INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng 2002; **55**:177–190 (DOI: 10.1002/nme.499)

A feature-preserving volumetric technique to merge surface triangulations

Juan R. Cebral, Fernando E. Camelli and Rainald Löhner[*,†]

Department of Civil Engineering, School of Computational Sciences, GMU/CSI, MS 5C3, George Mason University, Fairfax, VA 22030-4444, U.S.A.

SUMMARY

Several extensions and improvements to surface merging procedures based on the extraction of iso-surfaces from a distance map defined on an adaptive background grid are presented. The main objective is to extend the application of these algorithms to surfaces with sharp edges and corners. In order to deal with objects of different length scales, the initial background grids are created using a Delaunay triangulation method and local voxelizations. A point enrichment technique that introduces points into the background grid along detected surface features such as ridges is used to ensure that these features are preserved in the final merged surface. The surface merging methodology is extended to include other Boolean operations between surface triangulations. The iso-surface extraction algorithms are modified to obtain the correct iso-surface for multi-component objects. The procedures are demonstrated with various examples, ranging from simple geometrical entities to complex engineering applications. The present algorithms allow realistic modelling of a large number of complex engineering geometries using overlapping components defined discretely, i.e. via surface triangulations. This capability is very useful for grid generation starting from data originated in measurements or images. Copyright © 2002 John Wiley & Sons, Ltd.

KEY WORDS: surface intersection; geometric modelling; mesh generation; unstructured grids; discrete data

INTRODUCTION

The generation of three-dimensional grids for finite element analysis requires a surface representation of the computational domain. Typically, these surfaces are represented using analytical surface patches, such as Coon's patches, B-Splines or NURBS. However, in an increasingly large number of engineering applications, the geometrical information is given as discrete data, i.e. surface triangulations or clouds of points. This data usually originates from measurements. Examples include blood flow calculations in anatomical models reconstructed from medical images, atmospheric dispersion models with terrain geometry constructed from satellite data, seepage and secondary oil recovery computations in reservoir models derived from seismic

[*]Correspondence to: Rainald Löhner, GMU/CSI, MS 5C3, Department of Civil Engineering, George Mason University, Fairfax, VA 22030-4444, U.S.A.
[†]E-mail: rlohner@gmu.edu

Received 18 April 2001
Revised 30 June 2001

measurements, visualization, etc. At the same time, in other problems, the number of surface patches is so large that a surface definition based on discrete data may be preferable. Examples of this type include simulations of under hood cooling for automobiles, flow around complex molecules, etc.

The use of discrete data to model geometries as sets of intersecting components requires algorithms to perform Boolean operations between surface triangulations. Traditionally, surface mesh generation over intersecting triangulations has been carried out by computing the exact intersection between the triangulations [1, 2]. Experience with this approach indicates that the resulting triangulations tend to have very distorted elements (aspect ratios of $1:10^6$ are not uncommon) and cannot account for narrow gaps. Alternatively, volumetric techniques based on voxelization and iso-surface extraction have been used for visualization [3], simplification and repair of polygonal models [3], volume graphics [4] and finite element modelling [5].

Volumetric techniques reduce the calculation of surface intersections to the extraction of an iso-surface, yield good quality elements and avoid the problems with narrow gaps. Although this methodology works particularly well for smooth surfaces, it does not preserve features such as ridges, which may be present in the initial triangulations. A ridge is defined as an edge shared by two triangles with significant deviation of their unit normals, e.g. larger than 10°. The objective of this paper is to improve and extend the volumetric techniques described by Cebral *et al.* [5] to cases where geometric features must be preserved in the final triangulations and to implement other Boolean operations besides surface merging (union operation). These extensions and improvements will increase the range of applicability of discrete data to component-based geometric modelling of engineering shapes with sharp edges and corners.

SURFACE MERGING

The merging of surface triangulations, as explained by Cebral *et al.* [5], is carried out by creating a background grid of tetrahedra that covers the whole computational domain. The shortest distance from a background point to any object surface, i.e. a distance map, is then computed. A positive or negative distance indicates whether the point lies outside or inside of an object, respectively. This sign is obtained by checking the surface normal at the closest location to the point. The triangulation over the intersecting objects is then obtained by extracting the iso-surface of zero distance using a classic algorithm [6]. Mesh optimization procedures [7, 8] are applied as pre- and post-processing operators in order to improve the element quality of the initial and final triangulations. The merging procedure is illustrated schematically in Figure 1.

The distance map is computed as follows. A loop is performed over the triangles of each object surface. For each of these triangles, the list of background points enclosed in the bounding box of the triangle is found using octrees [9]. For each of these close points, the signed distance to the triangle is computed and stored if smaller in absolute magnitude than the previous value. This algorithm yields the distance map for a given object. Depending on how distance maps of different objects are combined, different Boolean operations can be implemented. Denoting by d' the distance map for a given object and by d the final distance map, the following pseudo-code summarizes the algorithm to implement different

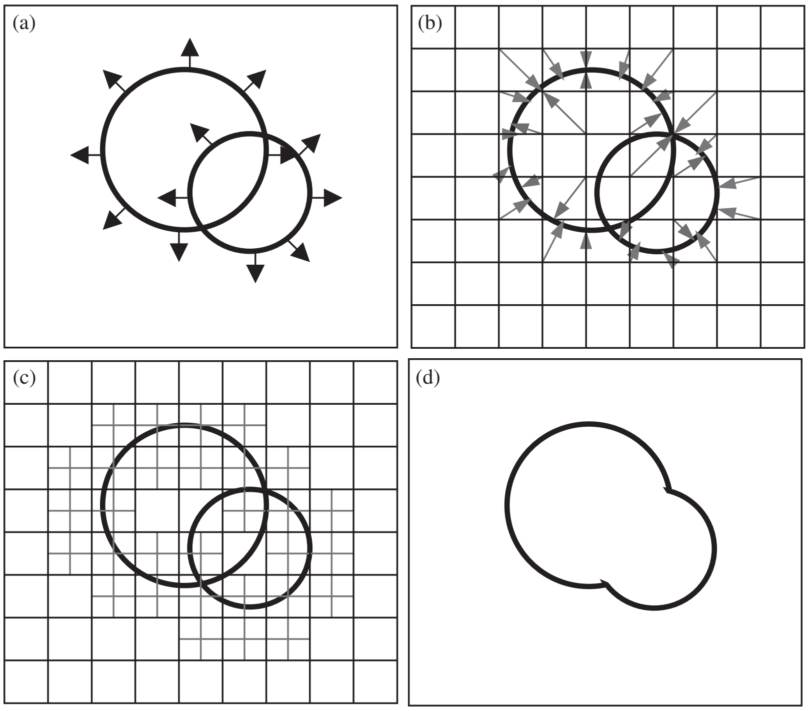

Figure 1. Surface merging procedure: (a) object triangulations and surface normals; (b) uniform voxelization and distance map; (c) adaptive refinement of the background grid; and (d) iso-surface extraction.

Boolean operations:

```
Initialize $d(ip) =$ large value for all points $ip$
FOR each object
    Get $d'(ip) =$ shortest distance to object for all points $ip$
    FOR each point $ip$
        $d(ip) = \min(d(ip), d'(ip))$                              ! union operation (merging)
        IF$(d(ip) > 0)$ $d(ip) = \min(d(ip), d'(ip))$              ! subtraction operation
        IF$(d(ip)^* d'(ip) < 0)$ $d(ip) = \max(d(ip), d'(ip))$     ! intersection operation
        ELSE                     $d(ip) = \min(d(ip), d'(ip))$     ! intersection operation
    END
END
```

Since the procedure used to compute the object distance map (d') operates only on the points enclosed in the bounding box of each surface triangle [5], it may happen that points far from the object surfaces are never considered, i.e. they will contain the large positive value used for initialization of the distance map. If these points are in the interior of an object, other points with negative distances will surround them; therefore, a second iso-surface will be

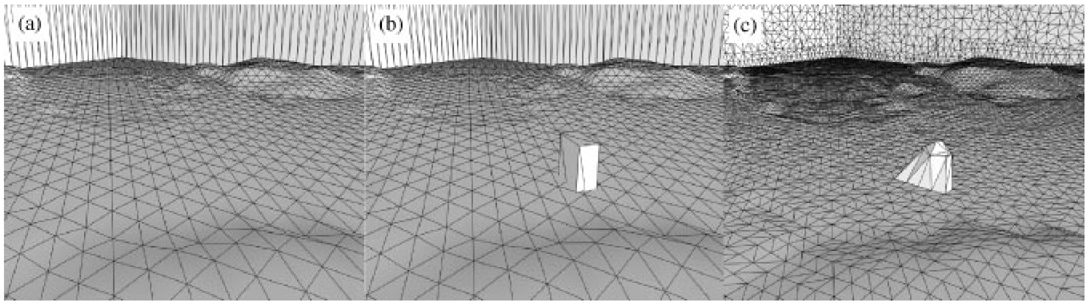

Figure 2. Surface merging without feature preservation: (a) terrain surface triangulation; (b) superposed terrain and building triangulations; and (c) merged surface.

generated in the interior of such an object. These undesired surfaces can be avoided by forcing the calculation of the distance map to all the points in the background grid. Alternatively, disconnected surface components can be automatically identified at the end of the calculation and the undesired surfaces interactively deleted.

The algorithm described above does not guarantee that features present in the initial surface triangulations will be preserved in the final merged surface. This is a property common to all surface-processing operators based on volumetric techniques. Consider, for example, the case of merging terrain data (Figure 2(a)) with a surface representing a building (Figure 2(b)). A straightforward application of the surface-merging procedures results in an iso-surface that does not preserve the sides of the building (Figure 2(c) and 2(d)). Increasing the resolution of the background grid via adaptive refinement does not solve this problem, it yields a better approximation to the shape of the building but its sides are never exactly recovered. This smoothing property of the surface-merging algorithm may be desirable in some cases such as the merging of arterial branches, which are represented by smooth surface triangulations. However, in other applications such as the building and terrain data, the results are not acceptable. In what follows we describe several enhancements added to the original algorithm in order to obtain merged surfaces that preserve desired surface features.

ADAPTIVE VOXELIZATION

Voxelization and adaptation

In the original surface-merging algorithm an initial Cartesian background grid with cubic elements (voxelization) is first created. Each cube is then split into either five or six tetrahedra using stencils. The splitting of cubes into tetrahedral elements must be done consistently so that tetrahedra from neighbouring cubes share triangular faces. The accuracy of the merging procedure is controlled by increasing the resolution of the background grid performing an adaptive refinement of the elements close to the object surfaces.

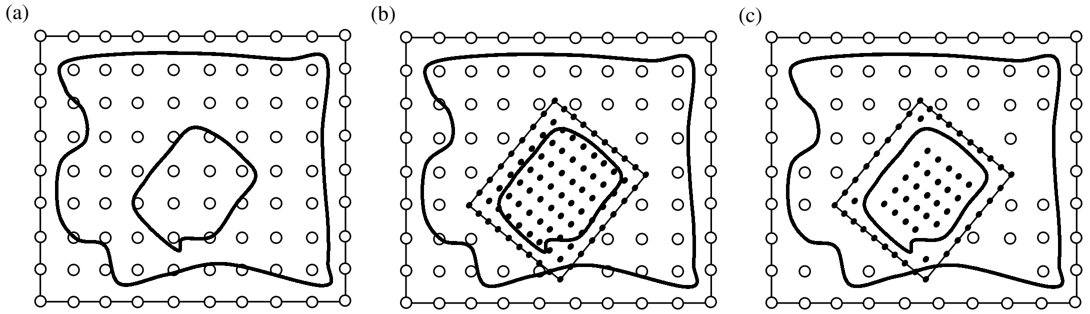

Figure 3. Creation of the background grid points: (a) voxelization of largest object (open circles); (b) removal of points and local voxelization of smaller objects (filled circles); and (c) removal of points close to the surfaces.

Multi-scale voxelization

The voxelization strategy described above works well when the objects to be merged are of similar sizes. However, in some applications it is necessary to merge objects of very different scales. Such is the case of the building and terrain presented above. The main problem in these cases is that they will require an initial background grid with very large number of voxels in order to resolve the smallest object. Therefore, it was decided to construct the initial background grid in a different way.

The creation of the background grid is carried out in two steps. The background grid points are created first and then background grid elements are constructed using a Delaunay triangulation technique [10, 11]. Background grid points are created as follows. A loop is performed over the objects to merge, from the largest to the smallest. For each object, the bounding box aligned with the object principal axes is computed. Points already created that fall within the bounding box of the current object are deleted and the bounding box of the current object is filled with points uniformly distributed. In this way a point distribution that resembles local uniform voxelization for each object is obtained. Finally, points that are too close to the object surfaces are then removed from the background grid. If the absolute distance from a point to any object surface is $<20\%$ of the size of the smallest background grid voxel, it is deleted. Once all the background grid points have been created in this manner, a Delaunay triangulation method is used to connect them to form tetrahedral elements. The process to create the background grid is illustrated in Figure 3.

Point enrichment

As mentioned above, this approach does not guarantee that features present in the initial triangulations will be preserved. This is due to the fact that the background grid points are not placed exactly on the desired features; therefore, there will always be interpolation errors. Consider, for example, the situation illustrated in Figure 4(a). Since there is no background grid point exactly on the corner of the object surface this corner will never be present in the final iso-surface. However, if a background grid point is introduced at the position of the corner, the final iso-surface does contain the corner (Figure 4(b)). Therefore, the algorithms were modified in order to introduce background grid points along surface features.

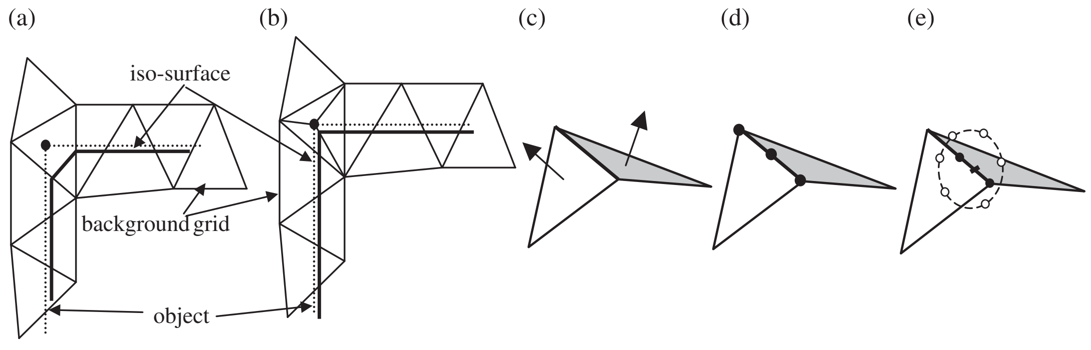

Figure 4. Feature preserving technique: (a) features not preserved with regular background grid; (b) features preserved by introducing points along ridges; (c) ridge identification; (d) introduction of points along ridges; and (e) introduction of points around ridges.

Features are detected as follows. *Ridges* are identified as edges shared by two triangles with a considerable deviation of their unit normals. In other words, if the scalar product of the unit normals of two adjacent triangles is less than a user-specified tolerance (typically $\cos(10^\circ)$), then the common edge is identified as a ridge (Figure 4(c)). Points touched by only one ridge or more than two ridges are identified as *corners*.

The key to preserve surface features is to create a proper background grid with points on the detected features. The algorithm proceeds as follows. First, background grid points are created as explained in the previous section. Then, new points are created along the detected ridges (Figure 4(d)). The separation between these points is determined by the size of the smallest background grid voxel. Extra points are then created in a ring-like disposition around ridges, between the points added along a ridge (Figure 4(e)). The radii of these rings are proportional to the background grid voxel size. Finally, the background grid elements are created using a Delaunay triangulation procedure. Background grids created using this point enrichment method tend to have tetrahedral elements with sides along the detected ridges; therefore, these ridges will be preserved in the final iso-surface.

ISO-SURFACE EXTRACTION

Traditional iso-surfacing

The iso-surface triangulation is obtained from the background grid using a classic algorithm [6]. The edges crossed by the desired iso-value (zero) are marked, and the points that will comprise the final triangulation are introduced along them. In a second pass over the elements, the number of edges marked is counted, and either one (three marked edges) or two triangles (four marked points) are introduced (see Figure 5). The triangles are oriented so that their normal points in the direction of increasing values of the distance-to-surface function.

Component-based iso-surfacing

The traditional iso-surface extraction algorithms can fail since in some cases it may so happen that the shortest distance to the extremities of a background grid edge correspond to different

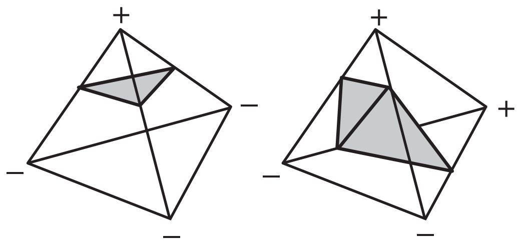

Figure 5. Construction of iso-surfaces from tetrahedral grids. The +, − signs indicate the sign of the distance map at the nodes of the tetrahedral element.

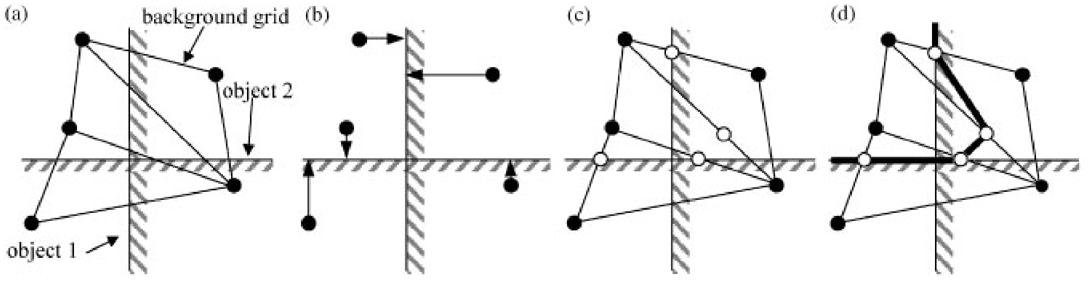

Figure 6. Iso-surface extraction: (a) object surfaces and background grid; (b) shortest distances; (c) iso-surface cuts along edges; and (d) resulting iso-surface.

object surfaces. Consider, for example, the situation illustrated in Figure 6. Figure 6(a) shows the two intersecting objects and some background grid elements. Figure 6(b) shows the shortest distance from the background grid points to the surfaces. If these distances are used to construct the iso-surface, the background grid edges are cut at the locations indicated in Figure 6(c) and the resulting iso-surface may deviate considerably from the object surfaces, as shown in Figure 6(d). This effect may be even more pronounced in three dimensions, and in some instances the iso-surface may not have the correct topology.

Since we are extracting the zero-level iso-surface, the distance between the two extremities of a background grid edge, cut by the iso-surface, must have different signs. As in the traditional algorithm, this criterion is used to first filter the edges crossed by the iso-surface. However, the location where the edge is cut by the iso-surface must be carefully computed [12]. We have found a solution to this problem, which is similar to a ray tracing approach. It requires storing the distance from the background grid points to each object surface (Figure 7(a)). For a given edge, the distance from its extremities to each object is checked for a change in sign. If only one pair of distances changes sign, it is taken to compute the iso-surface (Figure 7(b)). If more than one pair satisfy this condition, we keep the one that places the cut-point closest to the edge extremity that lies inside the domain (Figure 7(c)). The resulting cutting positions are shown in Figure 7(d).

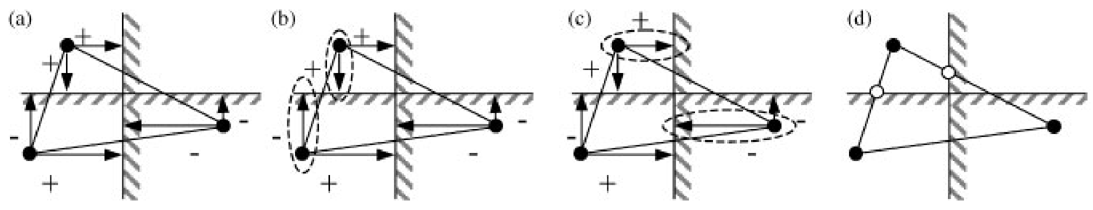

Figure 7. Cut edge decision: (a) all distances to the objects; (b), (c) cutting the edge using the circle distances; and (d) cut edges.

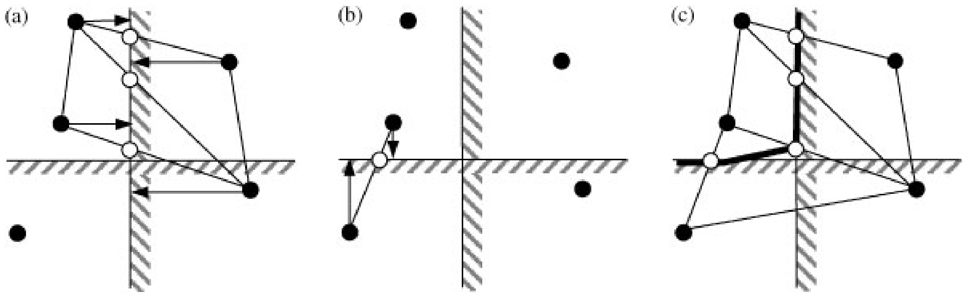

Figure 8. Iso-surface extraction: (a) distance to first object; (b) distance to second object; and (c) iso-surface.

The situation illustrated in Figure 8 demonstrates the complete process to extract the iso-surface. The distances to the first object and the corresponding cut-points are shown in Figure 8(a). The distances to the second object and the corresponding cut-point are shown in Figure 8(b) and the final iso-surface is shown in Figure 8(c).

RESULTS

Boolean operations

First, the union operation is demonstrated by merging three mutually perpendicular cylinders. Figure 9 shows: (a) the initial surface triangulations; (b) the surface triangulation of the merged surface with no edge-preservation; and (c) the surface model obtained introducing background grid points along the detected ridges in the cylinders in order to preserve these ridges.

Secondly, the subtraction operation is demonstrated using a sphere and a cylinder. The initial surface triangulations are shown in Figure 10(a), and the final surface triangulation and model in Figures 10(b) and 10(c), respectively.

In this example, a uniform voxelization was used, with no adaptive refinement, and no extra points were added to the background grid along detected features. As expected, it can be clearly seen that surface features in the original triangulations, i.e. the circular lines at the extremities of the cylinders, are not preserved in the final triangulation.

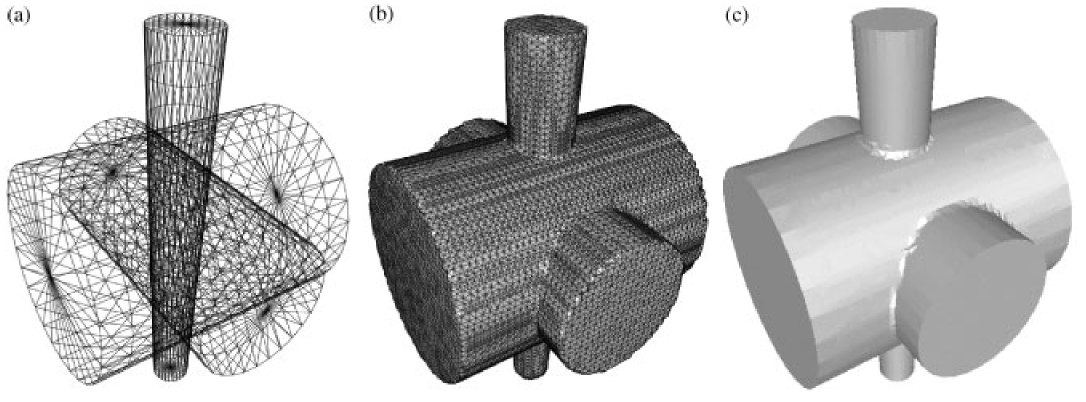

Figure 9. Union of three cylindrical surfaces: (a) initial surface triangulations; (b) merged surface triangulation with no edge preservation; and (c) surface model with edge preservation.

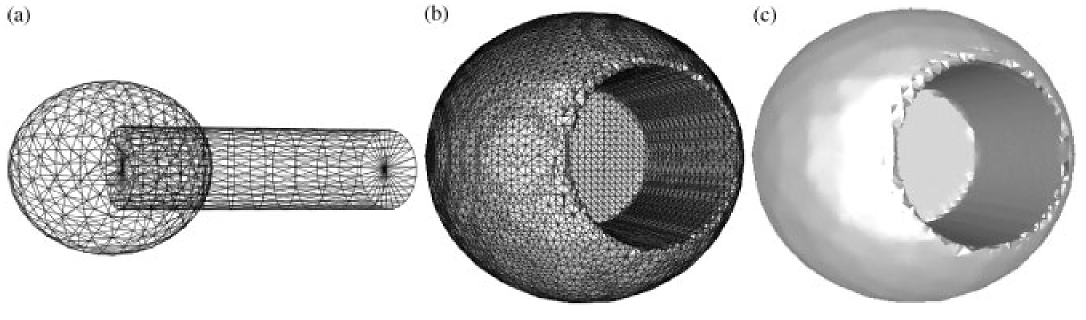

Figure 10. Subtraction operator applied to a sphere and a cylinder: (a) initial triangulations; (b) subtracted surface triangulation; and (c) final surface model.

Smooth surfaces with uniform voxelization

The methodology was applied to the merging of the surface triangulations representing different branches of a carotid artery. Each branch was independently reconstructed from magnetic resonance angiographic images of a human subject using a cylindrical deformable model [13] (Figure 11(a)). The final surface (Figure 11(b)) was obtained using a uniform voxelization with no adaptive refinement. Since all the components are smooth surfaces, there is no need to introduce any extra points into the background grid. This geometrical model was then used as a support surface to generate a volumetric finite element grid. A uniform element size distribution was specified for this mesh. The surface of the generated tetrahedral grid is shown in Figure 11(c).

Smooth surfaces with adaptive voxelizations

The adaptive voxelization technique was tested with data from a haemoglobin molecule. Each atom in the molecule was modelled as an independent sphere obtained by translating and

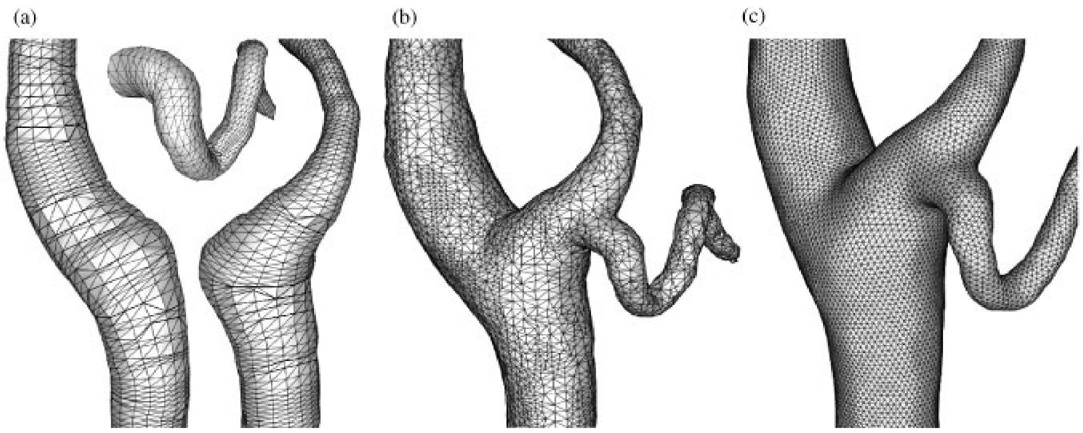

Figure 11. Merging of arterial surfaces: (a) initial triangulations; (b) merged surface triangulation; and (c) surface of tetrahedral finite element mesh.

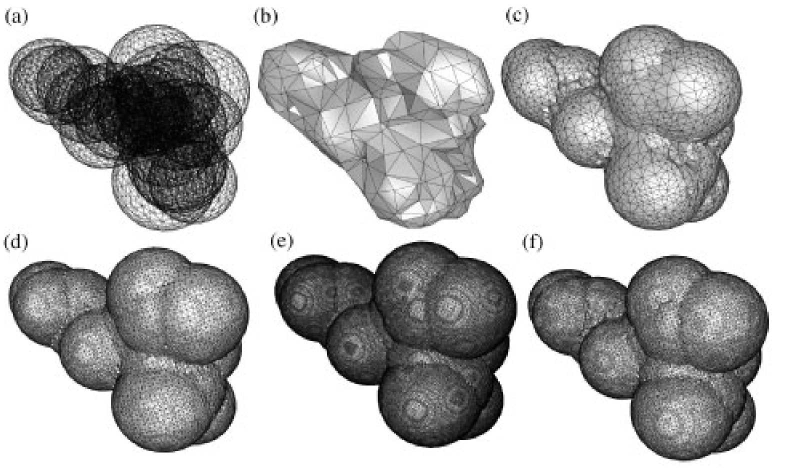

Figure 12. Merging of 20 atoms: (a) initial triangulations; (b), (c), (d), (e) merged surfaces with different background grid resolutions; and (f) merged surface after two levels of refinement.

scaling the surface tessellation of a unit sphere. The increase in accuracy obtained using different background grid resolutions was demonstrated by merging the first 20 atoms of the sphere. The initial surface triangulations are shown in Figure 12(a). The merged surfaces obtained using a uniform voxelization of 1, 11, 250 and 2500 K points are shown in Figures 12(b), 12(c), 12(d) and 12(e), respectively. Figure 12(f) shows the surface triangulation obtained using an adaptive background grid, which after two levels of refinement contained 150 K points.

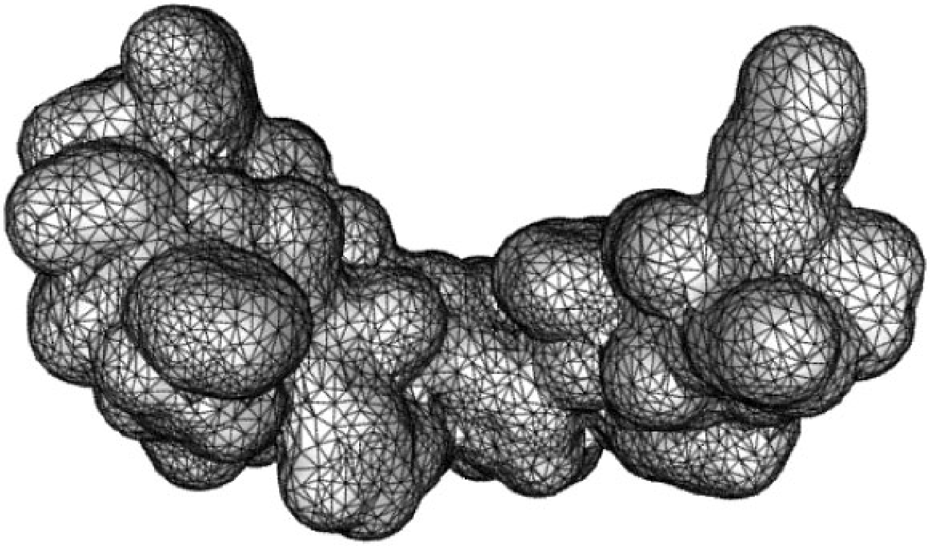

Figure 13. Surface triangulation of haemoglobin molecule obtained by merging 300 atoms.

Finally, the merging procedure was applied to the complete molecule, consisting in 300 atoms. The merged surface was obtained with no background grid refinement (Figure 13).

Feature preserving voxelizations

The modifications introduced into the voxelization algorithm in order to preserve features present in the original triangulations were tested using the surface of a cube. Since a single surface was considered, the procedures should be able to reproduce exactly the same surface. The initial triangulation consisted in the simplest triangular tessellation of a unit cube, using 2 triangles per face (Figure 14 (a)). The features detected in this initial triangulation were, as expected, the sides of the cube (Figure 14(b)). Once a uniform voxelization was created, points were introduced along the detected ridges as explained above. The separation between these points was taken as the element size of the uniform voxelization. The final background grid is shown in Figure 14(c). Note the background grid points located on the cube sides. The extracted iso-surface is shown in Figure 14(d).

Multi-scale feature-preserving voxelizations

Finally, the methodology was applied to the merging of terrain elevation data with three surface triangulations representing buildings. Figure 15(a) shows the triangulation of the terrain data, and Figure 15(b) shows a superposition of the terrain data and the buildings. The multi-scale voxelization method described above was used to construct the initial background grid. The points of this background grid are shown in Figure 15(c).

The point enrichment procedure described above was used to preserve ridges in the final merged surface. The background grid was adaptively refined close to the building surfaces. The extracted iso-surface and the preserved ridges after two levels of refinements are shown

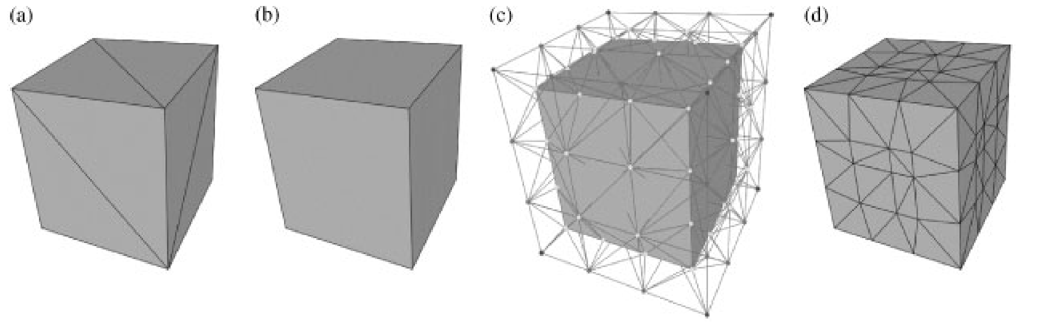

Figure 14. Cubic surface: (a) initial tessellation; (b) detected ridges; (c) background mesh with point along ridges; and (d) iso-surface.

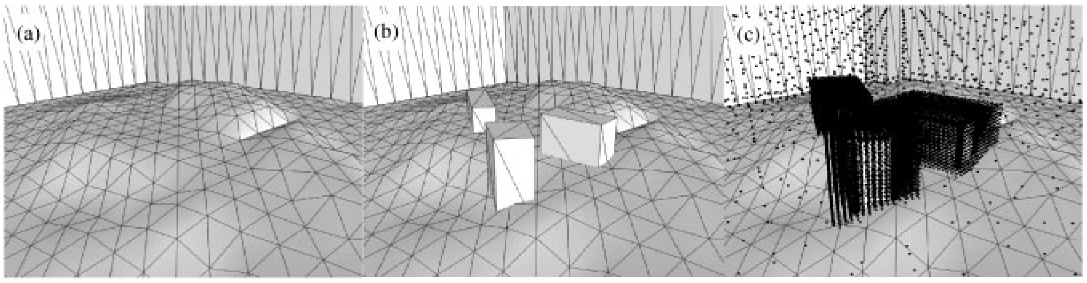

Figure 15. Merging three buildings and a terrain: (a) initial terrain triangulation; (b) superposition of terrain and building triangulations; and (c) initial background grid points.

in Figures 16(a) and 16(b). The triangulation of the merged surface is shown in Figure 16(c). This triangulation was then used as a support surface to generate a finite element grid of tetrahedral elements using an advancing front technique [14, 15]. The surface of this volumetric grid is shown in Figure 16(d).

The element size distribution was specified using line sources along the building sides and close to the surface intersections. The finite element grid contained approximately 900 K elements and 170 K points.

In order to demonstrate that the generated grid can be used to conduct finite element calculations, a steady state run was performed solving the laminar, incompressible Navier–Stokes equations [16]. A uniform velocity profile was specified at the inflow boundary and no-slip conditions were specified on the ground and the building walls. Results of this simulation are shown in Plate 1, where the flow pattern behind the building is visualized using streamlines colored according to the fluid velocity magnitude.

DISCUSSION

Several extensions and improvements to our previously developed surface merging procedures were presented. The original algorithms were based on the extraction of iso-surfaces from a

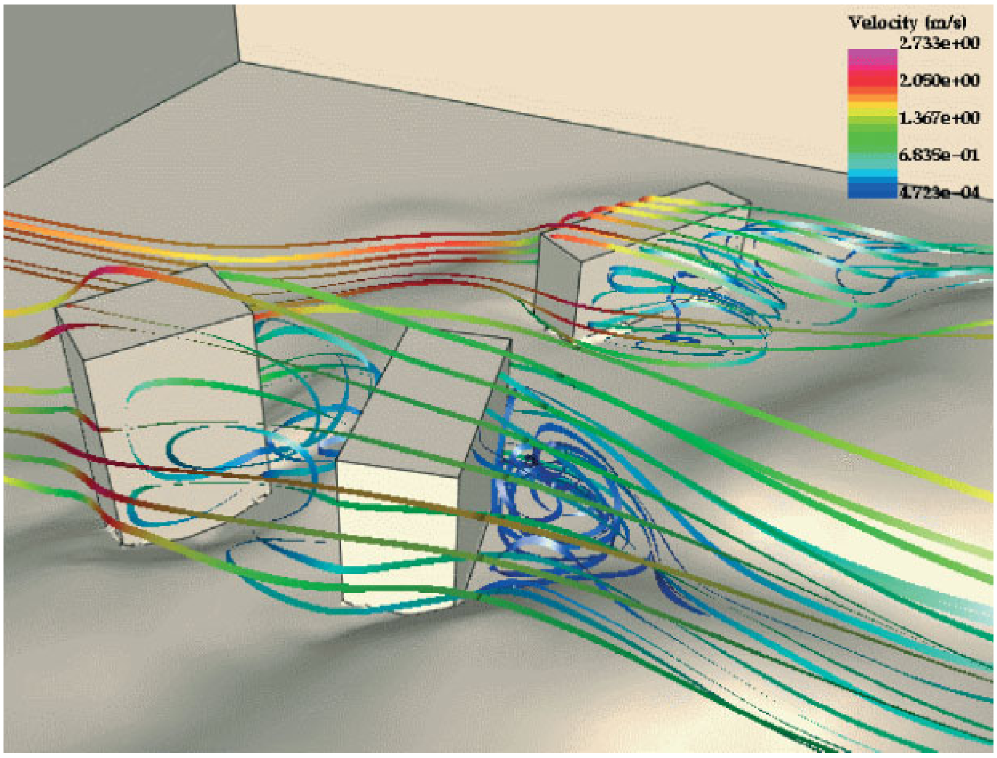

Plate 1. Flow visualization: streamlines colored according to velocity magnitude showing flow recirculation regions behind the buildings.

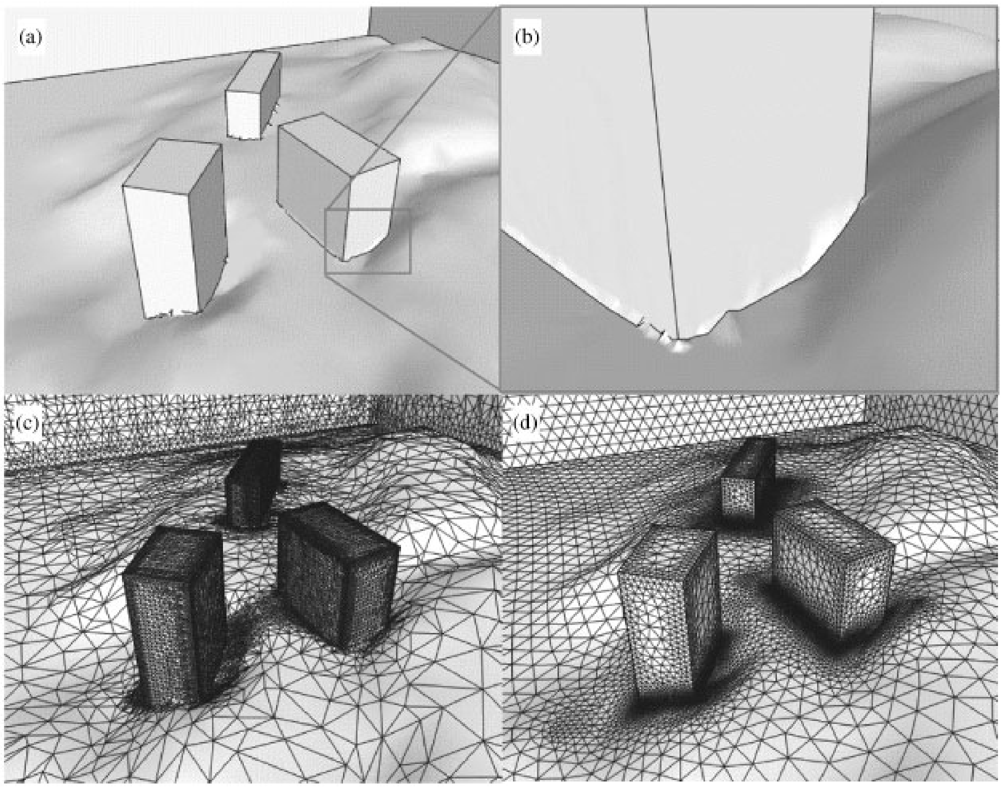

Figure 16. Merging three buildings and a terrain: (a) extracted iso-surface and preserved ridges; (b) close-up view of showing the terrain-building intersection; (c) iso-surface triangulation; and (d) the surface of tetrahedral mesh.

distance map defined on an adaptive background grid. They were originally devised to merge surface triangulations representing arterial branches, reconstructed from medical images. The main limitation was that only smooth surfaces could be properly treated. That is, features such as ridges and corners present in the original triangulations were not preserved in the final merged surface.

In this paper a new methodology to construct the initial background grid has been presented. Points are first created following local object voxelizations and then connected using a Delaunay triangulation algorithm. A point enrichment strategy is then used to introduce new points into the background grid along detected surface features, i.e. ridges. The background grid is then adaptively refined close to the object surfaces and the distance to the surface function is computed. The final merged surface is obtained by extracting the zero-distance iso-surface. This approach not only yields proper grids when merging objects of different scales, but also preserves surface features. Since the distance map is computed as the shortest distance to any object surface, the iso-surface extraction algorithms had to be modified in order to obtain the correct merged surface. The implementation of other Boolean operations besides the merging (union) has also been described.

The algorithms were demonstrated with various examples, ranging from simple geometrical entities to complex engineering applications. Two main limitations of these procedures can be identified. The first is that they work only with closed surfaces since it is necessary to decide whether a background grid point lies inside or outside a given object in order to properly define the sign of the distance map. Secondly, the intersection line between two objects is never represented exactly. In many applications, this may not be a limitation; it may even be desirable since an exact representation of the intersection may introduce very small-scale geometric features that are not relevant to the numerical simulation. Thus, the smoothing obtained in the intersection region could facilitate the volumetric grid generation. However, if the application does require an exact representation of the intersection lines, a rather simple extension could be added that will enforce preservation of intersections. The points that lie in the intersection between two surfaces may be found by computing the intersections between all edges of one surface with all triangles of the other, and vice versa. Introducing these points (or some of them) into the background grid, in a similar fashion as those added along detected ridges, will ensure the preservation of intersection lines.

In conclusion, the presented algorithms allow realistic modelling of a large number of complex engineering geometries using overlapping components defined discretely, i.e. via surface triangulations. This capability is very useful for grid generation starting from data originated in measurements or images.

REFERENCES

1. Lo HS. Automatic mesh generation over intersecting surfaces. *International Journal for Numerical Methods in Engineering* 1995; **38**:943–954.
2. Shostko A, Löhner R, Sandberg W. Surface triangulations over intersecting geometries. *International Journal for Numerical Methods in Engineering* 1999; **44**:1359–1376.
3. Nooruddin FS, Turk G. Simplification and repair of polygonal models using volumetric techniques. *GVU Technical Report No. GIT-GVU*-99-37, Georgia Tech., 1999.
4. Sramek M, Kaufman AE. Alias-free voxelization of geometric objects. *IEEE Transactions on Visualization and Computer Graphics* 1999; **5**(3):251–267.
5. Cebral JR, Löhner R, Choyke PL, Yim PJ. Merging of intersecting triangulations for finite element modeling. *Journal of Biomechanics* 2001; **34**:815–819 .
6. Löhner R, Parikh P, Gumbert C. Some algorithmic problems of plotting codes for unstructured grids. *AIAA*-89-1981-*CP*, 1989.
7. Hoppe H, DeRose T, Duchamp T, McDonald J, Stuetzle W. Mesh Optimization. *Proceedings of the SIGGRAPH* 93, 1993; 19–26.
8. Cebral JR, Löhner R. From medical images to anatomically accurate finite element grids. *International Journal for Numerical Methods in Engineering* 2001; **51**:985–1008.
9. Löhner R. Some useful data structures for the generation of unstructured grids. *Communications in Applied Numerical Methods* 1988; **4**:123–135.
10. Owen SJ. A survey of unstructured mesh generation technology. *Proceedings of the Seventh International Meshing Roundtable*, Sandia National Laboratory, 1998; 239–267.
11. George PL, Borouchaki H. *Delaunay Triangulation and Meshing*. Editions Hermes: Paris, 1998.
12. Kobbelt LP, Botsch M, Schwanecke U, Seidel HP. Feature sensitive surface extraction from volume data. *Computer Graphics Proceedings*, Annual Conference Series 2001; 57–66.
13. Yim PJ, Cebral JR, Mullick R, Choyke PL. Vessel surface reconstruction with a tubular deformable model. *IEEE Transactions on Medical Imaging* 2001, submitted.
14. Löhner R. Regridding surface triangulations. *Journal of Computational Physics* 1996; **126**:1–10.
15. Löhner R. Automatic unstructured grid generators. *Finite Elements in Analysis and Design* 1997; **25**:111–134.
16. Soto O, Löhner R, Cebral JR, Codina R. A time-accurate implicit monolithic finite element scheme for incompressible flow problems. *ECCOMAS CFD*, Swansea, U.K., September 2001.