INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng. **44**, 1317–1340 (1999)

Q-MORPH: AN INDIRECT APPROACH TO ADVANCING FRONT QUAD MESHING

S. J. OWEN[1,2,*], M. L. STATEN[2], S. A. CANANN[1,2] AND S. SAIGAL[1]

[1] *Department of Civil and Environmental Engineering, Carnegie Mellon University, Pittsburgh, PA, U.S.A.*
[2] *Ansys Inc. 275 Technology Drive, Canonsburg, PA, 15317, U.S.A.*

SUMMARY

Q-Morph is a new algorithm for generating all-quadrilateral meshes on bounded three-dimensional surfaces. After first triangulating the surface, the triangles are systematically transformed to create an all-quadrilateral mesh. An advancing front algorithm determines the sequence of triangle transformations. Quadrilaterals are formed by using existing edges in the triangulation, by inserting additional nodes, or by performing local transformations to the triangles. A method typically used for recovering the boundary of a Delaunay mesh is used on interior triangles to recover quadrilateral edges. Any number of triangles may be merged to form a single quadrilateral. Topological clean-up and smoothing are used to improve final element quality. Q-Morph generates well-aligned rows of quadrilaterals parallel to the boundary of the domain while maintaining a limited number of irregular internal nodes. The proposed method also offers the advantage of avoiding expensive intersection calculations commonly associated with advancing front procedures. A series of examples of Q-Morph meshes are also presented to demonstrate the versatility of the proposed method. Copyright © 1999 John Wiley & Sons, Ltd.

KEY WORDS: mesh generation; quadrilateral; advancing front; surface meshing; Q-Morph; paving

1. INTRODUCTION

Automatic meshing algorithms for three-dimensional surfaces have become valuable tools in the process of design and analysis. The resulting elements may be used as shell elements or as input to an automated tet or hex meshing algorithm. Because triangle meshing algorithms can be relatively simple to implement and tend to have convenient and provable mathematical properties, surface meshing algorithms including Delaunay[1–3] and advancing front[4–7] methods dominate much of the literature. In spite of this, many analysts prefer to use quadrilateral elements, noting their superior performance for various applications.[8] A smaller set of literature exists that describes quadrilateral meshing algorithms. Quadrilateral meshing does not have the convenient mathematical properties demonstrated by triangle meshing, and hence require more heuristic methods.

When the geometry of the domain is suitable, quadrilateral mapped meshing,[9] or sub-mapping[10] methods produce very high quality elements. These methods break the domain into a structured set of quadrilaterals, where all interior nodes have exactly four adjacent elements.

* Correspondence to: S. J. Owen, Ansys, Inc., 275 Technology Drive, Cannonsburg, PA 15317, U.S.A. E-mail: steve.owen@ansys.com

CCC 0029–5981/99/091317–24$17.50

Received 14 April 1998
Revised 15 June 1998

Where applicable, it is clear that these methods are faster and more reliable. Unfortunately, since only a limited class of problems may be easily resolved using mapping methods, more general 'unstructured' approaches must be formulated.

Unstructured quadrilateral meshing algorithms can, in general, be grouped into two main categories: *direct* and *indirect* approaches. With an indirect approach, the domain is first meshed with triangles. Various algorithms are then employed to convert the triangles into quadrilaterals. With a direct approach, quadrilaterals are placed on the surface directly, without first going through the process of triangle meshing.

1.1. Indirect methods

A simple approach to indirect quadrilateral mesh generation includes inserting a node at the centroid of each triangle dividing it into three quadrilaterals. This method guarantees an all-quadrilateral mesh, but a high number of irregular nodes are introduced into the mesh often resulting in poor element quality. An irregular node is an interior node that has more or less than four adjacent quadrilateral elements. An alternate method is to combine adjacent pairs of triangles to form a single quadrilateral. While the element quality may increase using this method, a large number of triangles may be left in the mesh.

The method of combining triangles may be improved, if care is taken in the order in which triangles are combined. In an effort to maximize the number of quadrilaterals, Lo[11] suggested several heuristic procedures for the order in which triangles may be combined. This resulted in a quad-dominant mesh containing a minimal number of triangles. Johnston *et al.*[12] later proposed additional local element splitting and swapping strategies to increase the number and quality of quads. Lee and Lo[13] developed an enhancement of Lo's strategy by including local triangle splitting. Additionally, an advancing front approach was used over the initial triangles. A set of fronts were defined consisting of the edges of triangles at the boundary of the domain. Triangles were systematically combined at the front, advancing towards the interior of the area. Each time a set of triangles were combined, the front was advanced. The front always defined the division between quadrilaterals already formed and triangles yet to be combined. With this technique, an all-quadrilateral mesh was guaranteed, provided the initial number of edges on the boundary was even.

Since all operations are local, indirect methods have the advantage of being very fast. Global intersection checks are not necessary as is required with advancing front direct methods. The drawback to indirect methods is that there are typically many irregular nodes left in the mesh. Even if few irregular nodes exist, there is no guarantee that the elements will align with the boundary—a desirable property for some applications. To reduce the number of irregular nodes, most indirect methods employ topological clean-up operations[14–16] as a post-processing step to meshing. These operations consist of various local topological modifications to the mesh with the objective of maintaining exactly or close to four elements adjacent to each node.

1.2. Direct methods

Many methods for direct generation of quad meshes have been proposed in the literature, and they may be divided into two main categories. The first are methods that rely on some form of decomposition of the domain into simpler, convex, or mappable regions. Geometry decomposition techniques include methods proposed by Baehmann *et al.*,[17] Talbert and Parkinson,[18]

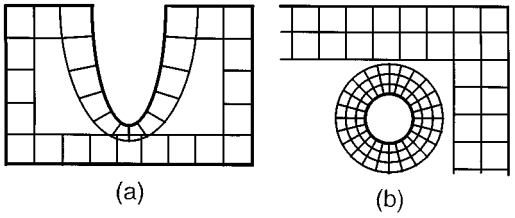

Figure 1. (a) First row of elements placed using paving algorithm illustrating interference of opposing elements; (b) large element size differences between opposing fronts often encountered in paving leading to poor meshes

Tam and Armstrong[19] and Joe,[20] among others. The second category consists of methods that utilize the advancing front method for element formation. Zhu *et al.*[21] proposed a quadrilateral meshing algorithm which starts with an initial placement of nodes on the boundary. Individual elements were then formed by projecting edges towards the interior. Two triangles were formed using traditional triangle advancing front methods[22] and then combined to form a single quadrilateral. Blacker and Stephenson,[23] presents a method, known as *paving*, where complete rows of elements were formed starting from the boundary and working towards the interior. Cass *et al.*[24] further developed paving, by generalizing the method for three-dimensional surfaces. White and Kinney[25] recently proposed enhancements to the paving algorithm suggesting individual placement of elements rather than complete rows.

Blacker and Stephenson,[23] described some of the characteristics of a quadrilateral mesh which are desirable for finite element analysis: '(a) *Boundary sensitive*: Mesh contours should closely follow the contours of the boundary. This characteristic is of particular importance since well-shaped elements are usually desirable near the boundary, (b) *Orientation insensitive*: Rotating or translating a given geometry should not change the resulting mesh topology. A mesh generated in a transformed geometry should be equivalent to the original mesh transformed, and (c) *Few irregular nodes*. This is a critical mesh topology feature because the number of elements sharing a node controls the final shape of the elements, even after smoothing. Thus a mesh with few irregular nodes, especially near the boundary where element shape is critical, is often preferred.'

Of the methods previously discussed, only mapping methods and paving provide all of these desirable features on a consistent basis. Since mapping methods are only applicable to specific geometry configurations, the paving method is particularly attractive.

While recognizing the desirable features of paving, there are some weaknesses that need to be addressed. Figure 1(a) illustrates the intersection problem common to all advancing front meshing schemes. As the elements advance towards the interior of the domain, intersection checks must be made to ensure that elements do not overlap. The detection and resolution of intersections can be very time consuming and also susceptible to floating point errors. If an intersection is missed, the paving algorithm has the potential to go on forever, *over-meshing* itself. The intersection problem is especially acute when dealing with three-dimensional surfaces.[24]

Another problem often encountered in paving occurs when fronts containing elements of greatly differing size must meet as in Figure 1(b). Heuristic decisions are made to make the fronts combine at this point, but in general poor element quality may result in this region.

This paper proposes an alternative to the traditional paving algorithm. The proposed Quad-morphing (Q-Morph) algorithm maintains the desirable features of paving while addressing some

of its weaknesses. Q-Morph can be categorized as an unstructured, indirect method that utilizes an advancing front algorithm to form an all-quad mesh. As an indirect method it is able to take advantage of local topology information from the initial triangulation. Although similar to Lee and Lo's[13] indirect method, Q-Morph utilizes additional features that help maintain well-aligned rows of elements and reduce the number of internal irregular nodes.

2. OUTLINE OF Q-MORPH ALGORITHM

The Q-Morph advancing front quadrilateral mesh generation algorithm is briefly outlined in the following steps:

1. Background mesh: The surface is first triangulated. This may be done using any surface triangulation method. Any sizing[26] or adaptivity information should be built into the background mesh. The local sizing for the final quadrilateral mesh will roughly follow that of the background mesh.
2. Front definition: The initial front is defined from the background mesh. Any edge in the triangulation that is adjacent to only one triangle becomes part of the initial front.
3. Front edge classification: Each edge in the front is initially sorted according to its *state*. The state of a front edge defines how the edge will eventually be used in forming a quadrilateral. Angles between adjacent front edges determine the state of an individual front. Front edges will be updated and reshuffled as the algorithm proceeds. Figure 2 shows the four possible states of a front, where the front edge is indicated by the bold line.
4. Front edge processing: Each front edge is individually processed to create a new quadrilateral from the triangles in the background mesh. Figure 3(a) shows front N_A–N_B in the triangulation ready to be processed. Front edges are handled differently according to their current state classification. As quadrilaterals are formed, the front is redefined and adjacent front edge states updated. The current front always defines the interface between quadrilateral elements in the final mesh and triangle elements in the initial triangle mesh. This process can be further subdivided into the following substeps:

   (a) Side edge definition: Using the front edge as the initial base edge of the quadrilateral, side edges are defined. Side edges may be defined by using an existing edge in the background mesh, by swapping the diagonal of adjacent triangles, or by splitting triangles to create a new edge. In Figure 3(b), side edge N_B–N_C shows the use of an existing edge, while the side edge N_A–N_D was formed from a local swap operation.
   (b) Top edge recovery: The final edge on the quadrilateral is created by an *edge recovery* process, which modifies the local triangulation using local edge swaps to enforce an edge between the two nodes at the ends of the two side edges. Edge N_C–N_D in Figure 3(c) was formed from a single swap operation. Any number of swaps may be required to form the top edge.
   (c) Quadrilateral formation: The final quadrilateral is formed by merging any triangles bounded by the front edge, and the newly created side edges and top edge as shown in Figure 3(d).
   (d) Local smoothing: The mesh is smoothed locally to improve both quadrilateral and triangle element quality as shown in Figure 3(e).

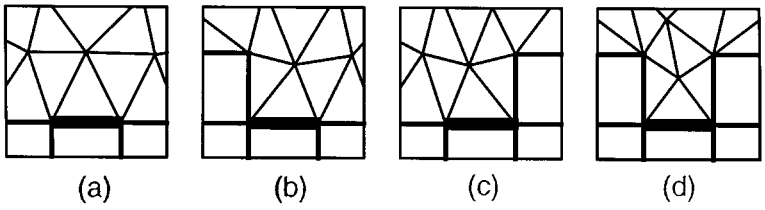

Figure 2. States of a front edge: (a) state 0–0; (b) state 1–0; (c) state 0–1; (d) state 1–1

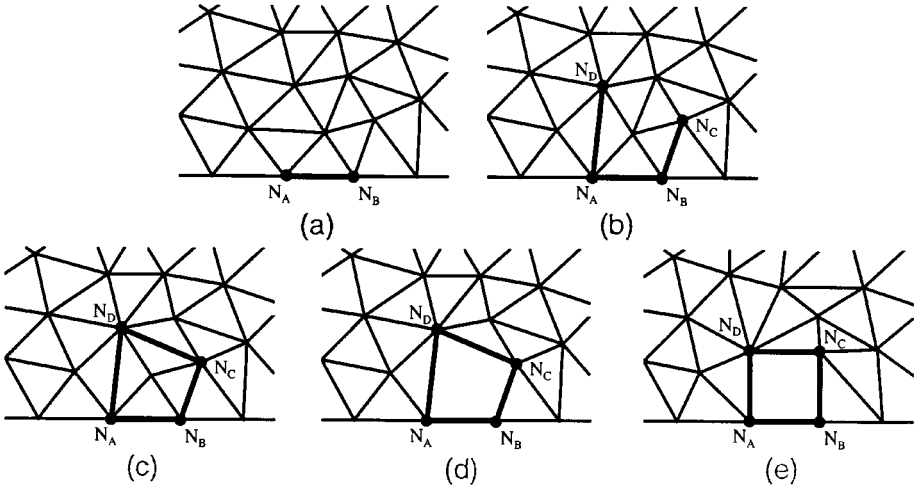

Figure 3. Steps demonstrating process of generating a quadrilateral from Front N_A–N_B: (a) initial front; (b) side edge definition; (c) top edge recovery; (d) quadrilateral formation; (e) local smooth

(e) Local front reclassification: The front is advanced by removing edges from the front that have two quadrilateral adjacencies and adding edges to the front that have one triangle and one quadrilateral adjacency. New front edges are classified by *state*. Existing fronts that may have been adjusted in the smoothing process are reclassified.

Front edge processing continues until all edges on the front have been depleted, in which case an all-quadrilateral mesh will remain, assuming an even number of initial front edges. When an odd number of boundary intervals is provided, a single triangle must be generated, usually towards the interior of the mesh.

5. Topological clean-up: Element quality is improved by performing local quadrilateral transformations in an attempt to improve the individual edge valences at the nodes of the mesh.
6. Smoothing: A final smoothing pass is performed further improving the element qualities.

3. IMPLEMENTATION

The following is a description of some of the implementation specifics of the Q-Morph algorithm.

3.1. Background mesh triangulation

While maintaining that any triangulation method may be successfully used as a background mesh, the final quadrilateral mesh may differ somewhat based on the initial node placement. To adequately represent the geometry with quadrilaterals, it is necessary that the initial triangles be of a sufficient density to capture model features and surface curvature. Triangles with excessively high aspect ratio may also result in poor quadrilaterals. In practice, a parametric space, advancing front triangulation algorithm[27] that takes into account surface curvature[26] and model details[28] is used. It is conceivable that a more judicious initial node placement scheme that places nodes in orthogonal alignment to the boundary would be advantageous to the algorithm, however this has not yet been investigated.

3.2. Front definition and classification

After initially triangulating the domain, the first step in the advancing front process is to define an initial set of front edges. The initial front is defined by examining edges in the background triangle mesh for edges with only one triangle adjacency. Edges should be oriented consistently so that a traversal of the edges will result in a counter-clockwise traversal of the domain. Front edges interior to the domain, or interior loops, should be oriented clockwise. Correct orientation for the front edges can be naturally achieved provided all triangles in the background mesh have a consistent counter-clockwise definition. Maintaining a consistent orientation for the edges on the front assists in local topology inquiries such as determining element adjacencies and front adjacencies.

The state of a front edge is determined by computing the angle at the nodes on either end of the edge with each of its adjacent front edges. Practically, the state of a front edge is defined by two bits, the first representing the state at the *left* node and the second, the state at the *right* node. If the angle at either node is less than a specified tolerance, (currently set to $3\pi/4$) the node bit is set (1), otherwise it is unset (0).

Edges are placed on one of four *state* lists as shown in Figure 2. Classifying front edges according to states serves two purposes. First, it defines which edges must be generated before a complete quadrilateral can be formed. Side edges must be defined only at the side of the front where the state bit has not been set. Second, it prioritizes which fronts will be processed first. Front edges in state 1–1 are given first priority followed by edges in states 0–1 and 1–0, followed by edges in state 0–0.

3.2.1. Angle calculation. In order to classify the state of a front edge, angle calculations must be done quite frequently. Computation of angles on an arbitrary three-dimensional surface can be computationally expensive. Cass *et al.*[24] suggests a method, which involves direct geometric evaluation of surface normals. A less expensive method, taking advantage of triangles in the background mesh is described here. The angle, α_k at node N_k on the front may be defined by summing the angles between triangle edges adjacent to N_k,

$$\alpha_k = \sum_{i=1}^{n} \alpha_i \text{ where } \alpha_i = \cos^{-1}\left(\frac{\mathbf{A}_i \cdot \mathbf{B}_i}{\|\mathbf{A}_i\| \cdot \|\mathbf{B}_i\|}\right) \tag{1}$$

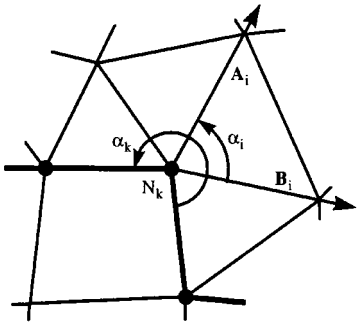

Figure 4. Definition of angle α_k at a node on the front

with vectors $\mathbf{A}_i$ and $\mathbf{B}_i$ defined by the node N_k and two edges of an adjacent triangle shown in Figure 4 and n representing the total number of triangles adjacent to N_k.

3.3. *Front edge processing*

Front edges are processed one at a time to form quadrilaterals from the background mesh. A front edge is *popped* from one of the four state lists, drawing from the higher states first. Priority is also given to the lowest *level* edge on the list. Edges in level zero are those on the initial front; level one are those on the front after the first row of quadrilaterals have been placed; level two after the second row; and so on. This ensures that an entire row of quadrilaterals will be placed before starting a new row.

Where large transitions are required, experience has shown that placing smaller quads first generally improves mesh grading. To accomplish this, it is sometimes necessary to select short, higher level fronts before selecting longer lower level fronts. The criteria used for selecting the next front to be processed is, therefore, based not only the current state and level of the front but also on its size.

3.3.1. Side edge definition. The current state of a front edge determines how the edge is processed. Front edges in state 0–0, 1–0 and 0–1 must first define either one or two *side* edges. A side edge may be formed in one of three ways: (1) an existing edge in the background mesh may be used, (2) the diagonal between two adjacent triangles may be swapped, or (3) an edge may be created by splitting a pair of triangles.

Figure 5 shows a situation in which an existing edge is used. A new side edge is to be defined at node N_k, a node on the front between edges E_{F1} and E_{F2}. The ideal vector $\mathbf{V}_k$ for the new side edge is defined by bisecting the vectors formed by E_{F1} and E_{F2}. Angles θ_i are computed between $\mathbf{V}_k$ and all edges, E_i, of triangles sharing node N_k. The edge with the smallest angle θ is selected as the candidate side edge. The edge is selected, provided θ is less than a constant ε (currently defined as $\pi/6$). Edge E_2 in Figure 5 is selected as the side edge in this situation.

When there is no angle θ_i less than ε, one of two options may be used. The opposite edge, E_0 in Figure 6 may either be swapped or split. The swap option is used if the angle β between $\mathbf{V}_k$ and $\mathbf{V}_m$ is less than ε. The split option is performed if $\beta > \varepsilon$ or the resulting length of E_k from a swap is

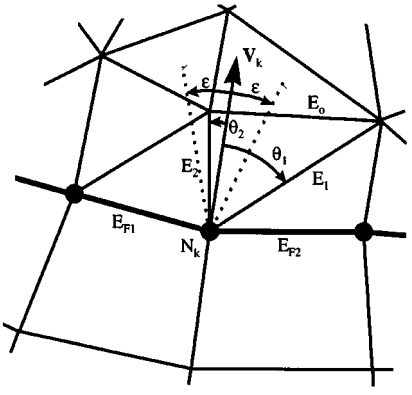

Figure 5. Side edge selection

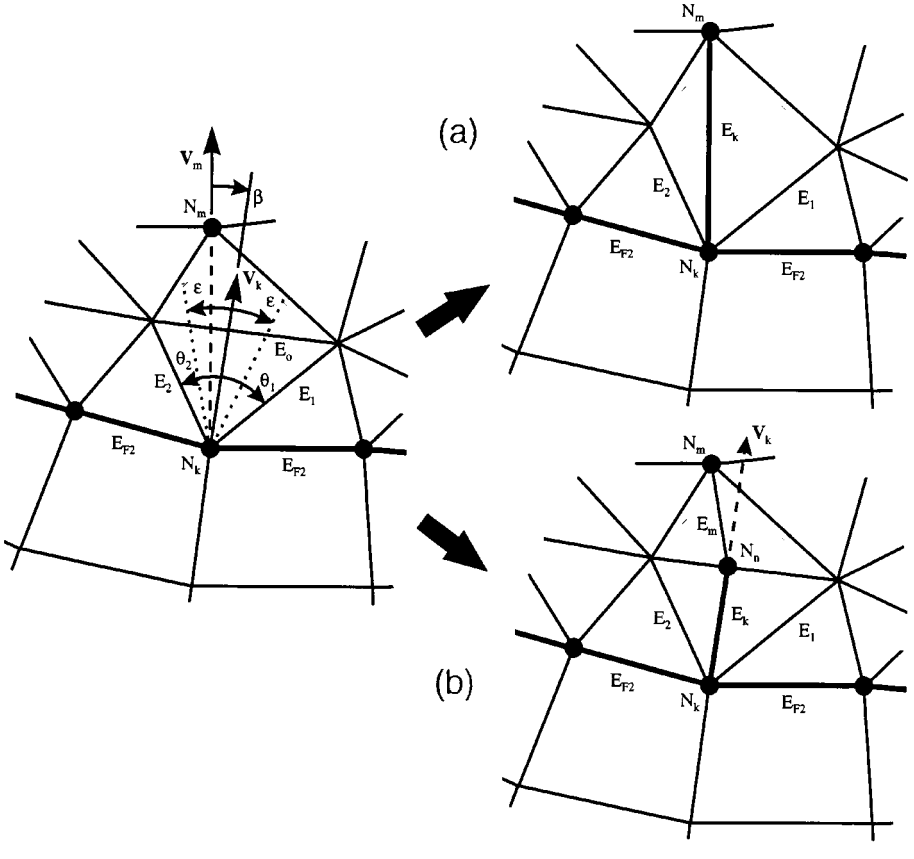

Figure 6. Side edge creation: (a) swap; (b) split

Algorithm 1. Edge recovery

```
1.  LET S be the line segment from N_C to N_D
2.  LET Λ(S) be a list of edges E_i that are intersected by S (see Algorithm 2)
3.  FOR EACH E_i ∈ Λ(S)
4.      LET T(E_i) be the set of 2 triangles adjacent E_i
5.      LET T^-1(E_i) be the set of 2 triangles where the diagonal edge E_i has
            been swapped.
6.      IF area of both triangles in T^-1(E_i) > 0 THEN
7.          Form T^-1(E_i)
8.          Delete E_i from Λ(S)
9.          LET E_j be the edge common to both triangles in T^-1(E_i)
10.         IF E_j intersects S add E_j last on Λ(S),
11.     ELSE,
            Place E_i last on Λ(S)
12. NEXT E_i on Λ(S)
```

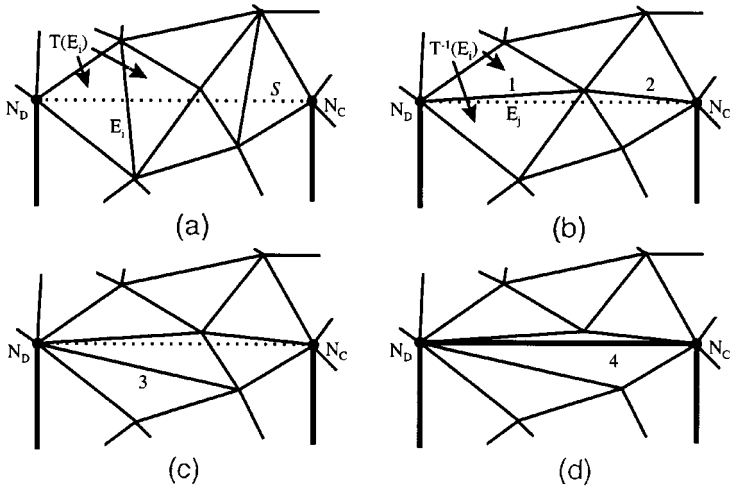

Figure 7. Edge recovery process: (a) initial triangulation; (b) swap 1, 2; (c) swap 3; (d) swap 4 edge recovered

excessively long compared to E_{F1} and E_{F2}. In this latter case, a new node N_n is defined, splitting edge E_0 at the intersection of vector $\mathbf{V}_k$ and edge E_0. Edges E_k and E_m are also added to the triangle mesh, splitting the two triangles adjacent to edge E_0. Edge E_k is then used as the side edge of the prototype quad. The following shows a summary of the criteria for selection or creation of edge E_k, to be used as a side edge in the new quadrilateral:

$$E_k = \begin{cases} E_i & \text{for } \theta_i < \varepsilon \\ \text{swap} \Rightarrow \overline{N_k N_m} & \text{for } \beta < \varepsilon \text{ and } \|N_k N_m\| < \sqrt{3}\,\dfrac{\|E_{F1}\| + \|E_{F2}\|}{2} \\ \text{split} \Rightarrow \overline{N_k N_n} & \text{otherwise} \end{cases} \tag{2}$$

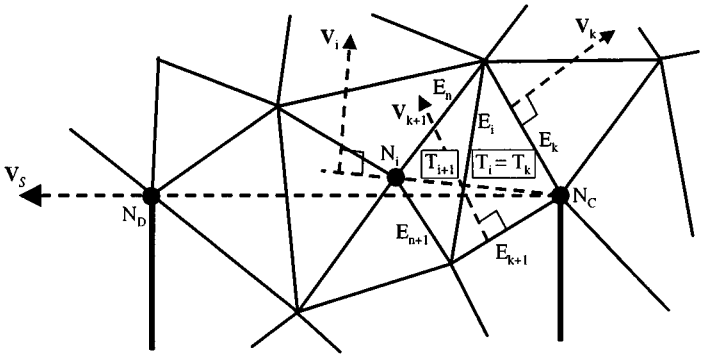

Figure 8. Example of procedure described in Algorithm 2

Algorithm 2. Formation of $\Lambda(S)$

1. LET $T(N_C)$ be the ordered set of ccw triangles and quads adjacent N_C, $T_k(N_C) \in T(N_C)$
2. LET $E(N_C)$ be the ordered set of ccw edges adjacent N_C; $E_k(N_C) \in E(N_C)$, where $E_k(N_C)$ and $E_{k+1}(N_C)$ are on $T_k(N_C)$
3. LET $\mathbf{V}_k$ be the vector normal to edge $E_k(N_C)$ and tangent to surface
4. LET $\mathbf{V}_S$ be the vector from N_C to N_D
5. FOR EACH $T_k(N_C) \in T(N_C)$
   IF $\mathbf{V}_s \cdot \mathbf{V}_k > 0$ and $\mathbf{V}_s \cdot \mathbf{V}_{k+1} < 0$, LET $T_i(E_i) = T_k(N_C)$
6. LET E_i be the edge opposite N_C on $T_k(N_C)$
7. IF E_i is not on front THEN, Add E_i to $\Lambda(S)$, ELSE *fail*
8. WHILE not *done*
9. LET $T_{i+1}(E_i)$ be the triangle adjacent E_i where $T_{i+1}(E_i) \neq T_i(E_i)$
10. IF N_D is on $T_{i+1}(E_i)$, THEN *done*
11. $T_i(E_i) = T_{i+1}(E_i)$
12. LET N_i be the node opposite E_i on $T_i(E_i)$
13. LET $\mathbf{V}_i$ be the vector normal to segment $N_C N_i$ and tangant to surface
14. LET E_n be the next ccw edge on $T_i(E_i)$ from E_i
15. LET E_{n+1} be the next cw edge on $T_i(E_i)$ from E_i
16. IF $\mathbf{V}_S \cdot \mathbf{V}_i < 0$, THEN $E_i = E_n$, ELSE $E_i = E_{n+1}$
17. IF E_i is not on front THEN, Add E_i to $\Lambda(S)$, ELSE *fail*
18. CONTINUE

3.3.2. Top edge recovery. Once the base and the two sides of the quadrilateral have been formed, the next step is to define the top edge. This is done by *recovering* the edge between the end nodes of the two sides. Edge recovery is a technique used commonly in boundary constrained Delaunay triangle meshing, and presented independently in the literature by Jones[29], Sloan[30] and George *et al.*[31]

If nodes N_C and N_D in Figure 3(b) are existing nodes in the triangle surface mesh inside or on the current front, the method for recovering edge N_C–N_D from the triangulation can be described

by Algorithm 1. An example of an edge recovery process is shown in Figure 7. The triangulation before recovery is shown at the top left with the successive swaps numbered. In this example a total of four local swaps were required to recover the edge N_C–N_D from the triangulation.

One of the critical aspects of the edge recovery procedure is the determination of $\Lambda(S)$ in step 2 of Algorithm 1. $\Lambda(S)$ is the list of edges that intersect the line segment S. This can be accomplished by taking advantage of the local triangle topology, avoiding expensive global intersection calculations. If N_C and N_D are not already connected by an existing edge in the triangulation, Algorithm 2 may be used for defining $\Lambda(S)$. Algorithm 2 first locates the triangle adjacent to N_C containing segment S. It then traverses from one triangle to the next adding edges to $\Lambda(S)$ until it finally terminates at N_D. Figure 8 shows an example of the procedure described in Algorithm 2.

It should be noted that edge E_i in Algorithm 2 cannot be an edge in the current front. Steps 7 and 17 indicate that during the formation of $\Lambda(S)$, if E_i is an edge on the front, then Algorithm 2 will fail. In the event of a failure, the current front is placed back on its appropriate state list and an alternate front is selected for processing.

In some cases segment S may intersect another node between N_C and N_D. This special case can be detected when the result of the dot product calculation in steps 5 or 16 of Algorithm 2 is within a floating point tolerance of zero. In practice, the node intersected can be moved out of the way using a simple Laplacian smoothing algorithm.[32] Algorithm 2 must then be restarted. Alternatively, the intersected node can be perturbed slightly from its current location normal to the direction $\mathbf{V}_S$ in the plane of the surface.

3.3.3. 3D edge recovery. Algorithm 2 assumes a planar domain. Modifications must be made to ensure that the algorithm will operate correctly on a three-dimensional surface. Specifically, the dot product calculations of steps 5 and 16 must be performed on vectors in a plane that is tangent to the surface. The tangent plane can be approximated from the neighbouring triangles. For example, the tangent plane normal $\mathbf{P}_i$ at edge E_i can be estimated from the average normal vector of triangles $T_i(E_i)$ and $T_{i+1}(E_i)$. The dot product calculation in step 16 can then be replaced as:

$$((\mathbf{P}_i \times \mathbf{V}_S) \times \mathbf{P}_i) \cdot ((\mathbf{P}_i \times \mathbf{V}_i) \times \mathbf{P}_i) < 0 \tag{3}$$

Step 5 can be modified in a similar manner. An approximated tangent plane normal, $\mathbf{P}_C$, at N_C can be defined as the average normal vector of the triangles in $T(N_C)$. The dot product calculation can then be replaced as:

$$((\mathbf{P}_C \times \mathbf{V}_S) \times \mathbf{P}_C) \cdot ((\mathbf{P}_C \times \mathbf{V}_k) \times \mathbf{P}_C) > 0 \quad \text{and} \quad ((\mathbf{P}_C \times \mathbf{V}_S) \times \mathbf{P}_C) \cdot ((\mathbf{P}_C \times \mathbf{V}_{k+1}) \times \mathbf{P}_C) > 0 \tag{4}$$

Equations (3) and (4) have proven to be reliable provided the background mesh triangles reasonably represent the underlying surface geometry. In rare cases where overlapping or inverted triangles would be created, the current front is placed back on its state list and an alternate front is selected to process.

3.4. Quadrilateral formation

The quadrilateral is formed from the edge on the front, two side edges, and recovered top edge. Before forming the quadrilateral, the triangles contained within the four edges must first be deleted. This can be accomplished with a procedure that starts with the triangle adjacent the front

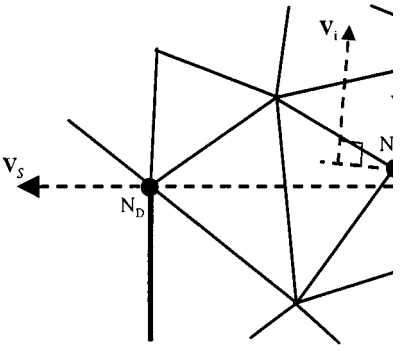

Figure 9. Definition of edge length l_D at node on front N_k

edge and recursively advancing to adjacent triangles deleting them as it proceeds. Unused nodes and edges are also removed. The recursion continues until the top or side edges of the prototype quadrilateral are encountered.

3.5. Local smoothing

Smoothing is an important part of the Q-Morph algorithm. Node locations local to the new quadrilateral are readjusted to improve element shape. This must be accomplished before processing the next front, as smoothing angles between adjacent fronts will affect the front *states* and hence the final topology of the quadrilateral mesh. In practice, any node on the new quadrilateral and any node connected by an edge are smoothed. Nodes on the front must be handled differently when compared to those behind or ahead of the front.

3.5.1. Interior smoothing. For nodes not located on the current front, a simple Laplacian[32] smooth is adequate. With this method, the node is placed at the centroid of its surrounding nodes. Alternatively, a modified length weighted Laplacian smooth for interior nodes can be used as suggested by Blacker and Stephenson.[23] This procedure, although slightly more expensive, can be beneficial.

3.5.2. Front smoothing. Since it is at the front where the new quadrilateral elements are formed, it is more critical at these nodes that the smoothing produce well proportioned quadrilaterals. A modified form of the smoothing process suggested by Blacker and Stephenson[23] is used for the *row* nodes defined in that reference. These are nodes connected to exactly two adjacent quadrilaterals at the front. The smoothing process presented by Blacker and Stephenson involves an isoparametric smooth,[33] followed by corrections for squareness and angle smoothness. For cases where large transitions may be involved, it is useful to take advantage of sizing information provided by the triangles ahead of the front. As a result, an improved transition can be achieved.

Let l_D be the length of the edge N_k–N_j where N_k is the node on the front to be smoothed as shown in Figure 9. Where a very large transition in element size is required, l_D can be defined as the average length of any edge connected to N_k. For smaller transitions, fewer irregular nodes will

be created if equation (5) is used. Let n be the number of nodes ahead of the front connected to N_i, then l_D can be defined as an average of edge lengths on adjacent quadrilaterals and edges ahead of the front as follows:

$$l_D = \frac{\|N_{k-1} - N_{j-1}\| + \|N_{j-1} - N_j\| + \|N_{k+1} - N_{j+1}\| + \|N_{j+1} - N_j\| + \sum_{i=1}^{n} \|N_k - N_{ti}\|}{4 + n} \quad (5)$$

The method used for computing l_D is decided purely on heuristics. For the current implementation, if the ratio of largest to smallest edge length, t_r, on the boundary is less than 2·5, then the smoothing method proposed by Blacker and Stephenson[23] is used unmodified. This method is preferred since it tends to produce the fewest number of irregular nodes. As t_r increases, it is necessary to introduce irregular nodes so that a smooth transition may be afforded. Equation (5) is used for l_D when t_r is greater than 2·5, and the average size of adjacent edges is used when t_r is greater than 20.

3.5.3. Smoothing adjustment. When smoothing nodes at the front, as a result of improving the quadrilaterals, it is possible that the triangles immediately ahead of the front become inverted. While the Q-Morph algorithm does not require triangles to be near equilateral, it does rely on the fact that all triangles are uninverted throughout the meshing process. An inverted element is one whose normal is opposite that of the surface normal and the elements immediately adjacent to it. It typically manifests itself as a region where elements appear to be overlapping. To ensure that this does not occur, triangles and quadrilaterals neighbouring the smoothed node must be checked for consistent normals. In the case of an inverted element, an adjustment must be made to the new node location. The node location can be adjusted incrementally on a vector from the old location to the new location until all neighbouring elements are no longer inverted.

3.6. Local update and reclassification of fronts

Once a new quadrilateral has been formed it becomes necessary to update the current list of fronts. To do this, the four edges on the new quadrilateral are examined. Any edge that is adjacent to exactly two quadrilaterals or is on a boundary is removed from its appropriate front state list. Edges adjacent to a triangle must be added to the state lists. New fronts are added to the top of their respective state lists. This ensures that the next front to be processed is typically immediately adjacent the last quadrilateral formed.

In addition to the edges on the current quadrilateral, other nearby edges on the front may need to be updated. By smoothing nodes on the front, angles between adjacent fronts may have been changed; perhaps enough to move the front into another state. To facilitate the reclassification of the front edges during the local smoothing process, any edge on the front connected to a node that is smoothed is marked. Once smoothing is complete, the angle at the front at each end of a marked front edge can be recomputed and the front edge reclassified according to its new state.

3.7. Closing the front

With any advancing front method, a facility must be provided for detecting when opposing fronts meet or intersect. It must also provide some process for merging or *closing* the fronts to define a continuous mesh. Blacker and Stephenson[23] and Cass *et al.*[24] propose a method for

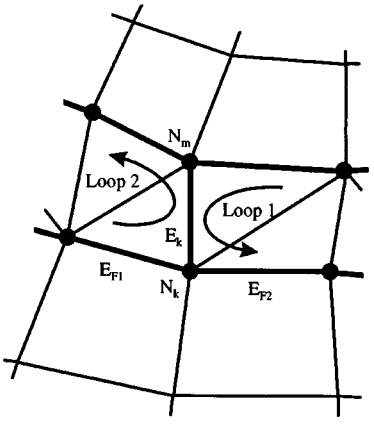

Figure 10. Selection of side edge forming of new front loop

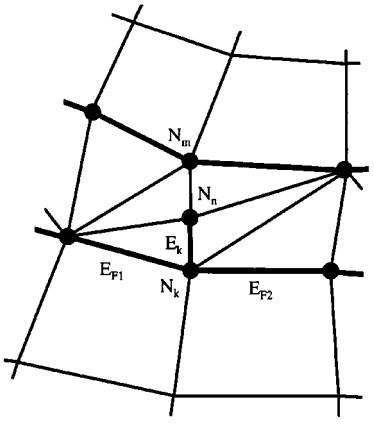

Figure 11. Splitting of side edge to maintain even loop

detecting when a front is to be closed by computing intersections of combinations of edges on the current front. Q-Morph avoids these intersection calculations by naturally handling the interference of opposing fronts by taking advantage of the background triangle mesh topology.

When defining a new side edge, the opposite node, N_m, as shown in Figure 10, may lie on an opposing front. Edge E_k may have been selected from the existing triangles as in Figure 5, or from a swap operation as in Figure 6. In either case, E_k can only be used if the number of edges on each resulting front loop is even. A front loop may be defined as all edges on a front comprising a continuous unbroken ring. Any number of loops may be active at a given time in the process of meshing. In order to ensure an all-quadrilateral mesh it is required that any individual loop be comprised of an even number of edges. Selecting an edge to be used as a new side edge may result in the formation of a loop with an odd number of front edges. To avoid this occurrence, the potential number of front edges on the new loop about to be formed is first determined. If the

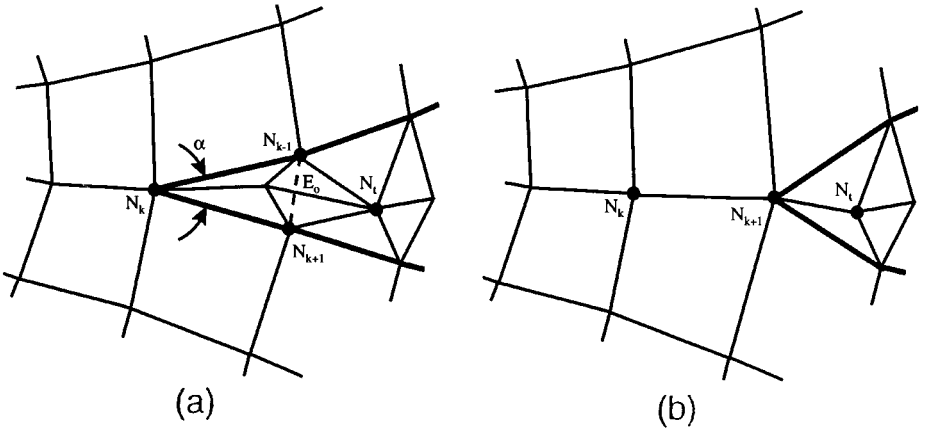

Figure 12. Seaming operation: (a) front to be seamed; (c) seam closed

number of edges is even, then the selection is made and a new loop is defined. If the number of edges on the new loop is odd, no connection is made. Instead the edge, E_k, is split creating a new node, N_n, as shown in Figure 11. This permits a subsequent side selection operation to define an even number of front edges on adjacent loops.

If the side edge is to be created from a swap or split operation, as in Figure 6, the edge E_0 should first be checked to see if it is part of the opposing front. Since swapping or splitting E_0 would destroy the continuity of the front, the operation should not be performed. For this reason, it is advantageous, when N_m is on an opposing front to allow for a larger value of ε. This increases the chances of selecting an existing edge and closing the front.

3.8. Seams

When the angle, α, between two adjacent edges on the front is small, then a seaming operation is performed. Although paving[23] incorporates a seaming operation, it must be defined within the context of the Q-Morph algorithm, in order to account for triangles ahead of the front. In addition to angle α, the criteria for seaming is also based on the number of quadrilateral elements, n_Q, adjacent the node to be seamed. Blacker and Stephenson[23] proposes that a node be seamed if:

$$\begin{Bmatrix} \alpha < \varepsilon_1 \text{ for } n_Q \geqslant 5 \\ \alpha < \varepsilon_2 \text{ otherwise} \end{Bmatrix} \quad \text{where } \varepsilon_1 < \varepsilon_2 \tag{6}$$

To accomplish the seam, nodes N_{k-1} and N_{k+1}, shown in Figure 12, must be merged. Let N_k be the node on the front whose angle, α, satisfies equation (6). The temporary edge, E_0, connecting N_{k-1} and N_{k+1}, if not already part of the background mesh, is first recovered using Algorithm 1 above. Knowing E_0, its adjacent triangle comprising nodes N_{k-1}, N_{k+1}, N_t, can be determined. With this information, all triangles and nodes bounded by the quadrilateral composed of nodes N_{k-1}, N_k, N_{k+1}, N_t can be deleted. Finally nodes N_{k-1} and N_{k+1} can be merged at a location midway between their initial positions. Local smoothing is then performed, followed by an update of the states of any adjacent fronts that may have changed. In rare occasions, the new

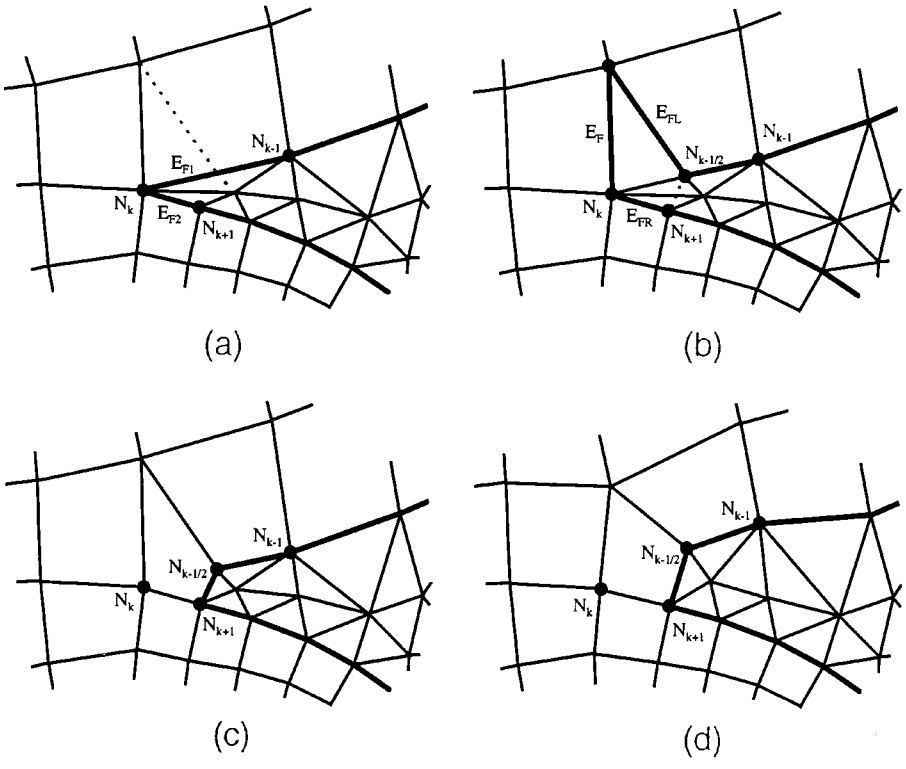

Figure 13. Transition seam operation: (a) split larger of E_{F1} and E_{F2}; (b) define E_F as front in state 1–1; (c) from now transition quad; (d) smooth and reclassify fronts

location of node N_{k+1} may result in one or more inverted elements. In this case, an optimization based smoothing algorithm[34] is employed which adjusts the node location with the objective of improving a local shape metric for neighbouring elements.

Another operation described by Blacker and Stephenson[23] is the transition seam. This is required when there is a large difference in size between adjacent fronts. In Figure 13 E_{F1} and E_{F2} are the edges on the front adjacent to N_k. If the ratio of lengths between E_{F1} and E_{F2} is greater than 2·5, then a transition seam operation is performed. The longer of the two edges, E_{F1} and E_{F2}, is first split at its midpoint adding node $N_{k-1/2}$ or $N_{k+1/2}$. In Figure 13(b), E_{F1} is split, dividing its adjacent triangle and quadrilateral as shown. Edges E_F, E_{FL}, and E_{FR} can then be defined as front edges. With this new configuration, edge E_F can be processed as a front in state 1–1, requiring only the recovery of the top edge between $N_{k-1/2}$ and N_{k+1} as in Figure 13(c). Finally, the transition seam is completed after local smoothing and updating of the front as shown in Figure 13(d).

3.9. Transition split

An operation useful for improving transitions is shown in Figure 14. Although similar to the transition seam operation in Figure 13, it is applicable when $\alpha > \varepsilon_1$ or $\alpha > \varepsilon_2$ (see equation (6)).

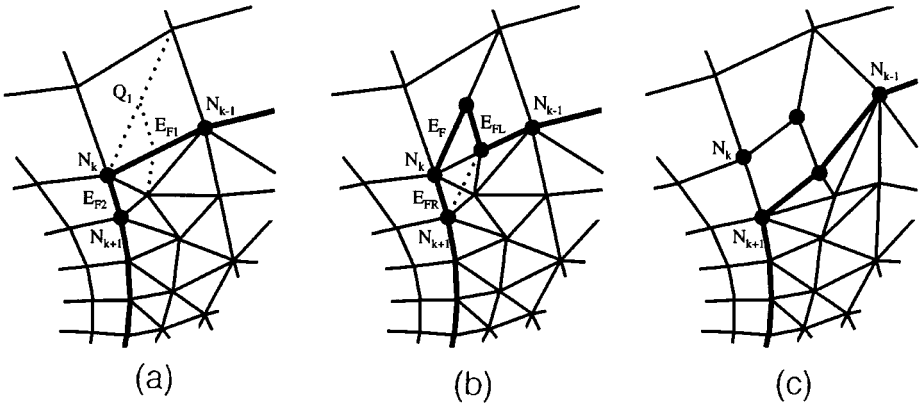

Figure 14. Transition split operation: (a) split Q_1; (b) define E_F as front in state 1–1; (c) form transition quad and smooth

The transition split operation is performed when the ratio of lengths between E_{F1} and E_{F2} is greater than 2·5. Let Q_1 be the quadrilateral adjacent the longer of E_{F1} or E_{F2}. Q_1 is split into two quadrilaterals and a single triangle, as shown in Figure 14(a). Front E_{F1} in Figure 14(a) is split at its midpoint also splitting its adjacent triangle, and adding an additional node to the centroid of Q_1. As a result, new front edges, E_F, E_{FL} and E_{FR}, can be defined as shown in Figure 14(b). Similar to the transition seam, E_F can now be defined as a front in state 1–1 and processed to create a new quadrilateral. Figure 14(c) shows the configuration after smoothing and reclassification of fronts.

3.10. Topological clean-up and smoothing

Once all of the front edges have been processed and an all-quadrilateral mesh is generated, it is often beneficial to perform local topological clean-up operations to decrease the number of irregular nodes. Although Q-Morph attempts to minimize the number of irregular nodes, they may, as a necessity, be introduced as a result of non-orthogonal boundaries or from element size transitions. Irregular nodes may also be introduced when the local nodal density and connectivity, provided by the background triangle mesh, is insufficient to generate equilateral quadrilaterals. Many of these irregular nodes can be eliminated through local topological clean-up. Topological clean-up modifies the connectivity of the quadrilaterals through a series of single-step operations including edge swaps, face opens, face closes, and two-edge node removals/insertions.[14–16] In addition, these single operation modifications can be combined into multi-step modifications[14] to further decrease the number of irregular nodes. By reducing irregular nodes through topological clean-up, the mesh contours can more closely follow the contours of the boundary.

The final smoothing step involves a limited number of iterations of a constrained Laplacian smoothing algorithm. Each node is moved to the centroid of its neighbours only if an improvement in element shape metric[13] would result. In situations where Laplacian smoothing produces poor results, an optimization based smoothing[34] operation may be performed.

4. EXAMPLE PROBLEMS

Five example problems, shown in Figures 15–19, demonstrate various features of the Q-Morph algorithm. The first example, shown in Figure 15, demonstrates the progression of the Q-Morph algorithm on a simple planar domain with two holes. Figure 15(a) shows the initial background triangle mesh before Q-Morph begins. In this case an advancing front triangle mesher[35] was used to create the triangles. The method used for triangulation is unimportant, inasmuch as the appropriate nodal density is provided. Figures 15(b)–15(g) show the progression of the algorithm as each successive layer of elements is completed. Figure 15(c) shows an additional layer of small elements meshed on the internal circle loop before meshing the larger elements of the outer loop. To improve element transitions, provision is made in Q-Morph to mesh loops with smaller elements before those with larger elements. The mesh is completed in Figure 15(h) after a final pass of clean-up and smoothing.

Figures 16 and 17 compare Q-Morph against Lee and Lo's[13] quad meshing algorithm, which uses an indirect method, coupled with an advancing front scheme to combine triangles into quadrilaterals. The toroidal surface of Figure 16 is composed of four surface patches represented as rational B-splines. Q-Morph utilizes projection and geometric evaluation routines as part of the local and final smoothing procedures to maintain nodal locations on the three-dimensional surface. Both Figures 16(a) and 16(b) were generated using the same initial triangle mesh as well as the same clean-up and smoothing procedures. Despite using an advancing front scheme, Lee's algorithm shown in Figure 16(b), has difficulty maintaining well-aligned rows of elements introducing many irregular internal nodes. Figure 17 further illustrates the ability of the

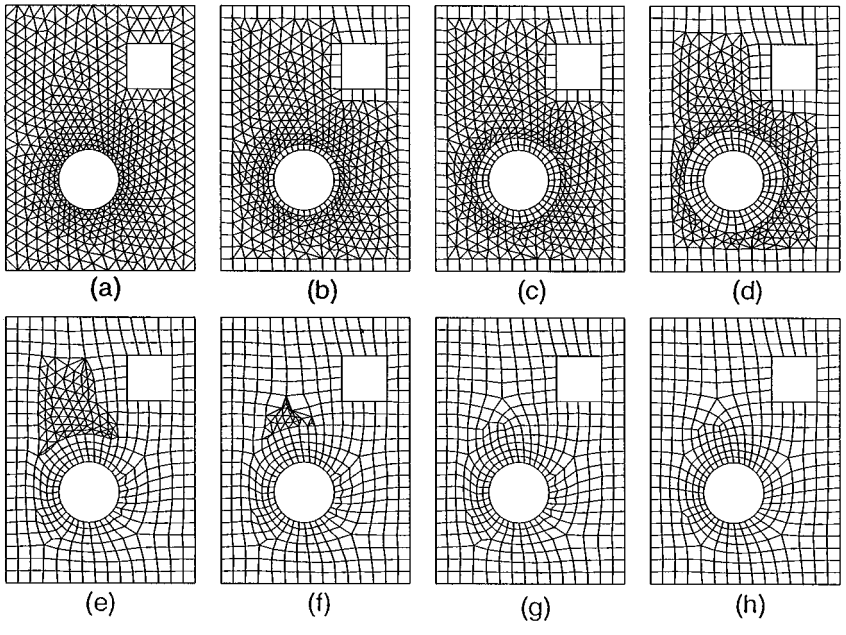

Figure 15. Progression of Q-Morph: (a) initial; (b) 1 layer; (c) 2 layers; (d) 3 layers; (e) 4 layers; (f) 5 layers; (g) 6 layers; (h) clean-up and smoothing

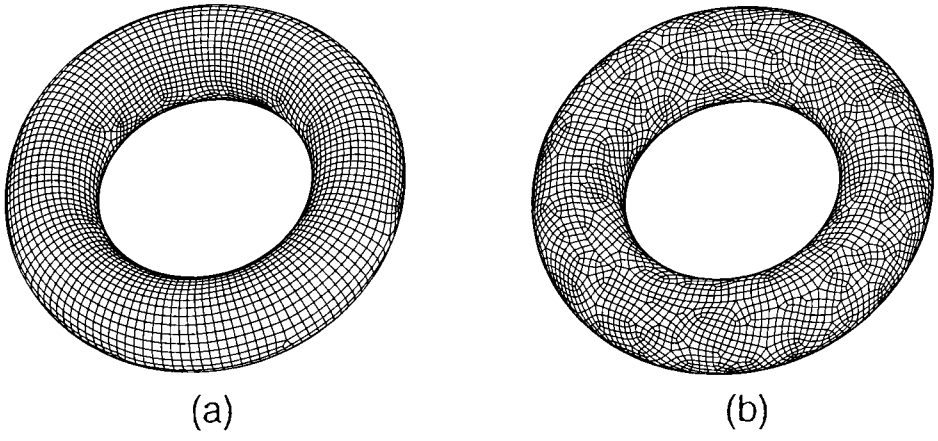

Figure 16. Results of Q-Morph compared with Lee's[13] advancing front indirect method on toroidal surface: (a) Q-Morph; (b) Lee's algorithm

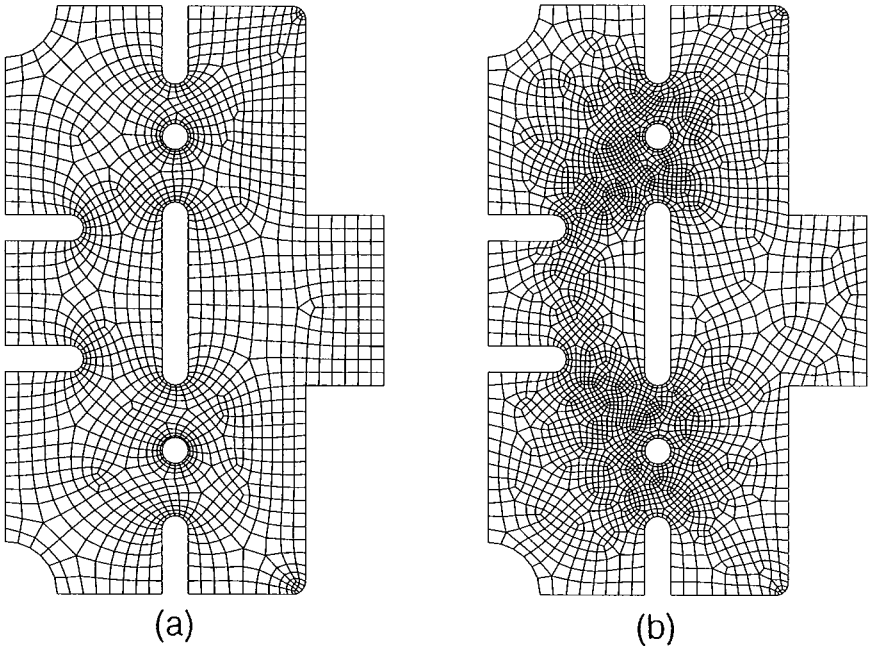

Figure 17. Comparison of Q-Morph with Lee's Algorithm illustrating element boundary alignment: (a) Q-Morph; (b) Lee's algorithm

Q-Morph algorithm to generate well-aligned rows of elements parallel to a complex domain boundary, while still maintaining the required element size transitions.

Figure 18 demonstrates the use of Q-Morph with a planar surface requiring a high degree of transition. Figure 18(a) shows the partially completed quad mesh with two layers of quads placed.

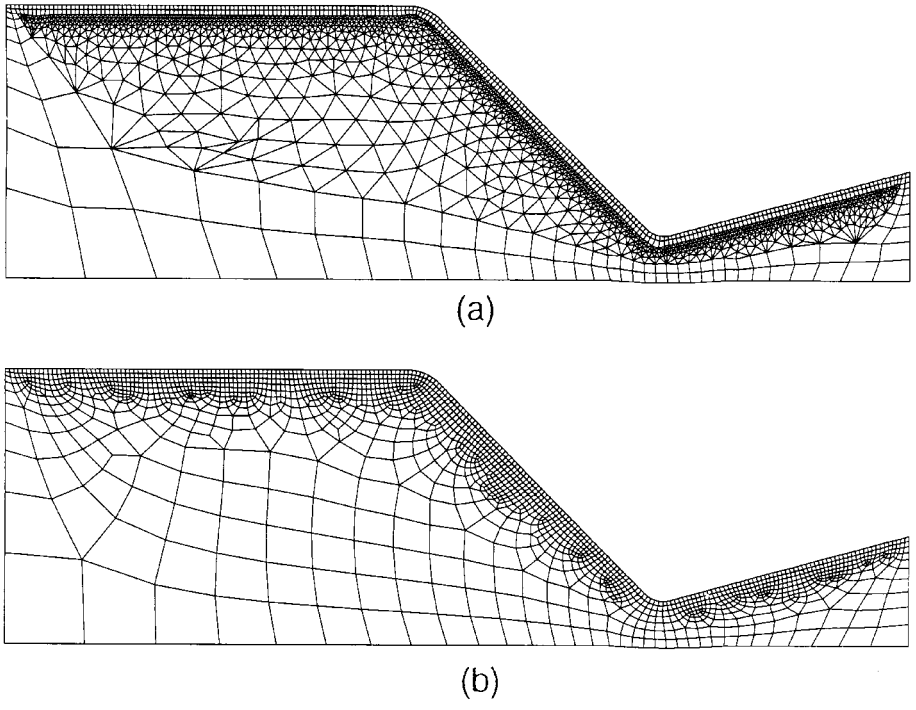

Figure 18. Large transition mesh for CFD application: (a) partially completed quad Mesh; (b) mesh after clean-up and smoothing

Figure 18(b) shows the same area after final clean-up and smoothing. In order to maintain a specified nodal density near the top of the area, a sizing function[26] was used during the triangle meshing process. The algorithm's ability to maintain the desired mesh density while still enforcing well-aligned rows of elements transitioning quickly to larger size elements is demonstrated in this example.

The final example in Figure 19 is an industrial application of the Q-Morph algorithm. For this example, the model consisting of 104 separate areas was first constructed using a commercial CAD software application. Surfaces are once again represented by rational B-splines. In practice, the Q-Morph algorithm is used as part of a set of meshing tools that also include mapping methods.[9] In this example, the narrow fillet regions are better represented with a mapped meshing technique, which can more appropriately create elements of high aspect ratio. Q-Morph is better suited to generating near-equilateral, isotropic quadrilaterals. Selection of the appropriate quad meshing method can be done automatically based on the number of lines comprising the area and its aspect ratio. After assigning line divisions, each area is first meshed with triangles and then transformed into quadrilaterals.

5. PERFORMANCE

Both speed and element quality of the resulting elements from Q-Morph was evaluated as part of this study. Table I shows performance results from two of the example problems above. For

the models in Figures 16 and 18, various element densities were specified and their results noted.

5.1. Speed

Table I shows CPU times for both the quad-conversion and the clean-up and smoothing portions of the Q-Morph algorithm. Tests were performed on a 195 MHz SGI UNIX workstation. For the toroidal surface in Figure 16, times are necessarily affected by the number of geometric evaluations required. Times range from 141 to 242 quads converted per CPU second. This is in contrast to the flat surface of Figure 18, where times ranged from 313 to 369 quads converted per CPU second. Clean-up and smoothing times were however slower for Figure 18 than for Figure 16 as the transition in element size defined by the quad conversion required additional iterations to converge. A wide variety of factors can affect the overall speed of the algorithm. Table I illustrates two cases where geometry and element transitions are critical.

5.2. Element quality

Element quality was measured by shape metric, β similar to that described by Lo and his coworkers[11,13] and Canann *et al.*[34] For this implementation, β is defined as the minimum

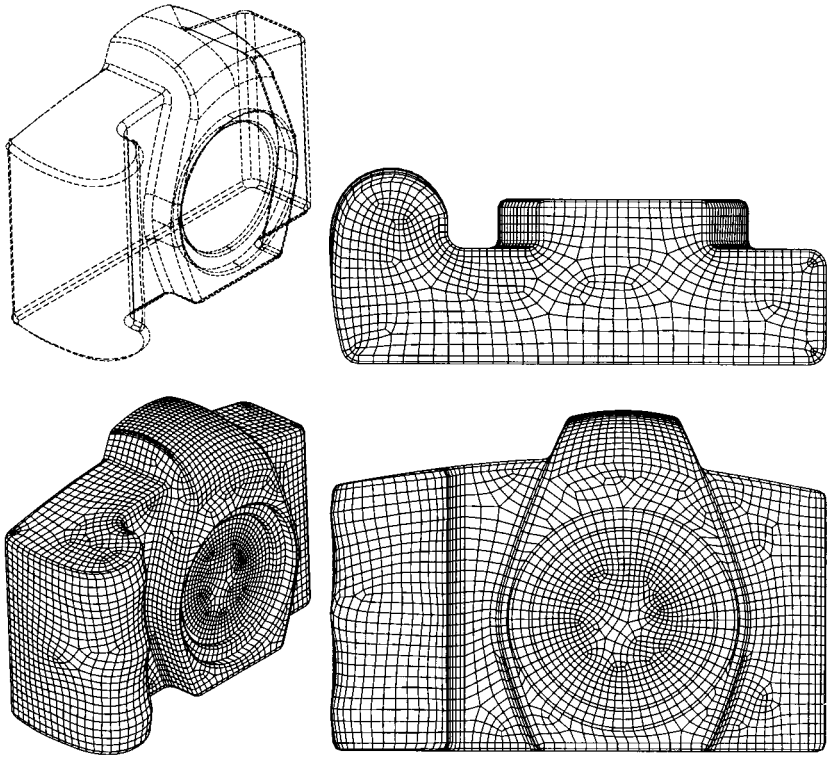

Figure 19. Industrial application of Q-Morph

Table I. Performance results from Q-Morph

	Triangle to Quad conversion					Clean-up and smoothing				
Model	Num. quads	Num. tris	Min. metric	Avg. metric	CPU time (s)	Num. quads	Num. tris	Min. metric	Avg. metric	CPU time (s)
Figure 15	351	0	0·371	0·893	1·45	350	0	0·515	0·905	0·44
	1208	0	0·391	0·905	5·31	1206	0	0·529	0·925	1·89
	4870	0	0·170	0·936	25·4	4845	0	0·376	0·948	10·9
	19 209	0	− 0·155	0·940	136	19 070	0	0·359	0·949	34·2
Figure 17	727	1	0·00	0·740	2·32	696	1	0·255	0·802	2·23
	1892	0	0·00	0·790	5·19	1785	0	0·344	0·859	4·36
	4472	1	0·00	0·811	12·1	4288	1	0·370	0·889	15·7
	10 581	0	− 0·155	0·817	33·4	10 231	0	0·382	0·889	31·4

triangle shape metric, α, defined by any of the four possible triangles formed by the vertices of the quadrilateral. A β value of 1·0 represents a perfect square, while a value of 0·0 represents a quadrilateral with a single corner angle of π. Concave or inverted quadrilaterals may be represented by negative values of β.

Both minimum and average metrics immediately following quad conversion and after clean-up and smoothing are shown in Table I. In some cases, inverted or poorly shaped quadrilaterals can be created during the quad conversion as indicated by the negative or zero metrics. Average metrics are however very high. In all cases tested, clean-up and smoothing improved the poorly shaped quads to well within usable limits. Table I also shows cases where a single triangle is created in the mesh. This occurs automatically in order to resolve situations where an odd number of boundary intervals are specified.

5.3. Robustness

A diversity of surfaces has been meshed using the Q-Morph algorithm and is currently part of a commercial FEA software release.[36] As such, it has been successfully ported to a wide variety of platforms, including Windows, NT and UNIX environments. In general, the Q-Morph algorithm is most beneficial on surfaces where the geometric feature sizes are larger than the specified element size. In addition, high quality quadrilaterals can be expected, provided the background triangle mesh captures the details of the surface and the background triangles are of reasonable quality (i.e. $\alpha > 0{\cdot}1$). In most cases where these conditions are not met, Q-Morph will be successful, however element quality may suffer.

6. CONCLUSION

The Q-Morph algorithm is an indirect quadrilateral meshing algorithm that utilizes an advancing front approach to transform triangles into quadrilaterals. It generates an all-quadrilateral mesh, provided the number of intervals on the boundary is even. The resulting mesh has few irregular internal nodes and produces elements whose contours, in general, follow the boundary

of the domain. Overall element quality is excellent. The Q-Morph algorithm borrows many of its techniques from the paving method,[2,3] but adapts them for use as an indirect method, operating on an existing set of triangles. In so doing, it is able to improve upon the paving technique by resolving some of its inherent difficulties. Relying on the topology of the initial triangle mesh to close opposing fronts eliminates the intersection problem, common to most direct methods of advancing front meshing. Improvements also include facility for handling individual element placement through the use of *states* for classifying front edges. Facility for handling transition in element sizes has also been addressed through the use of sizing information provided by the background triangle mesh and the definition of specific transformations that enable improved mesh transitions. Additionally, the background triangle mesh provides information that reduces the cost of direct evaluations on three-dimensional surface geometry.

REFERENCES

1. C. L. Lawson, 'Software for C^1 surface interpolation', *Mathematical Software III*, 161–194 (1977).
2. D. F. Watson, 'Computing the Delaunay tesselation with application to Voronoi polytope', *Comput. J.*, **24,** 167–172 (1981).
3. L. P. Chew, 'Guaranteed quality mesh generation for curved surfaces', *Proc. 9th Annual Comput. Geom.*, 1993, pp. 274–280.
4. J. C. Cavendish, 'Automatic triangulation of arbitrary planar domains for the finite element method', *Int. J. Numer. Meth. Engng.*, **8,** 679–696 (1974).
5. J. Peraire, J. Peiro, L. Formaggia, K. Morgan and O. C. Zienkiewicz, 'Finite element Euler computations in three dimensions', *Int. J. Numer. Meth. Engng.*, **26,** 679–696 (1988).
6. T. S. Lau and S. H. Lo, 'Finite element mesh generation over analytical surfaces', *Comput. Struct.*, **59,** 301–309 (1996).
7. R. Lohner, 'Extensions and improvements of the advancing front grid generation technique', *Commun. Numer. Meth. Engng.*, **12,** 683–702 (1996).
8. R. D. Cook, D. S. Malkus and M. E. Plesha, *Concepts and Applications of Finite Element Analysis*, 3rd ed., Wiley, New York, 1989.
9. W. A. Cook and W. R. Oakes, 'Mapping methods for generating three-dimensional meshes', *Comput. Mech. Engng.*, Aug., 67–72 (1982).
10. D. R. White, 'Automated hexahedral mesh generation by virtual decomposition', *Proc. 4th Int. Meshing Roundtable*, 1985, pp. 165–176.
11. S. H. Lo, 'Generating quadrilateral elements on plane and over curved surfaces', *Comput. Struct.*, **31,** 421–426 (1989).
12. B. P. Johnston, J. M. Sullivan Jr. and A. Kwasnik, 'Automatic conversion of triangular finite element meshes to quadrilateral elements', *Int. J. Numer. Meth. Engng.*, **31,** 67–84 (1991).
13. C. K. Lee and S. H. Lo, 'A new scheme for the generation of a graded quadrilateral mesh', *Comput. Struct.*, **52,** 847–857 (1994).
14. M. L. Staten and S. A. Canann, 'Post refinement element shape improvement for quadrilateral meshes', *Trends in Unstructured Mesh Generation*, *ASME*, AMD **220,** 9–16 (1997).
15. S. A. Canann, S. Muthukrishnan and B. Phillips, 'Topological improvement procedures for quadrilateral and triangular finite element meshes', *Proc. 3rd Int. Meshing Roundtable*, 1994, pp. 559–588.
16. P. Kinney, 'CleanUp: improving quadrilateral finite element meshes', *Proc. 6th Int. Meshing Roundtable*, 1995, 449–461.
17. P. L. Baehmann, S. L. Wittchen, M. S. Shephard, K. R. Grice and M. A. Yerry, 'Robust geometrically-based, automatic two-dimensional mesh generation', *Int. J. Numer. Meth. Engng.*, **24,** 1043–1078 (1987).
18. J. A. Talbert and A. R. Parkinson, 'Development of an automatic, two dimensional finite element mesh generator using quadrilateral elements and Bezier curve boundary definition', *Int. J. Numer. Meth. Engng.*, **29,** 1551–1567 (1991).
19. T. K. H. Tam and C. G. Armstrong, '2D finite element mesh generation by medial axis subdivision', *Adv. Engng. Software*, **13,** 313–324 (1991)
20. B. Joe, 'Quadrilateral mesh generation in polygonal regions', *Comput. Aid. Des.*, **27,** 209–222 (1995).
21. J. Z. Zhu, O. C. Zienkiewicz, E. Hinton and J. Wu, 'A new approach to the development of automatic quadrilateral mesh generation', *Int. J. Numer. Meth. Engng.*, **32,** 849–866 (1991).
22. S. H. Lo, 'A new mesh generation scheme for arbitrary planar domains', *Int. J. Numer. Meth. Engng.*, **21,** 1403–1426 (1985).
23. T. D. Blacker and M. B. Stephenson, 'Paving: A new approach to automated quadrilateral mesh generation', *Int. J. Numer. Meth. Engng.*, **32,** 811–847 (1991).

24. R. J. Cass, S. E. Benzley, R. J. Meyers and T. D. Blacker, 'Generalized 3-D Paving: An automated quadrilateral surface mesh generation algorithm', *Int. J. Numer. Meth. Engng.*, **39,** 1475–1489 (1996).
25. D. R. White and P. Kinney, 'Redesign of the Paving algorithm: Robustness enhancements through element by element meshing', *Proc. 6th Int. Meshing Roundtable*, 1997, pp. 323–335.
26. S. J. Owen and S. Saigal, 'Neighborhood-based element sizing control for finite element surface meshing', *Proc. 6th Int. Meshing Roundtable*, 1997, pp. 143–154.
27. J. R. Tristano, S. J. Owen and S. A. Canann, 'Advancing front surface mesh generation in parametric space using a Reimanian surface definition', *Proc. 7th Int. Meshing Roundtable*, 1998, pp. 429–445.
28. A. Cunha, S. A. Canann and S. Saigal, 'Automatic boundary sizing for 2D and 3D meshes', *Trends in Unstructured Mesh Generation*, *ASME*, AMD **220,** 65–72 (1997).
29. N. L. Jones, 'Solid modelling of earth masses for applications in geotechnical engineering', *Doctoral Dissertation*, University of Texas at Austin, 1990.
30. S. W. Sloan, 'A fast algorithm for generating constrained Delaunay triangulations', *Comput. Struct.*, **47,** 441–450 (1993).
31. P. L. George, F. Hecht and E. Saltel, 'Automatic mesh generator with specified boundary', *Comput. Meth. Appl. Mech. Engng.*, **92,** 269–288 (1991).
32. D. A. Field, 'Laplacian smoothing and Delaunay triangulations', *Commun. Appl. Numer. Meth.*, **4,** 709–712 (1988).
33. L. R. Hermann, 'Laplacian-isoparametric grid generation scheme', *J. Engng. Mech. Div. ASCE*, **102,** EM5, 749–756 (1976).
34. S. A. Canann, J. R. Tristano, and M. L. Staten, 'An approach to combined Laplacian and optimization-based smoothing for triangular, quadrilateral and tetrahedral meshes', *Proc. 7th Int. Meshing Roundtable*, 1998, 479–494.
35. S. A. Canann, Y. C. Liu, A. V. Mobley, 'Automatic 3D surface meshing to address today's industrial needs', *Finite Elements Anal. Des.*, **25,** 185–198 (1997).
36. ANSYS Version 5.5, ©Ansys Inc., Canonsburg, PA USA, 1998.

