JOURNAL OF COMPUTATIONAL PHYSICS **126,** 1–10 (1996)
ARTICLE NO. 0115

Regridding Surface Triangulations

RAINALD LÖHNER

GMU/CSI, The George Mason University, Fairfax, Virginia 22030-4444

Received August 11, 1995; revised December 26, 1995

An advancing front surface gridding technique that operates on discretely defined surfaces (i.e. triangulations) is presented. Different aspects that are required to make the procedure reliable for complex geometries are discussed. Notable among these are (a) the recovery of surface features and discrete surface patches from the discrete data, (b) filtering based on point and side-normals to remove undesirable data close to cusps and corners, (c) the proper choice of host faces for ridges, and (d) fast interpolation procedures suitable for complex geometries. Post-generation surface recovery or repositioning techniques are discussed. Several examples ranging from academic to industrial demonstrate the utility of the proposed procedure for *ab initio* surface meshing from discrete data, such as those encountered when the surface description is already given as discrete, the improvement of existing surface triangulations, as well as remeshing applications during runs exhibiting significant change of domain.

1. INTRODUCTION

The first and by far the most tedious step of any mesh generation procedure is the definition of the boundaries of the domain to be gridded. This may be accomplished in two ways: (a) analytically, i.e. via functions, or (b) using a tesselation or triangulation. From a practical point of view, it would seem that an analytic definition of the surface is the method of choice, given that nowadays most engineering data originates from some CAD-CAM package. However, in many instances, the boundaries of the domains to be gridded are not defined in terms of analytical functions, such as splines, B-splines, Coon's patches, or NURBS surfaces, but in terms of a triangulation, i.e., *discrete faces and points*. Several classes of applications where this is the case include:

—Visualization and manipulation of complex analytical functions, such as implicit analytic surfaces obtained via superposition or convolution [1, 2];

—Numerical simulations with geometric input data from measurements, such as

—Climate modeling, where the surface of the earth is available from remote sensing data;

—Groundwater and seepage modelling, where the geological layers have been obtained from drill data or seismic analysis; and

—Medical problems, where the patient data has been obtained from CAT scans;

—Numerical simulations that require remeshing, either

—Within the same field solver (e.g., forging simulations, where remeshing is required to regularize the grid, or simulations with adaptive remeshing); or

—For use with a different field solver (e.g., fluid–structure interaction problems, where the surface of the fluid domain is given by the structural surface grid [3], or hypersonic reentry problems, where ray-tracing based on the CFD mesh is used for heat transfer calculations).

Given this discrete data, one may either approximate it via analytical functions, or work directly with it. We prefer the second choice, as the proper approximation via analytical functions becomes cumbersome and problematic for complex geometries. A further reason for using directly discrete data is the fact that surface intersection and trimming are much easier on discrete data than on analytic surfaces. This allows the concurrent generation of surfaces by different users, that are then merged quickly to obtain the final configuration [4].

The present paper describes a surface meshing procedure for discrete data that employs the advancing front technique [5–12]. The technique is based on three steps: surface feature recovery, actual gridding, and surface recovery. The outline of the paper is as follows: having given the rationale for surface meshing *ab initio* from discrete data, Section 2 treats the problem of surface feature recovery. This step allows the surface gridding to obey sides, cusps, or other "ridge" features that may be present in the discrete data, and results in discrete surface patches. Section 3 describes the surface gridding of these discrete surface patches via the advancing front technique. Section 4 considers ways of making the procedure robust in the presence of sharp corners or convoluted patches. Section 5 considers the surface to surface interpolation problem, which is of fundamental importance for large surface grids, and Section 6 the postgeneration surface recovery. In Section 7 several examples that demonstrate the versatility and utility of the procedure are given. Finally, some conclusions are drawn and an outlook for future work is presented in Section 8.

0021-9991/96 $18.00

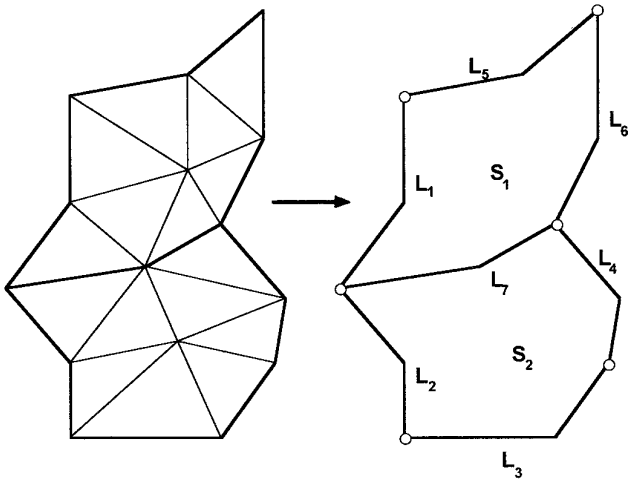

FIG. 1. Discrete surface patch recovery.

2. SURFACE FEATURE RECOVERY

A basic requirement for any surface gridder is that it obey sides, cusps, or other ''ridge'' features that may be present in the actual surface. In order to avoid gridding over these ''ridge'' features, sides are first generated along them, and then the surface is gridded with this initial front of sides. A simple way to determine ridges is by comparing the unit surface normals of adjacent faces. If the scalar product of them lies below a certain tolerance, a ridge is defined. *Corners* are defined as points that are attached to:

—Only one ridge;

—More than two ridges; or

—Two ridges with considerable deviation of unit side-vector.

Between corners, the ridges form *discrete lines*. These discrete lines either separate or are embedded completely (i.e., used twice) in *discrete surface patches*. The formation of discrete surface patches is performed with an advancing front algorithm. An arbitrary surface face is selected as a starting face and assigned a patch number. All neighbours that are not separated by a ridge are kept in a local list. The faces of this local list are interrogated for free neighbours in turn and are assigned the current patch number. This local list of neighbour faces becomes empty once all the contiguous faces not separated by a ridge have been marked. This procedure is repeated for all unmarked faces, yielding a list of patches. Using the information of which sides belong to a face, the discrete lines can be assigned to the patches in turn. Figure 1 sketches the recovery of surface features and the definition of discrete surface patches for a simple configuration.

3. ADVANCING FRONT TECHNIQUE

The advancing front technique has gained widespread acceptance for grid generation due to its versatility and speed [5–12]. The basic technique consists in marching into the as yet ungridded region by adding one face at a time. The border separating the gridded region from the as yet ungridded one is called the front. The algorithm may be summarized as follows:

F1. Define the surfaces to be gridded. In the present case, this is done via triangulations. At the same time, define the boundaries of the surfaces.

F2. Define the spatial variation of element size, stretchings, and stretching directions for the elements to be created.

F3. Using the information given for the distribution of element size and shape in space, as well as the line-definitions: generate sides along the lines that connect surface patches. These sides form an initial front for the triangulation of the surface patches.

F4. Select the next side to be deleted from the front; in order to avoid large faces crossing over regions of small faces, the side forming the smallest new face is selected as the next side to be deleted from the front.

F5. Determine the discrete surface face IFADS that contains or is close to the midpoint of the side to be deleted.

F6. Obtain the unit surface normal $\mathbf{n}_{\text{fds}}$ for IFADS.

F7. With the information of the desired element size and shape, and $\mathbf{n}_{\text{fds}}$: Select a ''best point'' position for the introduction of a new point IPNEW (see Fig. 2).

F8. Determine whether a point exists in the already generated grid that should be used in lieu of the new point. If there is such a point, set this point to IPNEW and continue searching.

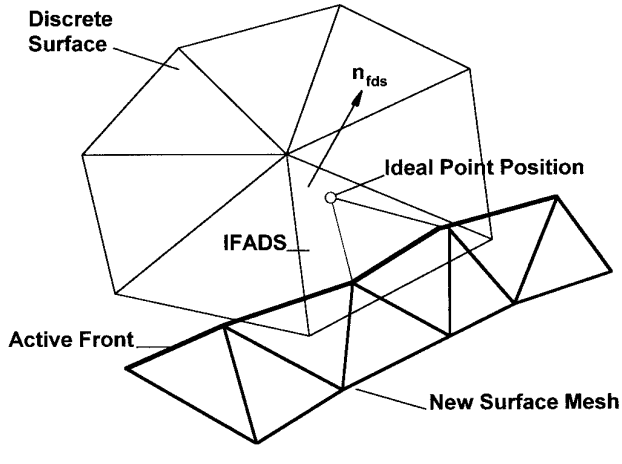

FIG. 2. Generation of surface triangulation on discrete surface.

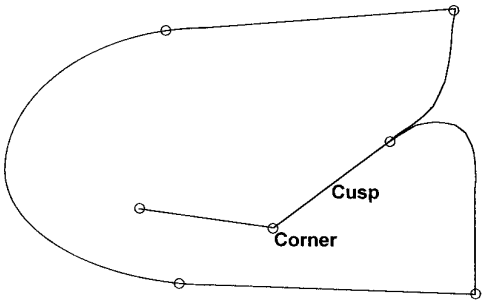

FIG. 3. Discrete surface patch with cusps and corners.

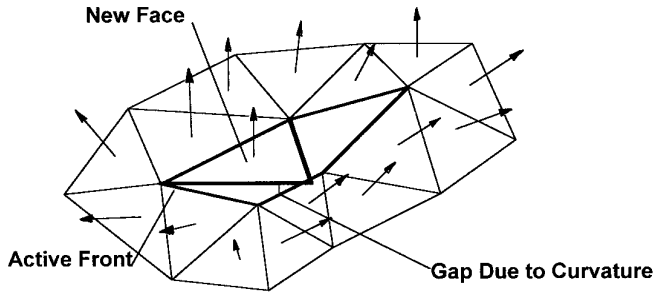

FIG. 4. Non-uniqueness of front crossing for curved surfaces.

F9. Determine whether the face formed with the selected point `IPNEW` does not cross any given sides. If it does, select a new point as `IPNEW` and try again.

F10. Add the new face, point, and sides to their respective lists.

F11. If a new point was added: reposition it on the discretely defined surface.

F12. Find the desired element size and stretching for the new sides.

F13. Delete the known sides from the list of sides.

F14. If there are any sides left in the front, go to F4.

As compared to the surface gridding of analytically defined surfaces, which has been treated by a number of authors [7–9, 11, 12], the only differences are:

—The search for the discrete surface face containing or close to a point (Steps F5, F11);

—The introduction of a normal vector $\mathbf{n}_{\text{fds}}$ for each face in order to determine the ideal point position (Step F7);

—The repositioning of new points on the discretely defined surface (F11).

4. ENHANCEMENTS FOR ROBUSTNESS

The procedure, as described above, will work well for smooth surfaces. In practice, however, one is often faced with discrete surfaces that exhibit cusps, sharp corners, or ridges with high curvature (see Fig. 3). In these instances, the procedure must be enhanced in order to work reliably. The most important of these enhancements: 2D crossing check, point and side normals, angle of visibility for filtering inappropriate data and proper host face for ridges, are described in the sequel.

4.1. *2D crossing check.* In regions of colliding fronts on 3D surfaces with curvature, the face/side-crossing check is not uniquely defined (see Fig. 4). This ambiguity is best circumvented by transforming all the points required, i.e., those in the list of close points and those attached to close sides, to the plane defined by the midpoint of the side to be deleted and the normal vector $\mathbf{n}_{\text{fds}}$. Thereafter, the face/side-crossing check is performed in 2D.

4.2. *Point and side normals.* It is advisable to use the point and side normals obtained by interrogating the host face of the discrete data in order to filter out undesired data from the list of close points and sides. In this way, the front data of the lower portion of the cusp shown in Figure 3 is automatically removed when generating a face on the upper portion and vice versa.

4.3. *Angle of visibility.* In order to avoid the improper choice of close wrong points that may have the correct point and side normals, but belong to another portion of the discrete surface patch (see Fig. 5), all points outside the allowable ''angle of visibility'' α are no longer considered. A meaningful value for α can be obtained by measuring the local surface curvature of the underlying discrete surface in the vicinity of the side to be removed from the front. In the present case, this is done by simply comparing the normals of the host face and its neighbours.

4.4. *Proper host face for ridges.* For the sides along ridges, there can be instances where the host face is not properly defined. As an example, consider the situation

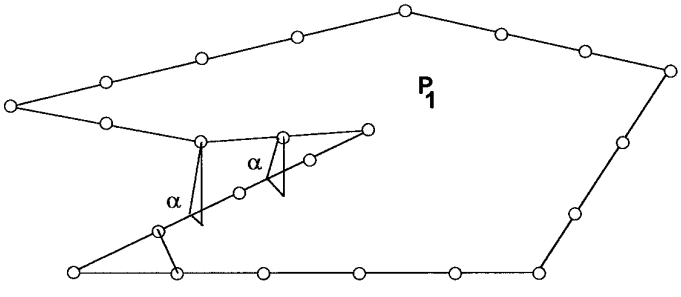

FIG. 5. Filtering with angle of visibility.

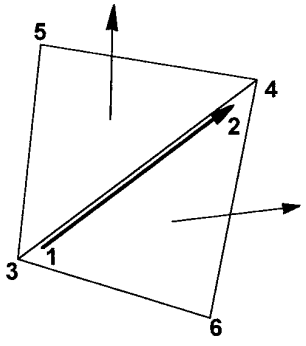

FIG. 6. Selection of proper host faces at ridges.

shown in Fig. 6. Given the orientation of the side 1-2, the proper host face is 3-4-5. However, face 3-6-4 could also be considered as a host face. In order to resolve this ambiguity, the point $\mathbf{x}_{\text{fds}}$ that is furthest from the side is determined for each possible host face. The proper host face has to satisfy

$$c_s = [(\mathbf{x}_2 - \mathbf{x}_1) \times (\mathbf{x}_{\text{fds}} - \mathbf{x}_1)] \cdot \mathbf{n}_{\text{fds}} > 0. \tag{1}$$

5. SURFACE TO SURFACE INTERPOLATION

One of the main differences between gridding discrete, as opposed to analytic, surfaces is the potentially very expensive search for the host face of each new point and side generated. Careless implementation of these operations would lead to an $O(N_p \cdot N_{dp})$ complexity, where N_p denotes the number of new surface points created and N_{dp} the number of points defining the discrete surface patch. If both of these are of similar magnitude, the result is a complexity of $O(N_p^2)$, clearly inappropriate for large surface grids. Given that the advancing front algorithm by its very nature adds points and sides in the vicinity of known data points, the host face of the side to be taken out can be used as a good starting guess from which to find the correct host face via a neighbour-to-neighbour search (see

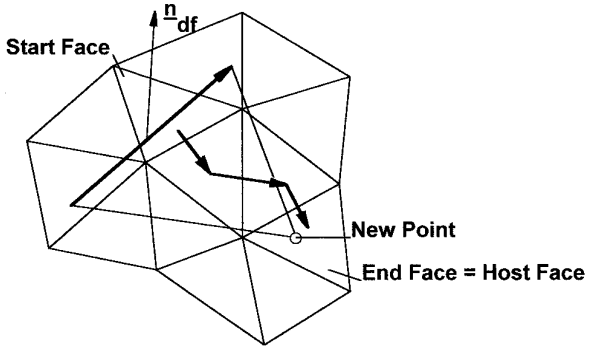

FIG. 7. Neighbour-to-neighbour jump search procedure.

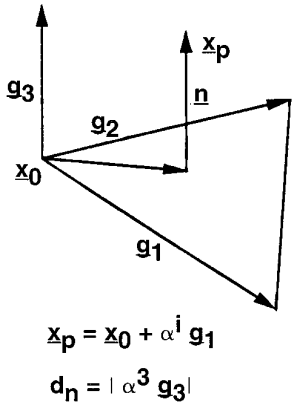

FIG. 8. Surface interpolation.

Fig. 7). With the notation of Fig. 8, the point to be interpolated $\mathbf{x}_p$ is given by

$$\mathbf{x}_p = \mathbf{x}_0 + \sum_{i=1}^{3} \alpha^i \mathbf{g}_i, \tag{2}$$

where

$$\mathbf{g}_i = \mathbf{x}_i - \mathbf{x}_0, \quad i = 1, 2; \quad \mathbf{g}_3 = \frac{\mathbf{g}_1 \times \mathbf{g}_2}{|\mathbf{g}_1 \times \mathbf{g}_2|}, \quad \text{(3a), (3b)}$$

and the shape-functions or barycentric coordinates N^i are given by

$$N^i = \alpha^i, \quad i = 1, 2; \quad N^0 = \alpha^0 = 1 - \alpha^1 - \alpha^2, \quad \text{(3c), (3d)}$$

the point $\mathbf{x}_p$ is considered as being on the surface face **Iff**:

$$\min(N^i, 1 - N^i) \geq 0, \quad \forall i = 0, 1, 2, \tag{4a}$$

and

$$d_n = |\alpha^3 \mathbf{g}_3| \leq \delta_n. \tag{4b}$$

Here δ_n denotes a tolerance for the relative distance normal to the surface face. Many search and interpolation algorithms have been devised over the years. We have found that for generality, a layered approach of different interpolation techniques works best. Wherever possible, a vectorized advancing front neighbour-to-neighbour algorithm is employed as the basic procedure [13]. Given that the advancing front algorithm by its very nature adds points and sides in the vicinity of known data points, the host face of the side to be taken out can be used as a good starting guess from which to find the correct host face via this neighbour-to-neighbour search (see Fig. 7). Should this fail, octrees [14, 15] are employed. Finally, if this ap-

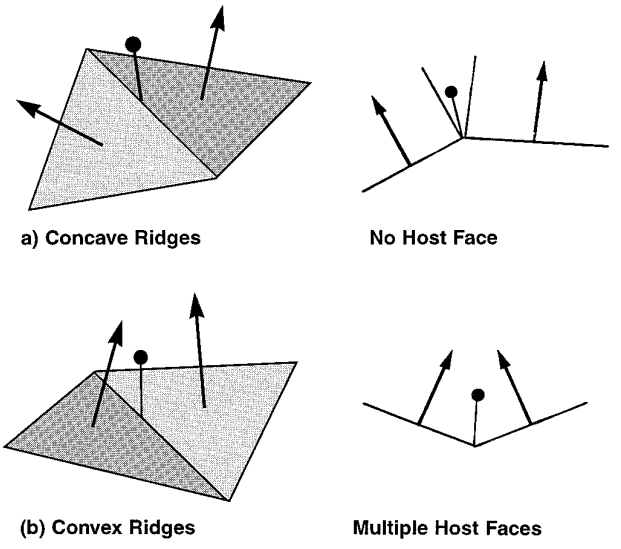

FIG. 9. Problems when searching for host faces.

proach fails too, a brute force search over all the surface faces is performed [13]. For realistic 3D surface geometries, the interpolation of surface grid information may be complicated by a number of the factors. The first of these factors is the proper choice of δ_n, i.e., the proper answer to the question: ''How close must a face be to a point in order to be acceptable ?'' This is not a trivial question for situations where narrow gaps exist in the discrete surface mesh, when there is a large discrepancy of face-sizes between the discrete surface grids and the new surface grid, as well as when the discrete surface grid exhibits highly stretched elements. Our experience indicates that the choice

$$\delta_n < c_n \cdot |\mathbf{g}_1 \times \mathbf{g}_2|^{0.5}, \quad c_n = 0.05, \tag{5}$$

works reliably, although the constant c_n may be problem dependent. A second complication often encountered arises due to the fact that Eq. (4a) may never be satisfied (e.g., the convex ridge shown in Fig. 9a), or it may be satisfied by more than one surface face (e.g., the concave ridge shown in Fig. 9b). In the first instance the criterion given by Eq. (4a) may be relaxed somewhat to

$$\min(N^i, 1 - N^i) \geq \varepsilon, \quad \forall i = 0, 1, 2, \tag{6}$$

where ε is a small number. For the second case, the discrete surface face with the smallest normal distance d_n is selected. We remark that in both of these instances the final point location is unaffected by the final host surface face, as the interpolation weights are such that only the points belonging to the ridge are used for interpolation. We have found that it is very important to take the face that has the smallest distance to the point being interpolated in order to mitigate any possible problems. For situations close to corners, gaps, or multi-surface configurations, an exhaustive search over all faces will be triggered. In order not to check in depth the complete surface mesh, only the faces that satisfy the relaxed closeness criteria $\varepsilon \geq -1$, $c_n \leq 0.5$ are considered. The face with the closest distance to the point is kept. If a face satisfies Eq. (4a), the closest distance is indeed d_n. Should this not be the case, the closest distance to the three sides ij of the face is taken:

$$\delta = \min_{ij} |\mathbf{x}_p - (1 - \beta_{ij})\mathbf{x}_i - \beta_{ij}\mathbf{x}_j|, \qquad \beta_{ij} = \frac{(\mathbf{x}_p - \mathbf{x}_i) \cdot (\mathbf{x}_j - \mathbf{x}_i)}{(\mathbf{x}_j - \mathbf{x}_i) \cdot (\mathbf{x}_j - \mathbf{x}_i)}. \tag{7a), (7b}$$

Should two faces have the same normal distance, the one with the largest minimum shape-function α^i, $i = 0, 1, 2$ is retained.

A third complication arises for cases where cusps or close surfaces are present. For these cases, the ''best'' face may actually lie on the opposite side of the face being interpolated. This ambiguity is avoided by defining a surface normal, and then only considering the faces and points whose normals are aligned, i.e., those for which

$$\mathbf{n}_{\text{fds}} \cdot \mathbf{n}_p > c_s, \quad c_s = 0.5. \tag{8}$$

Here $\mathbf{n}_{\text{fds}}, \mathbf{n}_p$ denote the discrete surface face and the point/side-normals respectively. Experience indicates that it is advisable to perform a local exhaustive search for all faces surrounding the best host face found in order to obtain the host face that satisfies Eqs. (4), (7) as best possible.

Although these extra steps for interpolation seem complex, they are not only indispensable for discrete data that exhibits cusps, high surface curvature, and internal ridges, but their cost is not significant.

6. POSTGENERATION SURFACE RECOVERY

After the surface grid has been generated, it may be desirable to reposition the points in order to meet certain surface fidelity criteria. Obvious choices, shown in Fig. 10, are:

(a) *Keep as is.* I.e., no postprocessing. This will be the preferred choice if the loss of surface fidelity due to curvature and/or different face-sizes is small.

(b) *Move to closest discrete point.* The rationale for this option is that if the underlying discrete data is of much finer resolution than the newly generated mesh, moving each point to a given point will not distort the mesh significantly while assuring an exact pointwise representation.

(c) *Higher order recovery.* In this case, the new sur-

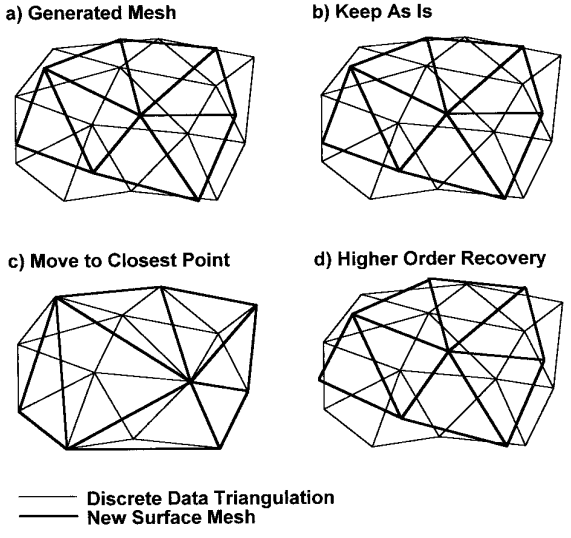

FIG. 10. Postgeneration surface recovery.

face points are repositioned using higher order recovery procedures for the discrete data. This option is attractive if the discrete surface data is much coarser than the newly generated mesh and exhibits surface curvature. The first step consists in computing average normals at the points of the discrete surface. In a second step, the information of which host face a point belongs to, and which are its local area coordinates ζ_1, ζ_2, ζ_3, together with the point normals, is used to reposition the point. We have considered to date:

(c1) *Quadratic recovery.* For each side of the discrete surface triangulation, a mid-point location is estimated from a Hermitian polynomial as

$$\begin{aligned}\mathbf{x} = (1-\xi)^2(1+2\xi)\mathbf{x}_1 + \xi(1-\xi)^2\mathbf{r}_1 \\ + \xi^2(3-2\xi)\mathbf{x}_2 - \xi^2(1-\xi)\mathbf{r}_2,\end{aligned} \tag{9}$$

where

$$\mathbf{r} = s\frac{\mathbf{n}\times(\mathbf{s}\times\mathbf{n})}{|\mathbf{n}\times(\mathbf{s}\times\mathbf{n})|}, \quad \mathbf{s} = \mathbf{x}_2 - \mathbf{x}_1, \quad s = |\mathbf{s}|, \tag{10}$$

and $\xi = 0.5$. With this information, and using the notation in Fig. 11, the recovered point location is given by the standard quadratic triangle shape functions [16]:

$$\begin{aligned}\mathbf{x} = \zeta_1(2\zeta_1-1)\mathbf{x}_1 + \zeta_2(2\zeta_2-1)\mathbf{x}_2 + \zeta_3(2\zeta_3-1)\mathbf{x}_3 \\ + 4\zeta_1\zeta_2\mathbf{x}_4 + 4\zeta_2\zeta_3\mathbf{x}_5 + 4\zeta_3\zeta_1\mathbf{x}_6.\end{aligned} \tag{11}$$

(c2) *Cubic recovery.* For each face of the discrete surface triangulation, we have nine pieces of information: location of the end-points, and inclination of the normals with respect to the plane formed by the plane. For a complete cubic, 10 pieces of information are required. We use the Zienkiewicz triangle, derived for plate elements [16], to account for this deficit.

The recovery procedures described represent just two instances of many possible alternatives, such as mid-normals, local spline, Clugh–Tocher, Doo–Sabin, etc. [17, 18].

7. EXAMPLES

The described procedure was applied to a number of cases in order to test its applicability in production environments. For all of these cases, the surface was recovered using quadratic and cubic functions. However, no graphically discernable difference was encountered.

6.1. *Sphere.* We start with this academic example to show the basic possibilities of surface gridding based on discrete data representations. An initial surface mesh, shown in Fig. 12a, is taken as the starting point. This mesh contains no faces whose normals vary by more that $\beta = 10°$ among neighbours. For this reason, the whole surface is treated as one patch, with the largest side taken as a discrete line. Two new surface grids, one of constant element size and one with a prescribed source at one end of the sphere, were generated and are shown in Fig. 12b, c.

6.2. *Forging piece.* This second problem demonstrates the use of surface remeshing for discretely defined domains within the same numerical simulation. A piece that originally started out as pie-shaped has been distorted significantly due to forging. A complete remeshing of the computational domain is required. The surface of the mesh at this stage is shown in Fig. 13a. The edge-detection algo-

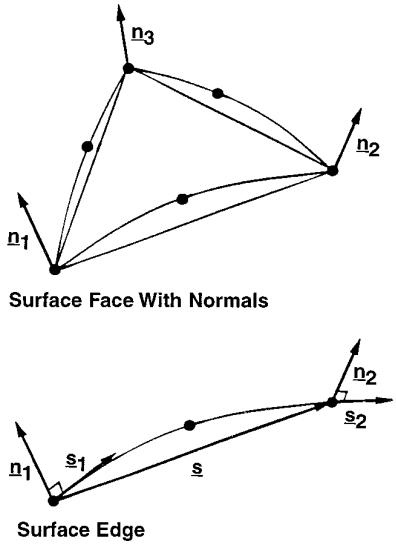

FIG. 11. Quadratic surface reconstruction.

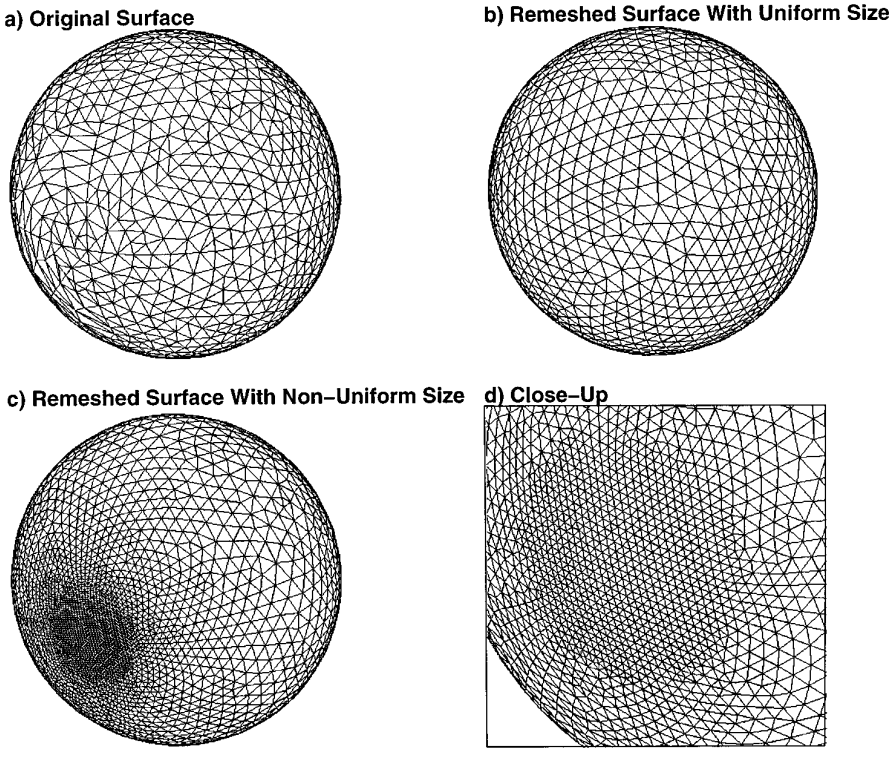

FIG. 12. Sphere.

rithm then forms the discrete line and surface patch definition shown in Fig. 13b. The angle used to determine ridges was set to $\beta = 25°$. An adaptive background grid is generated automatically [10], starting from a cube. With this background mesh, a new surface triangulation, shown in Fig. 13c, is generated.

6.3. *Ship.* This case illustrates the use of surface meshing from discrete data as a means to expedite the domain definition process, as well as the possibility of correcting an initially improper surface discretization. An initial surface mesh for the ship, shown in Fig. 14a, was provided as a starting point. The discrete lines and surface patches obtained using an angle tolerance of $\beta = 30°$ between adjacent faces are shown in Fig. 14b. A finer meshing region close to the water line was specified by using two surface sources [10]. The final surface mesh, given in Fig. 14c, not only exhibits a better discretization (i.e., less small angles), but it is also better suited for the numerical simulation.

6.4. *Car fender die.* This case shows the use of surface meshing from discrete data as a means of streamlining data input within industrial simulations. The original CAD dataset had over 500 surface patches, many of them overlapping and in need of trimming. Instead, a cloud of points, obtained from a digitization of the actual part, is taken as the starting point. This cloud of approximately 5,000 points, shown in Fig. 15a, is then triangulated using an automatic surface recovery tool developed by the author [19] (for automatic surface recovery, see also [18, 20]). The surface is now defined discretely, and lines and patches, shown in Fig. 15c, are recovered. The final surface mesh, suitable for stamping calculations, is shown in Fig. 15d. This example clearly demonstates the possible advantages of discrete surface gridding. Trimming and combining over 500 surfaces is a tedious and time-consuming effort, which can be reduced drastically as shown here.

6.5. *Generic hypersonic airplane geometry.* This final case shows the combination of discrete and analytically defined surfaces to obtain rapid turnaround in preliminary design calculations. The airplane fuselage is given from a structural dynamics calculation and shown in Fig. 16a. The recovered discrete surface patches, together with the added outer box and some further analytical patches for nozzle entry and exit planes, is shown in Fig. 16b. The new surface discretization, suitable for preliminary aerodynamic design calculations, is shown in Fig. 16c.

All of the surface grids shown were obtained in less than 5 min on an IBM RISC-550 workstation, indicating that it is feasible to port these automatic surface meshing and remeshing techniques into production codes.

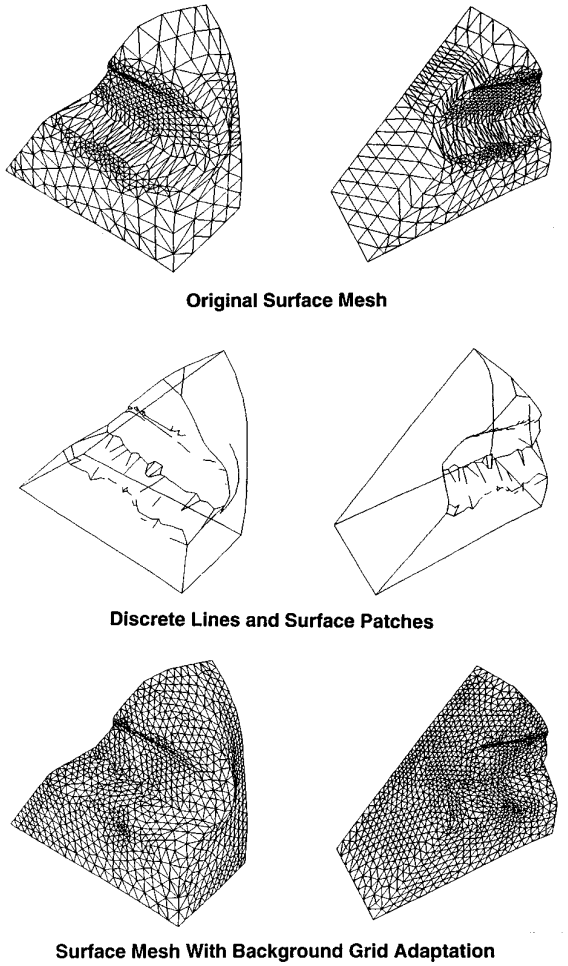

FIG. 13. Forging piece.

8. CONCLUSIONS AND OUTLOOK

An advancing front surface gridding technique that operates on discretely defined surfaces has been presented. Different aspects that are required to make the procedure reliable for complex geometries are discussed. These include:

—Recovery of surface features and discrete surface patches;

—Filtering based on point and side-normals to remove undesirable data close to cusps and corners;

—Filtering based on an angle of visibility to remove irrelevant close-point/side data;

—The proper choice of host faces for ridges; and

—Fast interpolation procedures suitable for complex geometries.

The task of postgeneration surface recovery or repositioning is also discussed, and some of the many possible alternatives are given.

Several examples ranging from academic to industrial demonstrate the utility of the developed procedure for *ab initio* surface meshing from discrete data, such as those encountered when the surface description is already given as discrete, the improvement of existing surface triangulations, as well as remeshing applications during runs exhibiting significant change of domain.

As with any other technique, improvements are always possible. They will center on better postgeneration surface recovery schemes and further enhancements in robustness and reliability for complex geometries.

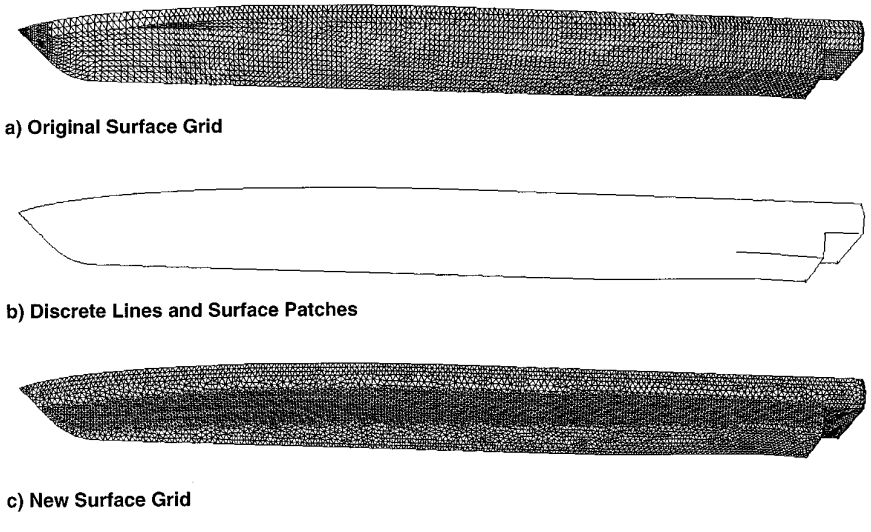

FIG. 14. Ship hull.

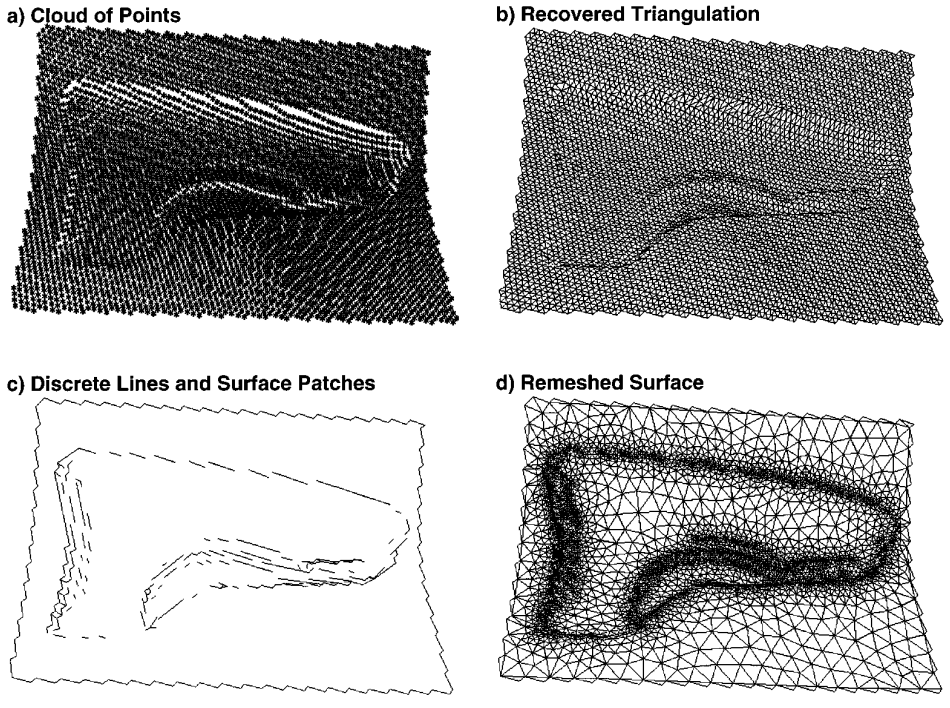

FIG. 15. Car fender die.

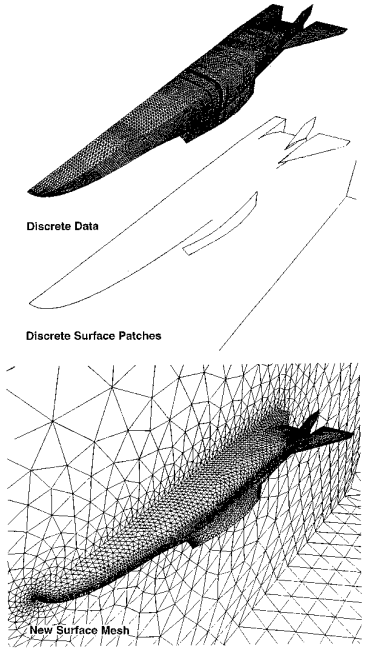

FIG. 16. Generic hypersonic airplane geometry.

ACKNOWLEDGMENTS

A considerable portion of this work was carried out while the author was visiting the Centro Internacional de Métodos Numericos en Ingenieria (CIMNE) at the Universidad Politécnica de Catalunya, Barcelona, Spain. The support for this visit is gratefully acknowledged.

REFERENCES

1. J. Bloomenthal, *Comput. Aided Geom. Design*, November (1988).
2. J. Bloomenthal and K. Ferguson, *Proc. SIGGRAPH*, August (1995).
3. R. Löhner, C. Yang, J. Cebral, J. D. Baum, H. Luo, D. Pelessone, and C. Charman, AIAA-95-2259, (1995) (unpublished).
4. S. H. Lo, *Int. J. Numer. Methods Eng.* **38,** 943 (1995).
5. J. Peraire, M. Vahdati, K. Morgan, and O. C. Zienkiewicz, *J. Comput. Phys.* **72,** 449 (1987).
6. R. Löhner, *Commun. Appl. Numer. Methods* **4,** 123 (1988).
7. R. Löhner and P. Parikh, *Int. J. Numer. Methods Fluids* **8,** 1135 (1988).
8. J. Peiro, J. Peraire, and K. Morgan, *Proceeding. POLYMODEL XII Conf., Newcastle-upon-Tyne, May 23–24* (1989) (unpublished).
9. J. Peraire, K. Morgan, and J. Peiro, AGARD-CP-464, 18, (1990) (unpublished).
10. R. Löhner, *Commun. Appl. Numer. Methods,* submitted.
11. K. Nakahashi and D. Sharov, AIAA-95-1686-CP, 1995 (unpublished).
12. C.-J. Woan, AIAA-95-2202, 1995 (unpublished).
13. R. Löhner, *J. Comput. Phys.* **118,** 380 (1995).
14. D. N. Knuth, *The Art of Computer Programming*, Vol. 3 (Addison–

Wesley, Reading, MA, 1973).
15. R. Sedgewick, *Algorithms* (Addison–Wesley, Reading, MA, 1983).
16. O. C. Zienkiewicz, *The Finite Element Method,* (McGraw–Hill, New York, 1982).
17. G. Farin, *Comput. Aided Geom. Design* **3**(2), 83 (1986).
18. J. Hoschek and D. Lasser, *Fundamentals of Computer Aided Geometric Design* (Peters, 1993).
19. R. Löhner, Surface reconstruction from clouds of points, preprint.
20. H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, *Comput. Graph.* **26**(2), 71 (1992).