed methods to handle short MA branches by modifying the MA. In contrast, in our approach, the narrow corridors

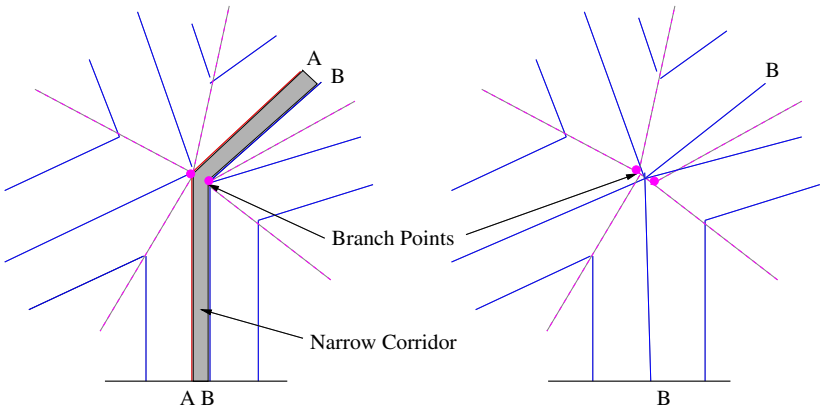

Figure 11. Case 4: MA-CUT-VERT of A and B are branch points.

are removed only by deleting/modifying the rails and the MA is not modified. Note that in all four cases given in Table I the MA-VERTs are not altered, i.e. MA is retained as before, thus retaining the perfect two-way mapping between the boundary and the MA. Only the two adjacent CUT-GRAPHs are merged along the narrow corridor. After removing the narrow corridor, flag *deleted* is set to 1 in both the CUT-VERTs representing the deleted rail and the CUT-SEG representing the narrow corridor. Note that the CUT-VERTs which represent the deleted rail are not deleted from the data structure. These are marked so that no additional rails are laid in these corridors. Retaining the two-way mapping and marking the deleted narrow corridors are important factors which ensure robustness of the algorithm.

4.2.4. Formation of tracks by laying rails. After the thin corridors are removed, the next step is to determine the spacing between rails to be laid in each corridor to form the tracks. Uniform spacing between the rails results in uniform quad mesh. Adaptive mesh is obtained by varying the width of the tracks based on the radius function of the MA.

Notation:

L	width of the corridor
m_i	gradient of radius function corresponding to ith CUT-VERT
s	unit vector along the direction of CUT-SEG
R_i	radius function (length of rail) at ith CUT-VERT
P_i	position vector of ith CUT-VERT
$\triangle S_i$	width of the track at P_i
$\triangle\theta_i$	incremental rotation of a circle at ith CUT-VERT

For uniform mesh on multiple surfaces, the nominal track width is calculated based on the desired quad element size (w). The number of CUT-VERT to be placed in each CUT-SEG (corridor) is calculated as below:

$$num\_cut\_vert = \left\lfloor \left(\frac{L}{w}\right) - 0.5 \right\rfloor \tag{1}$$

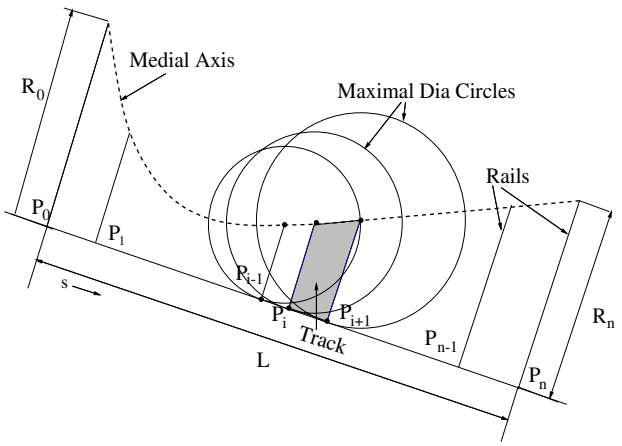

Figure 12. Placing CUT-VERTs adaptively inside a corridor.

For adaptive meshing on a single surface, spacing between the CUT-VERTs is based on the radius function and is explained in Algorithm III. The MA of a corridor in general consists of a set of piecewise linear MA-SEGs as shown in the Figure 12. In regions where the feature sizes are small, the radius function R_i is small, resulting in smaller track width and finer mesh. $\triangle\theta_i$ is composed of a constant input component $\triangle\theta_{in}$ and a variable component $\triangle\theta_{m_i}$. $\triangle\theta_{in}$ controls the overall resolution of the mesh. A small value of $\triangle\theta_{in}$, results in smaller width tracks and hence finer mesh and vice versa. $\triangle\theta_{m_i}$ is made to vary inversely proportional to the absolute value of m_i as given in Step 1 in Algorithm III, so that the elements are not skewed and comes closer to square. When the absolute value of m_i is 0.0, $\triangle\theta_{m_i}$ is equal to MAX_ANG_INCREMENT, which by default is set to 20% of $\triangle\theta_{in}$.

When either of the end CUT-VERTs P_0 and P_n of a corridor is a convex vertex, i.e. R_i is equal to 0.0, then the width of the track adjacent to the convex vertex is set to MIN_END_WIDTH, while finding the position of P_1 and/or P_{n-1} (Figure 12 and Step 2 in Algorithm III). The value of the MIN_END_WIDTH by default set to 3% of maximum MA radius, but can be found base on the internal angle at the convex vertex, so that aspect ratio of the corner element at the convex vertex becomes approximately 1.0 (Figure 15).

Step 3 in the algorithm checks for the cases where either no or only one CUT-VERT can be placed at a time in the corridor. These cases are identified by finding the *ratio* of the sum of the widths of the possible end tracks to the width of the corridor. If the *ratio* is less than 0.7, it is good not to split the corridor, thus avoiding bad aspect ratio quad elements. If the *ratio* lies between 0.7 and 1.3 it is good to place only one CUT-VERT, such that ratio of the widths of the two tracks is equal to $((P_n - P_{n-1})/(P_1 - P_0))$. If the *ratio* is greater than 1.3, it implies that corridor is wide enough, so a CUT-VERT is placed at the expected position P_1. The values 0.7 and 1.3 have been chosen to ensure that, the elements are as close to a square as possible.

Algorithm III
Input: R_0, P_0, R_n, P_n and $\triangle\theta_{in}$.
Output: Paving CUT-VERTs adaptively.

begin

Step 1: Find angular increments $\triangle\theta$ at P_0 and P_n.
Find m_0
if$(m_0 \leqslant 1)\{$

$\triangle\theta_{m_0} = |1 - m_0| * \textbf{MAX\_ANG\_INCR}\}$

else$\{\triangle\theta_{m_0} = 0.0\}$
$\triangle\theta_0 = \triangle\theta_{in} + \triangle\theta_{m_0}$
Find m_n
if$(m_n \leqslant 1)\{$

$\triangle\theta_{m_n} = |1 - m_n| * \textbf{MAX\_ANG\_INCR}\}$

else$\{\triangle\theta_{m_n} = 0.0\}$
$\triangle\theta_n = \triangle\theta_{in} + \triangle\theta_{m_n}$

Step 2: Find position of the rails of possible end tracks.
if$(R_0 \neq 0)\{\triangle S_0 = R_0 * \triangle\theta_0\}$
else$\{\triangle S_0 =$ **MIN_END_WIDTH**$\}$
$P_1 = P_0 + s * \triangle S_0$
if$(R_n \neq 0)\{\triangle S_{n-1} = R_n * \triangle\theta_n\}$
else$\{\triangle S_{n-1} =$ **MIN_END_WIDTH**$\}$
$P_{n-1} = P_n - s * \triangle S_{n-1}$

Step 3: Placing CUT-VERTs inside corridor.
$L = P_n - P_0$
$L_1 = P_1 - P_0 + P_n - P_{n-1}$
ratio $= L/L_1$
if$(0.7 \leqslant \text{ratio} \leqslant 1.3)\{$

$P_{\text{mid}} = P_0 + \frac{(P_n - P_0)}{(1+(P_n - P_{n-1})/(P_1 - P_0))}$

Place CUT-VERT at P_{mid}
goto end$\}$
if$(1.3 \leqslant \text{ratio})\{$
Place CUT-VERT at P_1
$P_0 = P_1$
$R_0 =$ Radius at P_0
repeat Steps $1, 2, 3\}$
end

The above procedure to place CUT-VERTs inside a corridor is repeated in all the corridors. Starting from each CUT-VERT, rails are then laid by connecting the CUT-VERT to its corresponding MA-CUT-VERT as explained in the previous section (Figure 3), to subdivide the corridors into tracks. A Concave vertex is visualized as made up of several connected small CUT-SEGs of zero length, as shown in Figure 13. The possible shapes of tracks inside a surface patch are shown in the Figure 14.

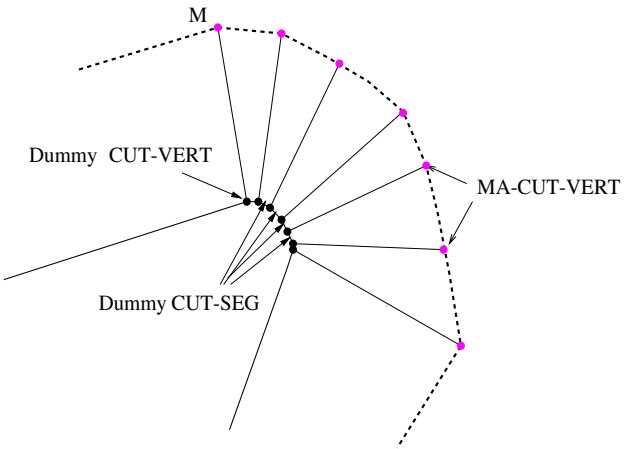

Figure 13. Tracks at a concave vertex.

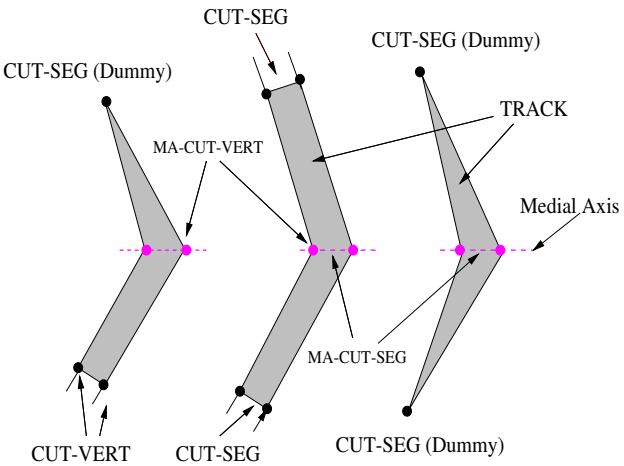

Figure 14. Possible types of tracks.

4.3. Generation of all-quad mesh in each track

The quad nodes and elements are generated in this step. Once the rails are in place, generating the quad elements involves placing nodes along the rails and connecting these correctly to form the quad elements. This is similar to laying sleepers along the track, except that in the case of railway track the sleepers are laid before the rails.

Elements are formed in a track between the boundary (CUT-SEG) and MA (MA-CUT-SEG) and then to the other boundary CUT-SEG. Formation of the corridors first ensures that as the tracks continue through to other surface patches in the corridor, nodes will match. The main advantage of the proposed meshing of individual tracks is that a conformal mesh is obtained at

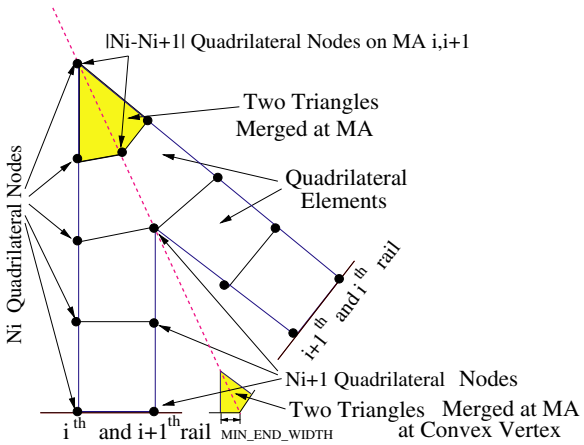

Figure 15. Placing extra quad nodes on MA.

the junctions of the tracks. In the case of mapping techniques [18] to mesh the simple domains obtained by decomposition (either using MAT or otherwise), additional processing is required to ensure that the elements conform at the boundaries of the simple domains.

4.3.1. Placing nodes on the rails. Let the maximal circle, with MA-CUT-VERT as the centre touches the boundary at 'n' CUT-VERTs and L_{avg} be the average length of the rails incident on the MA-CUT-VERT. Note that theoretically all the rails incident on the MA-CUT-VERT should be of equal length, but numerically/practically it may not be.

Let $\triangle S_{\text{l}}$ and $\triangle S_{\text{r}}$ be the width of the tracks on the left and right side of ith rail, respectively. Number of quad nodes (N) on each of the 'n' rails is given by Equation (2). Step size h_i for placing quad nodes on the ith rail, is given by Equation (3), where, R_i is the length of ith rail (radius).

$$N = \left\lfloor \frac{L_{\text{avg}}}{\left(\frac{\sum_{i=1}^{n} (\triangle S_{\text{l}} + \triangle S_{\text{r}}/2)_i}{n} \right)} + 0.5 \right\rfloor \tag{2}$$

$$h_i = \frac{R_i}{N} \tag{3}$$

4.3.2. Placing nodes on MA. For placing nodes inside a track, it is sufficient to consider at a time, a pair of tracks meeting at the MA (Figure 15), thereby rendering this task invariant to the complexity of the overall domain being meshed. Two rails at either end of a CUT-SEG may not have equal number of quad nodes. This difference in number of nodes is adjusted by placing nodes on the MA (Equation (4)), where N_i and N_{i+1} are the number of nodes on ith and $(i+1)$th rail, respectively, and $\text{MA}_{i,i+1}$ is the MA segment between two MA-CUT-VERT as shown in Figure 15.

$$N(\text{MA}_{i,i+1}) = |N_i - N_{i+1}| \tag{4}$$

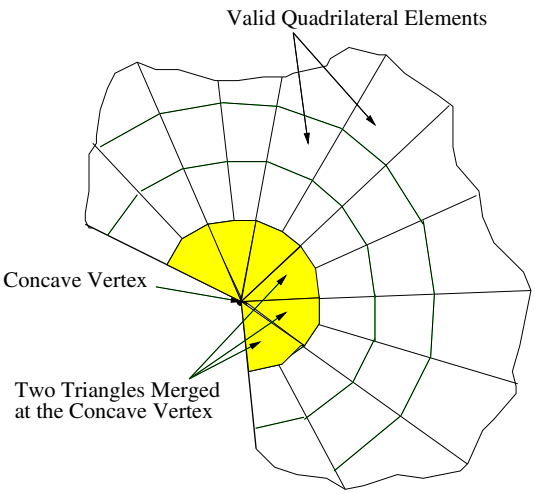

Figure 16. Merging two triangles at concave vertex.

4.3.3. Building quad element in each track and merging at MA. For an all-quad mesh, even number of nodes should exist on the boundary [36]. From Equation (5) it is clear that quad nodes on the boundary of a pair of tracks meeting at the MA is even in number (Figure 15). Since quad nodes at MA-CUT-VERT are repeated twice, two quad nodes are subtracted in Equation (5).

$$N_{\text{total}} = (N_i \times 2 + N_{i+1} \times 2 - 2)$$

$$N_{\text{total}} = 2(N_i + N_{i+1} - 1) \tag{5}$$

Quadrilateral elements are built by laying sleepers across nodes on adjacent rails, from the boundary till the medial axis. This ensures that any non-quad element (triangular only) inside track will be in the vicinity of MA (Figure 15). At the convex vertex, two triangular elements exist at the MA, which meets the boundary (Figure 15). Non-quad elements are also likely at concave vertices. Two opposite triangular elements are merged locally at that MA to ensure all quad mesh inside the track (Figure 15). Removal of common edge between the two consecutive triangles results in quad-element at the concave vertex (Figure 16).

5. RESULTS AND DISCUSSION

The LayTracks algorithm has been implemented on Suse Linux6.4 operating system, using Intel PentiumIII 450 MHz processor with 128 MB main memory and OpenGL for display. The algorithm has been tested and the intermediate stages of the algorithm such as, input planar surfaces with the MA, mapping between the MA and the boundary, the corridors, the tracks and the final quad mesh obtained are shown in Figures 12–25. The mesh generated for the cup, bearing, star shaped polygon and spanner contain 1736, 8716, 2967 and 1271 quad elements,

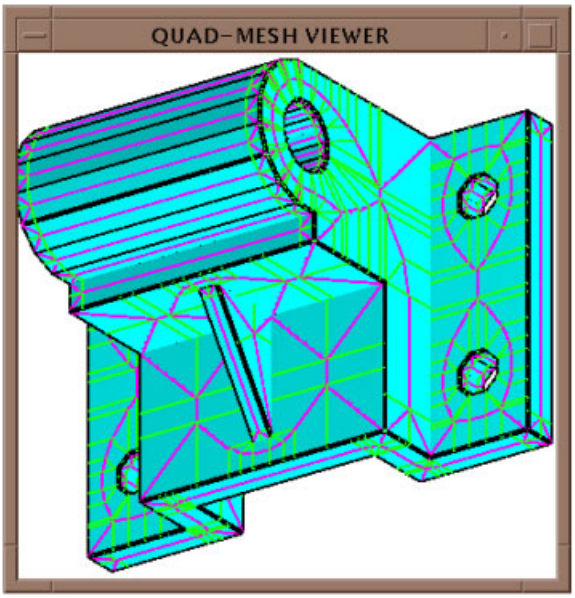

Figure 17. Two-way mapping using MAT.

respectively. The mesh is generated under 1 s of CPU time in all the cases except the bearing which took around 2 s. As mentioned before the MA of each patch is considered as input to the algorithm. As the algorithm requires the MAT of the domain, the time for the construction of MAT will add to the time complexity of the overall algorithm. However, as can be seen from the steps outlined in Algorithm I, the order complexity of the mesh generation algorithm LayTracks itself is linear in terms of the entities forming the boundary of the input domain (edges and concave vertices) or in terms of the number of branch points in the MA of the patches.

Quality of the output mesh is measured using metrics like skew angle, aspect ratio and taper [32]. Table II presents the average values of the above metrics for the output mesh obtained in the four examples. Commercial packages like NASTRAN classify elements with skew angle above 30^{o}, aspect ratio above 5 or taper below 0.8 as bad elements. Table III shows the percentage of elements below the threshold values indicated above in each case. It must be mentioned that no smoothing has been done on the meshes generated. Due to the decomposition of the domain into corridors, any irregular nodes (requireing local smoothing) that are present will only be in the vicinity of the medial axis.

Uniform quad meshing on multiple surfaces in 3D space are shown in Figures 12–19 and adaptive meshing on two single surfaces are shown in Figures 20–25. Figure 18 shows the closure of the loops of the rails which is explained in Figure 4. Adaptive meshing on multiply-connected single planar surface can be used to solve many 3D problems that can be idealized as plane stress, plane strain or axi-symmetric problem. Figure 23 shows that, quad elements at the centre are larger than those near sharp corners (close up view is shown in Figures 15 and 16). The main advantage of using the MAT for adaptive meshing is that user

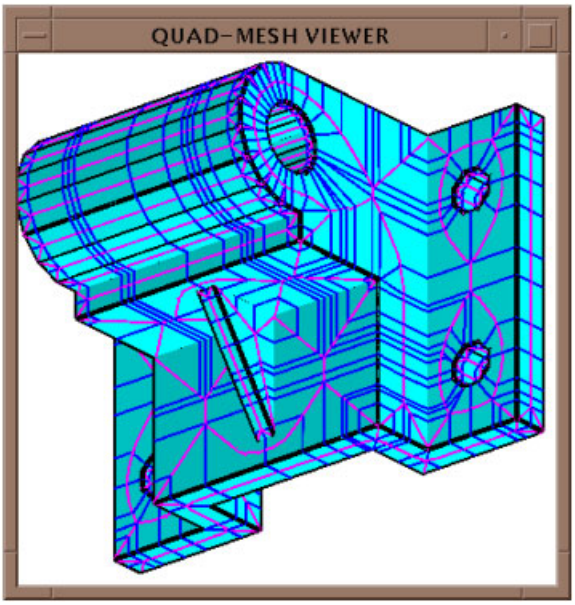

Figure 18. Object subdivided into corridors.

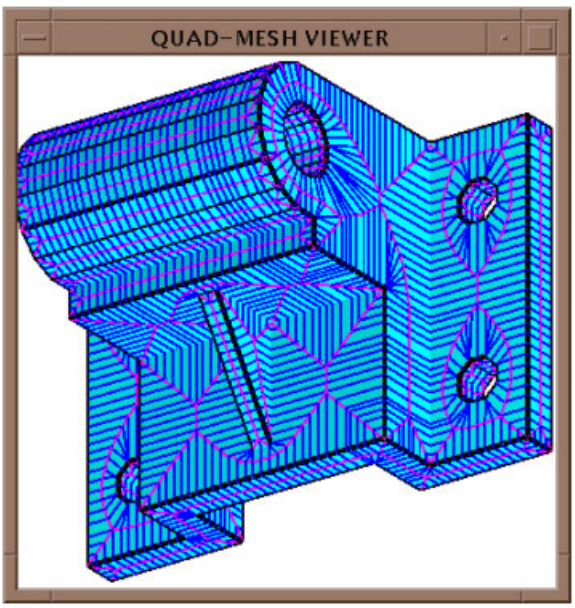

Figure 19. Set of connected tracks.

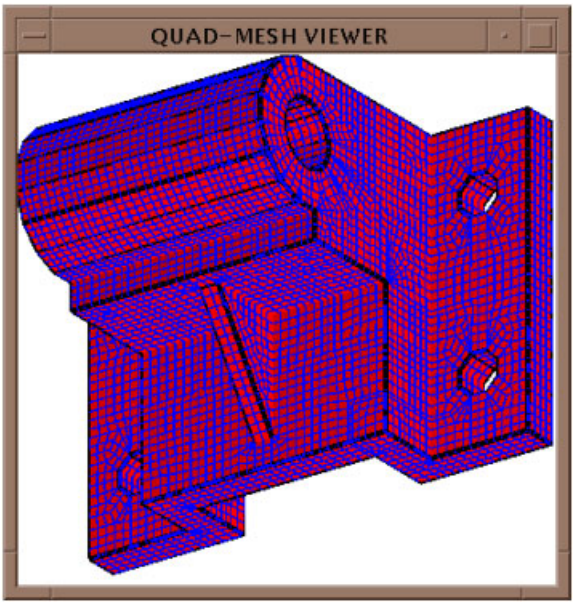

Figure 20. All quad mesh.

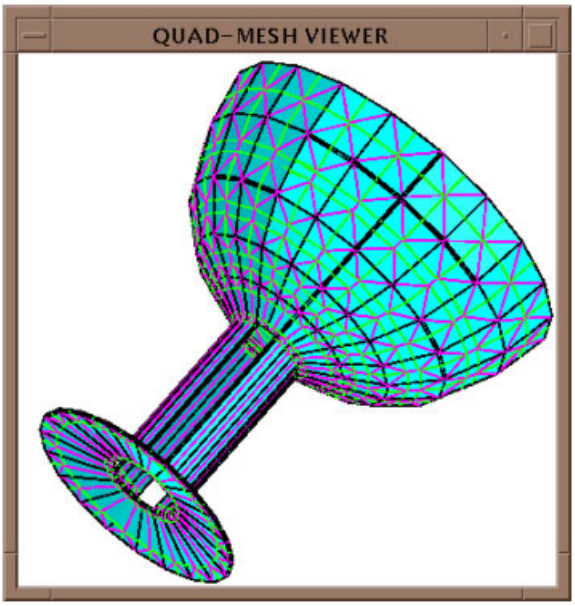

Figure 21. Two way mapping using MAT.

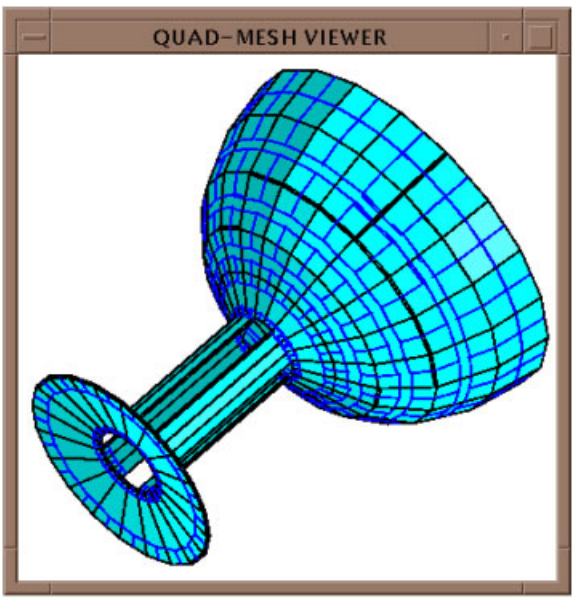

Figure 22. Object subdivided into corridors.

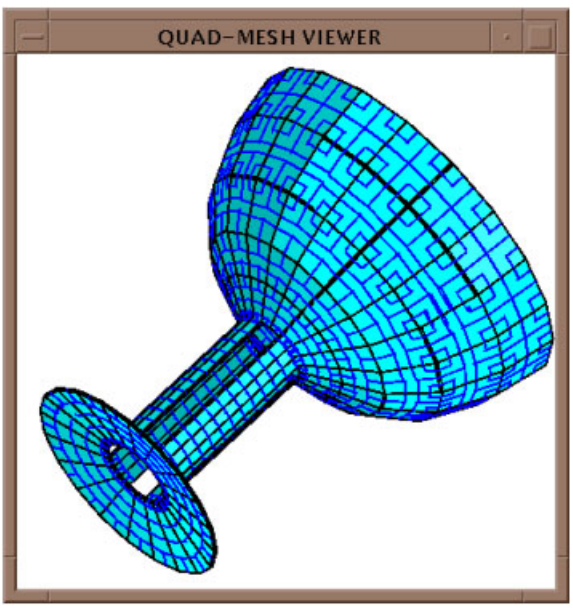

Figure 23. Set of connected tracks.

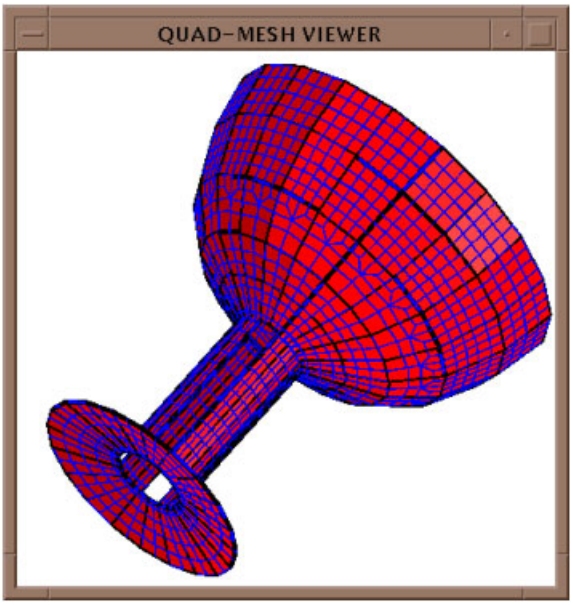

Figure 24. All quad mesh.

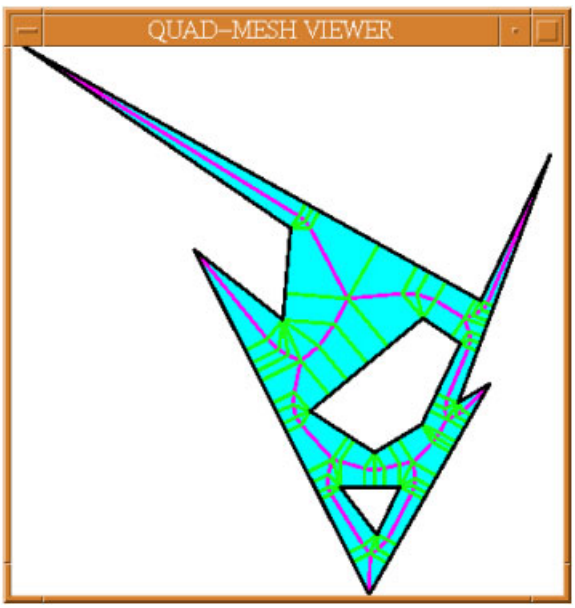

Figure 25. Two-way mapping using MAT.

Table II. Average quality of quad elements.

Test method	Bush bearing	Revolved cup	Star polygon	Spanner
Skew angle	3.410491°	1.988051°	3.487°	16.036°
Aspect ratio	1.481247	1.364381	1.036740	1.197994
Taper ratio	0.946564	0.950603	0.961896	0.924602

Table III. Percentage of elements below threshold value.

Test method	Bush bearing(%)	Revolved cup(%)	Star polygon(%)	Spanner(%)
Skew angle $\geqslant 30^{o}$	2.1297	3.1007	1.44	12.5
Aspect ratio $\geqslant 5$	0.2756	0.1831	0.002	0.001
Taper ratio $\leqslant 0.8$	5.3117	6.819	2.42	10.4

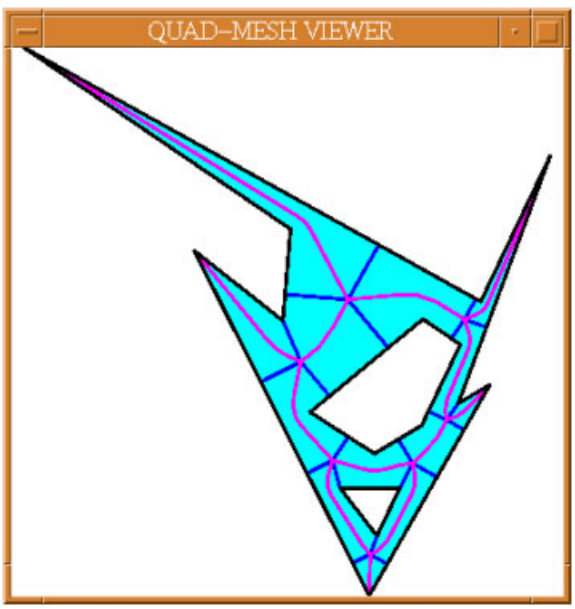

Figure 26. Object subdivided into corridors.

intervention is not required to cluster nodes/grid points in geometrically critical regions, as is the current practice. Adapting the mesh to analysis error indicators however may require additional inputs or user intervention. In such cases the mesh generated using the proposed algorithm (Figure 30) can be used as the initial mesh for preliminary analysis.

Adaptive meshing on set of connected planar surfaces could be achieved using the proposed algorithm by calculating the angular increment using the average gradient of all the MA segment in the adjacent surfaces that correspond to a CUT-SEG. Angular increment multiplied by the average of the radius function in all adjacent surfaces would yield the track width. Alternately,

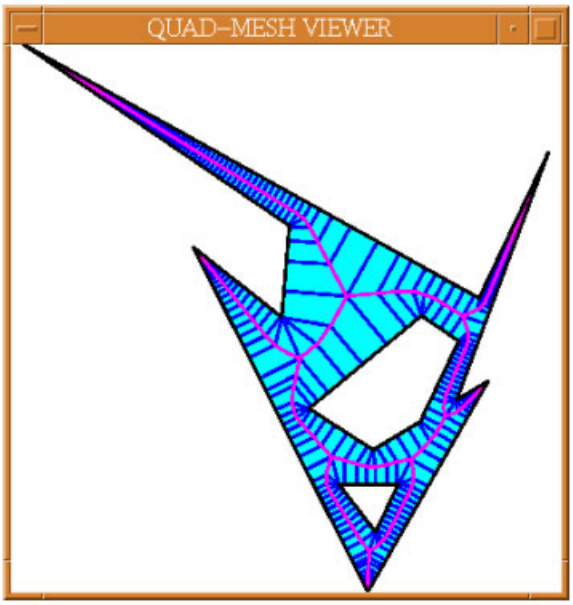

Figure 27. Set of connected tracks.

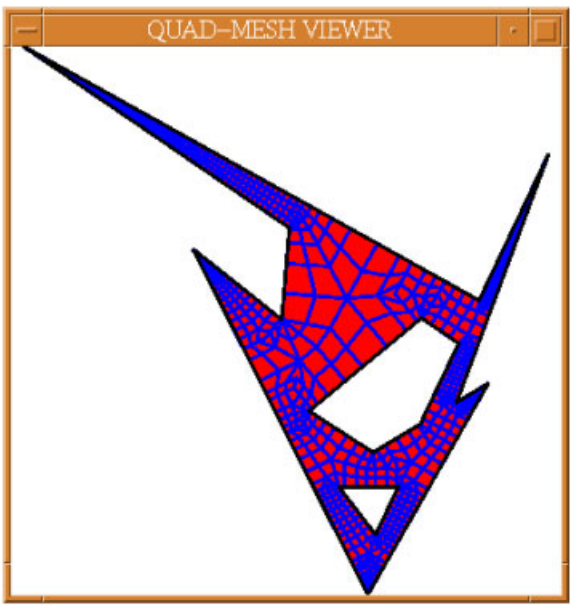

Figure 28. All quad mesh.

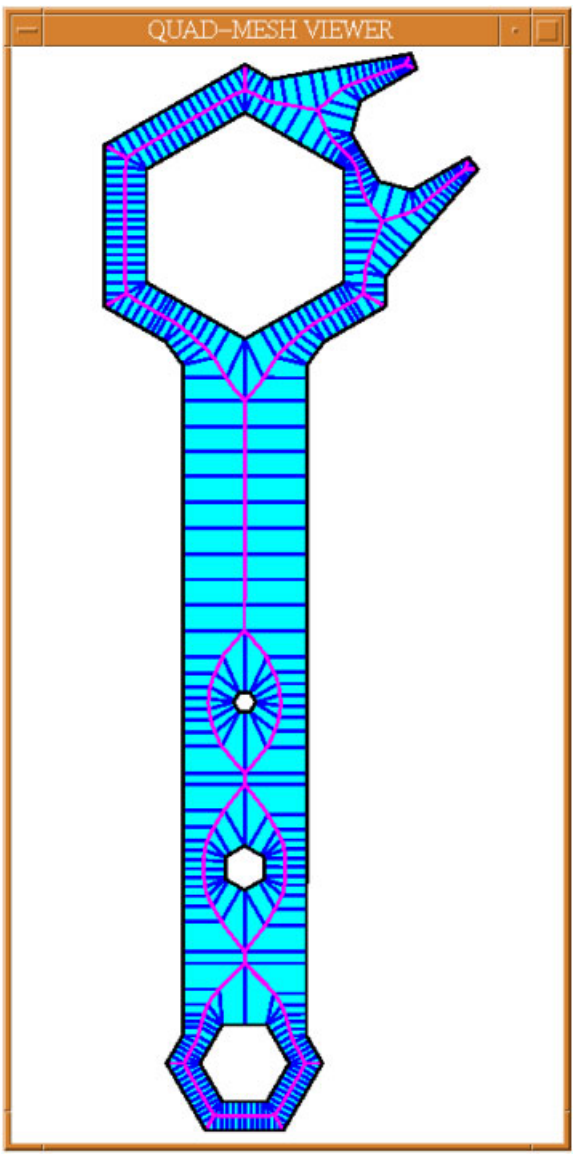

Figure 29. Set of connected tracks.

if the variations in radius function are large, the smallest value could decide the width. An additional step to remove narrow tracks may be required. Also, currently the width of a track is forced to be constant throughout the corridor and transition in the size of the quadrilateral elements inside a track is not possible. The use of non-linear rails (defined by Bezier or B-spline segments) is being explored to overcome this limitation [37]. Algorithm can be extended to mesh non-planar surfaces, provided the input MAT of such a domain is available.

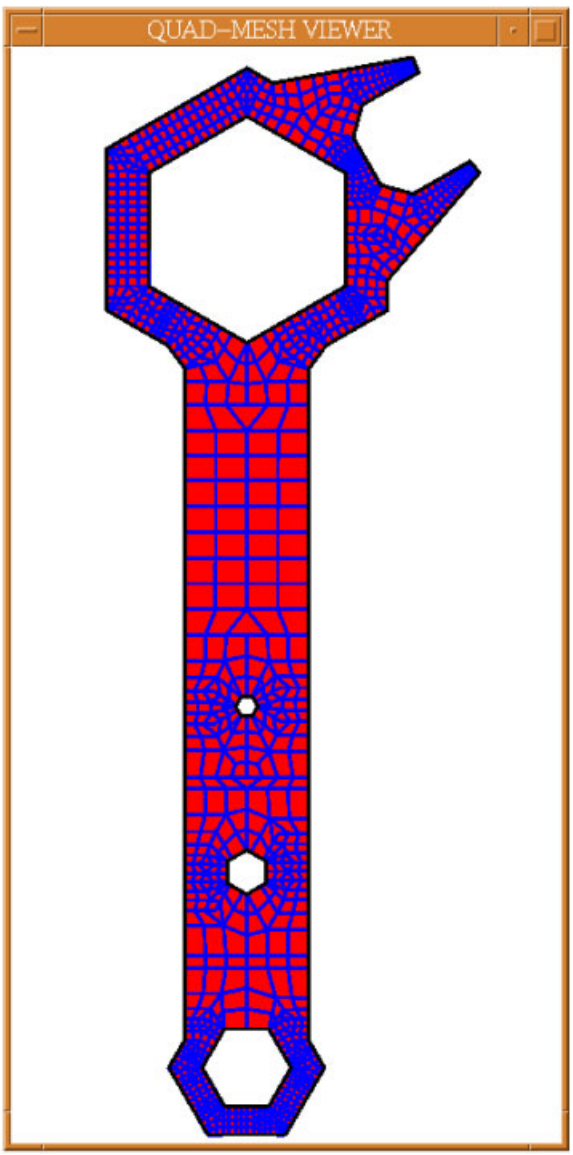

Figure 30. All quad mesh.

5.1. Hexahedral meshing

The idea of laying tracks to obtain quad meshing can be extended to 3D to obtain hexahedral meshing [37]. Like laying of rails from the boundary to the MA, in 3D it will be necessary to lay columns from the boundary to the medial surface. The corridors in 3D space can be obtained, by projecting seams and junctions of the medial surface onto the boundary of the solid, resulting in the subdivision of solid and partitioning of the outer boundary. Quad mesh generation on this set of partitioned boundary surfaces using the LayTracks algorithm, will

control the size and spacing of the columns to be laid in 3D. Hexahedral elements can then be built in every column from the boundary till the medial surface. Merging at the medial surface remains a open question for the completion of robust hexahedral mesh generation. Work is underway to extend the algorithm to hex meshing. Hex meshing based on the above extension of LayTracks will give better quality elements (as compared to the approach reported in Reference [23]) because the quad mesh will be generated on the boundary rather than the medial surface. Using a quad mesh on the boundary of the domain (to obtain a hex mesh) will also allow meshing of an assembly with conformal nodes at the interface of the volumes. Radius function of the medial surface can be used to adaptively control the element size in a manner analogous to the use of radius function of the medial axis described in this paper.

6. CONCLUSION

A new algorithm LayTracks, that combines the strengths of existing approaches has been described for all-quad meshing. LayTracks uses the MAT, to decompose the region into tracks, to terminate the moving front of the quad elements and to provide the sizing function for geometry adaptive meshing. The algorithm is able to generate uniform quad mesh on a set of planar connected surfaces in 3D space. Quad mesh adapted to the geometric details has been implemented on single surface. The radius function is used to automatically track variations in geometric feature sizes and to adaptively size the elements inside the domain. The procedure ensures robust quad mesh generation with almost square type elements especially at the boundary. As the approach is based on the MAT it is also well suited for extension to 3D adaptive all hexahedral mesh generation.

REFERENCES

1. Cook RD, Malkus DS, Plesha ME. *Concepts and Application of Finite Element Analysis* (3rd edn). Wiley: New York, January 1989.
2. Owen SJ. A survey of unstructured mesh generation technology. *Proceedings of the 7th International Meshing Roundtable*, Dearborn, MI, USA, October 1998.
3. Lo SH. Generating quadrilateral elements on plane and over curved surfaces. *Computers and Structures* 1989; **31**(3):421–426.
4. Johnston BP, Sullivan Jr. JM, Kwasnik A. Automatic conversion of triangular finite element meshes to quadrilateral elements. *International Journal for Numerical Methods in Engineering* 1991; **31**:67–84.
5. Owen SJ, Staten ML, Canann SA, Saigal S. Advancing front quad meshing using local triangle transformations. *Proceedings of the 7th International Meshing Roundtable*, Dearborn, MI, USA, October 1998.
6. Blacker TD, Stephenson MB. PAVING: a new approach to automatic quadrilateral mesh generation. *International Journal for Numerical Methods in Engineering* 1991; **32**:811–847.
7. White DR, Kinney P. Redesign of the paving algorithm: robustness enhancements through element by element meshing. *Proceedings*, *6th International Meshing Roundtable*, Park City, UT, USA, Sandia National Laboratories, 1997; 323–335.
8. Baehmann PL, Wittchen SL, Shephard MS, Grice KR, Yerry MA. Robust geometrically-based automatic two-dimensional mesh generation. *International Journal for Numerical Methods in Engineering* 1987; **24**:1043–1078.
9. Talbert JA, Parkinson AR. Development of an automatic, two dimensional finite element mesh generator using quadrilateral elements and Bezier curve boundary definitions. *International Journal for Numerical Methods in Engineering* 1991; **29**:1551–1567.
10. Chae S, Jeong J. Unstructured surface meshing using operators. *Proceedings*, *6th International Meshing Roundtable*, Roundtable, 1997; 281–291.

11. Nowottny D. Quadrilateral mesh generation via geometrically optimized domain decomposition. *Proceedings*, *6th International Meshing Roundtable*, Park City, UT, USA, 1997; 309–320.
12. Joe B. Quadrilateral mesh generation in polygonal regions. *Computer Aided Design* 1995; **27**:209–222.
13. Gursoy HN. Shape interrogation by medial axis transform for automated analysis. *Ph.D. Thesis*, MIT, November 1989.
14. Patrikalakis M, Gursoy HN. Shape interrogation by medial axis transform. In *Advances in Design Automation*: *Computer Aided and Computational Design*, Ravani B (ed.). *Proceedings of the 16th ASME Design Automation Conference*, Chicago, September 1990, vol. I. ASME: NY, 1990; 77–88.
15. Gursoy HN, Patrikalakis NM. Automated interrogation and adaptive subdivision of shape using medial axis transform. *Advances in Engineering Software and Workstations* 1991; **13**(5/6):287–302.
16. Gursoy HN, Patrikalakis NM. An automatic coarse and fine surface mesh generation scheme based on MAT part I: algorithms. *Engineering with Computers* 1992; **8**:121–137.
17. Gursoy HN, Patrikalakis NM. An automatic coarse and fine surface mesh generation scheme based on MAT part II: implementation. *Engineering with Computers* 1992; **8**:179–196.
18. Tam TKH, Armstrong CG. 2D finite element mesh generation by medial axis subdivision. *Advances in Engineering Software* 1991; **13**:313–324.
19. CADfix. *FEGS Ltd*, http://www.fegs.co.uk/index.html.
20. Turbo Mesh. *4th International Meshing Roundtable*, October 16–17, 1995; 141, http://www.99main.com/diholm/.
21. Srinivasan V, Nackman LR, Tang JM, Meshkat SN. Automatic mesh generation using the symmetric axis transformation of polygonal domains. *Proceedings of IEEE* 1992; **80**(9):1485–1501.
22. Srinivasan V, Nackman LR. Voronoi diagram for multiply connected polygonal domains, I: algorithm. *IBM Journal of Research and Development* 1987; **31**(3):361–372.
23. Sampl P. Semi-structured mesh generation based on medial axis. *Proceedings*, *9th International Meshing Roundtable*, New Orleans, LA, USA, Sandia National Laboratories, October 2000; 21–32.
24. Quadros WR, Ramaswami K, Prinz FB, Gurumoorthy B. LayTracks: a new approach to automated quadrilateral mesh generation using MAT. *9th International Meshing Roundtable Meshing Conference*, New Orleans, LA, USA, October 2000.
25. Xiang-Yang L, Teng S-H, Ungor A. Simultaneous refinement and coarsening adaptive meshing with moving boundaries. *7th International Meshing Roundtable*, Dearborn, MI, USA, October 1998, 201–210.
26. Borouchaki H, George PL, Mohammedi B. Delaunay mesh generation governed by metric specification. *Finite Elements in Analysis and Design* 1997; **25**:85–109.
27. Borouchaki H, Hecht F, Frey PJ. Mesh gradation control. *Proceedings of 6th International Meshing Roundtable*, Park City, U.S.A., October 1997; 131–141.
28. Pascal F, Marechal JL. Fast adaptive quadtree mesh generation. *7th International Meshing Roundtable*, Dearborn, MI, U.S.A., October 1998; 211–224.
29. Aly K, Kallinderis Y, McMorris H. Adaptive hybrid grids for diverse industrial applications. *7th International Meshing Roundtable*, Dearborn, MI, U.S.A., October 1998; 167–184.
30. Price MA, Armstrong CG. Hexahedral mesh generation by medial surface subdivision: part I. *International Journal for Numerical Methods in Engineering* 1995; **38**(19):3335–3359.
31. Price MA, Armstrong CG. Hexahedral mesh generation by medial surface subdivision: part II. *International Journal for Numerical Methods in Engineering* 1997; **40**:111–136.
32. Robinson J, Haggenmacher GW. Element warning diagnostics. *Finite Element News*, June and August 1982.
33. Blum H. A transformation for extracting new descriptors of shape. *Models for the Perception of Speech and Visual Form*. The MIT Press: Cambridge, MA, 1967; 326–380.
34. Blum H. Biological shape and visual science (part I). *Journal of Theoretical Biology* 1973; **38**:205–287.
35. Sherbrooke EC, Patrikalakis NM, Wolter F. Note on differential and topological properties of medial axis transforms. *Graphical Models and Image Processing* 1996; **58**(6):547–592.
36. Zhu JZ, Zienkiewicz OC. A new approach to the development of automatic quadrilateral mesh generation. *International Journal for Numerical Methods in Engineering* 1991; **32**:849–866.
37. Quadros WR. Laytracks: an automatic quad mesh generator based on MAT. *M.Sc. Thesis*, Department of Mechanical Engineering, Indian Institute of Science, Bangalore, India, January 2001.
38. Kim D, Hwang I, Joopark B. Representing the Voronoi diagram of a simple polygon using rational quadratic Bèzier curves. *CAD* 1995; **27**(8):605–614.
39. Quadros WR, Ramaswami K, Prinz FB, Gurumoorthy B. Automatic geometry adaptive quadrilateral mesh generation using MAT. *Proceedings of ASME*, Pittsburgh, PA, USA, DETC, September 2001.

