Automated Hexahedral Mesh Generation of Complex Biological Objects

Barry Canton and Michael D. Gilchrist ·

School of Electrical, Electronic & Mechanical Engineering, University College Dublin, Ireland

Abstract

This paper describes the development of a new automatic mesh generator for complex biological objects based on voxel hexahedral meshing (VHM). The system produces hexahedral meshes with multiple material classifications from three-dimensional (3D) computed tomography (CT) datasets. The quality of the VHM meshes is improved using a series of mesh connectivity improvement algorithms, leading to the application of a new and novel boundary smoothing algorithm. The boundary smoothing algorithm classifies boundary nodes into different groups based on the configuration of their neighbouring elements: this provides a varying compromise between smoothness and element distortion. The high anatomic detail of which the method is capable is shown for meshes of single and mixed material types. This system illustrates how to harness the speed and simplicity of VHM as an initial mesh generator while simultaneously improving the quality of those meshes with additional meshing techniques. MATLAB code for this software is freely available for research purposes upon request.

Keywords: Mesh generation; Laplacian smoothing; voxel mesh; hexahedral element

· Corresponding author. Email: michael.gilchrist@ucd.ie

1. Introduction

Finite element (FE) simulation is a key tool for injury biomechanics research. It is one of the most important techniques for evaluating the tolerance of the human body to impact. A major obstacle to its application is the difficulty of generating finite element meshes that are both anatomically accurate and numerically efficient and well conditioned. An automated method of generating high quality meshes of biological objects from medical images has long been a goal of the scientific community.

Many researchers [Taylor et al., 2002; Kwon et al., 2003; Duprey et al., 2007] have chosen to generate meshes using tetrahedral elements due to the relative ease of automatically meshing an object with tetrahedrons. It has been shown, however, that hexahedral elements often outperform tetrahedral elements [Benzley et al., 1995; Ramos & Simoes, 2006]: these instances can be of particular relevance to injury biomechanics. A robust and automatic method of generating hexahedral meshes of arbitrarily complicated objects has yet to be developed. The human head is particularly difficult biological object to mesh and is the primary focus of this paper. The best hexahedral meshes which exist of the head are of a generic anatomy and have involved much manual construction. The typical method is to extract 3D contours from medical image sets and use a combination of automatic and manual techniques to discretise the 3D solid [DiMasi et al., 1991; Kang et al., 1997; Verhoeve, 1998; Zhang et al., 2001; Kleiven & van Holst, 2002; Horgan & Gilchrist, 2003, 2004; Gelaude et al., 2006]. This typically involves dividing the object into simpler topological entities prior to discretisation. This process is iterative, hard to automate and requires expertise on the part of the user. These models can take a number of years to develop. It would be desirable to be able to rapidly produce highly patient-specific models, as the ability of a generic mesh to accurately predict the behavior of a specific object is unconfirmed. Some of the most advanced automatic hexahedral mesh generating algorithms such as plastering [Blackers & Meyers, 1993], whisker weaving [Tautges et al., 1996], or dual cycle elimination [Muller-Hanneman, 1998] are not yet able to mesh extremely complex topologies of which the human head and skull are prime examples.

The voxel hexahedron meshing (VHM) algorithm first proposed by Keyak et al [1993] is a robust and highly automated algorithm for constructing all-hexahedral meshes from segmented medical image sets. It operates by generating a regular hexahedral element for every voxel of interest in the image set. Voxels of interest can be defined by any image segmentation technique. By utilizing the fact that an image set is a discretised representation of the object, the method avoids the problem of extracting 3D surface contours from medical images and then trying to discretise the solids formed by those contours. Although this means the method will operate robustly on any object, there is a trade off because the perfect hexahedral elements lead to a stepped or 'digitized' mesh surface. The errors introduced by this fact are particularly significant when modeling objects with relatively thin cross-section. This inaccurate surface representation may not prevent the mesh from accurately describing the global behavior of the object but local results can be quite inaccurate [Guldberg et al., 1998]. A number of authors [Krabbel & Muller, 1996; Camacho et al., 1997; Grosland & Brown, 2002; Kaminsky et al., 2005] attempted to harness the simplicity and robustness of the VHM algorithm and subsequently improve the quality of the resulting surface using boundary smoothing. In this paper we describe a new system, based on VHM, which uses novel and improved techniques to improve the quality of the meshes while retaining the high level of automation and rapidity that makes VHM such an attractive meshing option.

2. Methods

All code was developed using the MATLAB programming language and image processing toolbox (The Mathworks, Natick, Ma, USA). Using the MATLAB image processing toolbox, grayscale images in a wide range of formats can be accepted as input data. The purpose-written code, which is freely available upon request, outputs meshes in the neutral file format used by PATRAN (MSC Software Corporation, Santa Ana, Ca, USA) or as an ABAQUS input deck (Abaqus, Inc., Pawtucket, RI, USA). The code consists of two main components: an image segmentation module and a mesh generation module called DICER. The operation of the image segmentation system will be described briefly before looking at DICER in more detail.

2.1 Image Processing

The image segmentation system is designed to operate on computed tomography (CT) image sets, but with some modification could be adapted to accept magnetic resonance (MR) image sets. A range of general segmentation schemes exist: thresholding, edge detection [Canny, 1986], deformable models [Kass et al., 1988; Chan & Vese, 2001] and region growing (see [Gonzalez & Woods, 2002] for a general review of all of these). Being the most robust, thresholding was chosen as the main segmentation technique. Despite its simplicity relative to the more advanced techniques it is still capable of providing accurate segmentation results for the kind of image sets currently being considered. Thresholds were set using the shape connectivity method first proposed by Lie [1995]. Other alternatives for threshold finding can be found in [Dempster et al., 1977; Tesuo et al., 1998]. The shape connectivity method attempts to optimize thresholds by maximizing the solidity of objects and minimizing the "busyness" of the segmented images. The gray level co-occurrence matrix [Haralick et al., 1973] is required by this method. The algorithm searches for optimal thresholds within a user-specified number of gray level ranges. Although these ranges require some *a-priori* knowledge on the part of the user, they can be relatively wide, so little user expertise is required.

This global thresholding process results in an image set where each voxel has been classified into one of the user specified material classes. Morphological operations are then applied to improve the local connectivity of the mesh and remove any locally misclassified voxels. These operations are opening by reconstruction [Vincent & Soille, 1991] and morphological closing.

Typically, the resolution of the dataset must be reduced at this point due to constraints on the numbers of elements in the resulting FE meshes. Following resolution reduction, opening by reconstruction and closing are once again applied to improve the connectivity of thin sections that may have deteriorated after the resolution reduction process. An alternative to reducing the resolution of the dataset will be discussed later.

Figure 1 shows sample images at various stages of the segmentation process. The original image is taken from the visible female dataset (National Library of Medicine, Bethesda, Md). The segmented dataset is output as a 3D array of voxel classifications. The physical size of the voxels is also output as a three-element vector. This is the format accepted as input by DICER. The simplicity of the data output format makes it straightforward to transform the output of any image segmentation system into the required format for the DICER module.

2.2 Meshing System

At the heart of DICER, which is summarised in the flowchart of Figure 2, is a VHM algorithm. However, a number of additional functions are used to improve the quality of the resulting meshes. The first step in the mesh generation process is to ensure that all voxels (and hence elements) are well connected to their neighbours. Poorly connected elements include those where an element is only joined to its neighbour by one edge (see Figure 3(A)), or when two diagonally opposed elements in a 2x2x2 block of elements are of a different classification to the other six elements (see Figure 3(B)), or where a single element is connected to its neighbours by just two adjacent faces (see Figure 3(C)).

Three algorithms were written to search for and improve the connectivity of these regions of the mesh. Two of these algorithms operate by searching all 2D planes or all unit blocks in the dataset to find poorly connected regions of the mesh (a unit block is centered on a node and is composed of the eight elements that share that node). Elements of the appropriate classification are then added to improve the mesh connectivity. These algorithms are then repeated a number of times for each element classification until no further changes occur. Typically, this requires two to three iterations for each classification. These connectivity improvement algorithms are described in more detail in Box 1. The algorithm to modify the arrangement of Figure 3(C) does not operate on the voxel set but rather operates on the nodes and elements of the mesh. As a result it requires data structures and concepts that will be described later in the context of boundary smoothing. The result of applying these algorithms is a dataset where the voxels yield a well-connected structure but which is neither smooth nor optimized for smoothing.

2.3 Plane-merging

As has been reported previously the major disadvantage of VHM is the stepped surface that occurs when two adjacent planes of elements in 3D, or rows of elements in 2D, have different arrangements of elements as shown below in Figure 4(A) where the top row has eight consecutive elements while the bottom row has two groups of three adjacent elements. These configurations are very common but lead to problems when boundary smoothing is performed as shown in Figure 4(B). The four labeled elements are severely distorted by any attempt to smooth the surface. Any smoothing algorithm will have to compromise between boundary smoothness and element distortion. An approach that has not been taken previously with VHM is to alter the configuration of the mesh itself. An algorithm was developed to merge adjacent planes of elements, by essentially deleting all internal nodes that are shared by elements of each row or plane. An internal node is any node that is shared by exactly eight elements in 3D, or four elements in 2D. The result of this procedure for the 2D mesh sample of Figure 4(A) is shown in Figure 4(C). By combining the elements of two rows into one single row, the boundary of the row of elements now more closely follows the curvature of the original object. When boundary smoothing is applied to this element configuration, as in Figure 4(D), it can be seen that element distortion has been greatly reduced, yet the loss of surface detail is minimal and the smoothed boundary profile is almost identical to the results before plane-merging. This is because very few nodes that lie on the boundary of the region have been removed. Note also that the number of elements has been significantly reduced. When this algorithm is applied throughout the mesh and along all three principal directions, the effects are the same – a large reduction in the number of distorted elements, and a minimal reduction in the number of nodes on the mesh surface but

with a large reduction in the total numbers of nodes in the mesh. In many instances, a number of iterative plane merges can be performed. The steps of this algorithm are outlined in Box 2. Currently, this algorithm can only operate on meshes with one material class.

2.4 Boundary Smoothing

Following plane-merging, the surface of the mesh is still relatively highly stepped and angular. We present here the algorithm developed to smooth the surface of the mesh. Before describing the smoothing process, some definitions should be introduced. In the 3D orthogonal mesh created by generating an element from each voxel of interest, each node is shared by a maximum of eight elements in a 2x2x2 arrangement. This property is preserved by the connectivity improvement and plane-merging algorithms described above. This eight-element block will be referred to as a unit block. A node is termed a boundary node and should be moved to smooth the mesh surface if all elements in its unit block are not of the same classification. A node can form up to six edges with its neighbouring nodes. Each of these edges can be shared by up to four elements. If an edge is not shared by four elements of the same classification then it is termed a boundary edge. By definition, if a boundary node shares a boundary edge with one of its neighbours, that neighbour must also be a boundary node and is called a boundary neighbour of the former node. In this way, the boundary nodes and boundary edges form a 2D graph for a 3D mesh. In a mesh with multiple material classifications, there may be more than one connected graph of nodes. A boundary node should be moved relative to its boundary neighbours to smooth the mesh surface.

The goal of smoothing is to reduce the discontinuities in the spatial second finite difference of this graph of boundary nodes without overly distorting the elements they form or changing the global shape of the mesh. This is achieved in the current system by moving each boundary node an appropriate distance in the direction of its boundary neighbours using the equation given below, essentially moving the node to a weighted average of its original position and the centroid of its boundary neighbours' final positions.

$$\vec{r}(x_n^f) = c_1\vec{r}(x_n^i) + c_2 \sum_{j=1, j\neq n}^{R} \vec{r}(x_{n,j}^f)$$

$$c_1 = 1 - k,$$

$$c_2 = k / R,$$

$$0 \le k < 1$$

Here, $\vec{r}(x_n^i)$ and $\vec{r}(x_n^f)$ are the initial and final positions respectively of the nth node. R is the number of boundary neighbours for node n. k is the smoothing or weighting coefficient: this governs the tradeoff between element distortion and mesh smoothness. A value of 0 corresponds to no smoothing and a value of 1 with no further constraints will collapse all the nodes to a single point in space. The appropriate choice of smoothing coefficient will be discussed later. By assembling this equation for each boundary node, smoothing is reduced to solving a system of simultaneous linear equations of the form Ax=b, where A is a coefficient matrix of the weighted contributions of the boundary neighbours, b is the vector of weighted contributions of the nodes' original positions and x is the unknown vector of the nodes' final, smoothed positions. This system of equations is solved for each of the spatial dimensions. With this method, smoothing is

spread continuously throughout the mesh, whereas in an independent equation method, such as that described in [Camacho et al., 1997], smoothing will only occur at those nodes where there is a discontinuity in the second finite difference typically leading to many flat regions in the mesh. The simultaneous linear equation method could be approximated by an iterative application of an independent equation system. However, the iterative approach makes it harder to ensure that nodes move correctly relative to their boundary neighbours, which are also moving, since a node will not be "aware" of how its neighbours are moving.

It was important to decide how to set the smoothing coefficient. It was realized that certain nodes should move less than others, depending on the configuration of their unit block. This is because the arrangement of boundary neighbours can cause a node to move a significant distance in a direction that rapidly distorts the elements of which it forms a part. A balance must be struck between smoothing the surface and not overly distorting elements lying on the surface. In order to decide the extent to which a node should move, it is necessary to be able to classify it based on its neighbourhood. For a single material mesh, the unit block for a node can have 2^8=256 different configurations where each element location can contain an element, or be free space. By realizing that many of these are inverses of each other, or merely rotations in space of each other, that number can be greatly reduced. Since the connectivity improvement algorithms further reduce the number of allowable configurations, the total number of possible configurations can be reduced to six (ignoring the trivial case where all eight elements are of the same classification, such that the node is an internal node). These six cases are shown in Figure 5. The smoothing algorithm sorts all boundary nodes into these categories by counting the number of boundary neighbours for each node. This information is found from the nodal adjacency matrix, formed by considering the set of boundary nodes as a graph of connected nodes. The algorithm distinguishes between cases B and D, which both have four boundary neighbours, by finding the centroid of the boundary neighbours and comparing this to the central node's position, which for case D are coincident points. In the case of D, four elements share the edge between the central node and the bottom node, so that is not a boundary edge and consequently the bottom node is not a boundary neighbour. In the case of a multiple material mesh, it is possible that a unit block could contain more than two element classifications. Even in this unlikely event, the algorithm can still correctly classify the node into one of the six categories shown above, based on the dissimilarity of elements sharing each edge.

Once all boundary nodes have been classified, the system of linear equations can be solved using an appropriate smoothing coefficient for each classification. For example, movement of case D nodes causes negligible element deformation. This is because the arrangement of nodes around it stabilizes its motion and so the smoothing coefficient can be high, a value of 0.95 for the smoothing coefficient appears to be suitable. On the other hand, case A and B nodes must move a considerable distance to produce a smooth surface but it is these nodes that also lead to the most severe element distortion. A coefficient between 0.8 and 0.9 appears to be the best compromise in this instance. At the other extreme, case F nodes are "stabilized" by boundary neighbours in each direction and so will typically move a very small amount regardless of the smoothing coefficient. At this point it is appropriate to return to the third mesh connectivity algorithm. If an element contains two class A nodes (see Figure 3(C)) and these nodes are connected to one another, then the smoothing algorithm will cause that element to become extremely distorted for any reasonable smoothing coefficient. Hence, it was decided to delete any such elements. Typically they occur less than 10 times in a mesh of ~100000 elements so their removal has a negligible effect on mesh geometry. Box 1 describes

the operation of this algorithm. The implementation of the smoothing algorithm is outlined in Box 3.

2.6 Internal Smoothing

The process of boundary smoothing can move some boundary nodes very close to the internal nodes with which it forms elements, thereby leading to element distortion. Element distortion can be reduced somewhat by fixing the positions of all boundary nodes after boundary smoothing and then smoothing the locations of all nodes not lying on the surface.

Many algorithms exist to smooth the position of internal nodes in a mesh, see [Owen, 1998] for a general review. One of the simplest methods is Laplacian smoothing. Laplacian smoothing can give undesired results when the nodes are very poorly arranged. In the current case however, all internal nodes are initially arranged in a perfect orthogonal arrangement and are all connected to the optimum number of neighbours, meaning that Laplacian smoothing is more than adequate for the present purpose. Laplacian smoothing is the basis of the boundary-smoothing algorithm but in the case of internal smoothing it is considerably simpler because there is no need to classify nodes and all nodes can move all the way to the centroid of their neighbours' final positions. This is because the fixed positions of the boundary nodes provide sufficient boundary conditions to constrain the movement of the internal nodes.

This completes the mesh generation process. The nodes are now transformed from relative positions into actual physical locations, the mesh displayed in the user interface and output in the desired format. The option exists to output 8-noded, linear hexahedral elements, or 20-noded quadratic hexahedral elements.

3. Results

The high anatomic detail of the resulting meshes can be seen in Figure 6 and Figure 7 for both a single and two material type mesh. The anatomic detail that can be achieved is limited only by the quality of the image segmentation results and the number of elements permitted in the final mesh. Some of the benefits of plane-merging can be seen in Figure 6. This shows three cutaway meshes of the same skull. Mesh A has a large number of elements (135315) and so can be considered to be a very accurate representation of the object in the segmented image sets. The second mesh, Figure 6(B), shows the same mesh following one plane merge in each direction. The number of elements has been reduced to 25489 without significant loss of anatomic detail and it has reduced the number of distorted elements. The third mesh, Figure 6(C), shows a mesh built with similar element numbers but without any plane merges. This necessitates the use of a much lower resolution dataset and it can be seen that this causes a significant loss of anatomic detail and requires certain areas of the mesh to be artificially thickened to produce a well-connected mesh. By reducing the number of elements, plane-merging allows the use of a far higher resolution segmented dataset, resulting in greater detail being preserved from the medical images.

To evaluate the quality of the DICER meshes, some simple 2D tests were performed based on standard problems in stress analysis. Similar to the test carried out in [Guldberg et al., 1998], a semi-infinite plate with a circular hole was modelled. A quarter plate was meshed using both PATRAN's inbuilt paver algorithm and the Dicer system. The physical arrangement is shown in Figure 8(a). Dicer meshes were built at a relatively coarse mesh density where element

length was equal to one seventh of the hole radius. Smoothed and unsmoothed meshes were constructed corresponding to this mesh density. Smoothing was restricted to the rim of the hole. Mesh detail in the vicinity of the hole is shown in Figure 8(b)-(d). All meshes were analysed using linear 2D continuum plane stress elements, of reduced integration element formulation, in the ABAQUS solver.

The normal stress along edge A and the stress in the x direction around the rim of the hole were examined. The variation in normal stress along edge A relative to the applied stress, σ_0, is shown in Figure 9 for the PATRAN mesh and Figure 10 for the unsmoothed mesh. Both graphs agree closely with analytical predictions where normal stress is expected to equal the applied stress far from the hole and to increase up to three times the applied stress at the edge of the hole. The PATRAN mesh shows a peak error of 3%. As expected, the unsmooth DICER mesh shows similar results far from the hole where the meshes become almost identical. In the vicinity of the hole, the stress is under predicted by the DICER mesh. Since element quality is perfect, this must be due to the inaccurate representation of the hole geometry. Figure 11 shows how this peak stress for the model varies with the degree of smoothing applied. It can be seen that the peak stress, normalized by the applied stress, gradually approaches the correct value of three near a smoothing coefficient of 0.8 and above 0.9 it rapidly overshoots. This is due to the fact that at high smoothing coefficients the shape of the hole becomes severely distorted as the nodes all move towards the center of curvature, finally forming a straight line at a smoothing coefficient of 1.

The effect of smoothing and mesh density can be seen more clearly by examining the stress in the x direction around the rim of the hole. This stress can be predicted analytically [Roylance, 1996] and is given by $\sigma_x = \sigma_0(1 - 2\cos 2\theta)\sin\theta$, where θ is the angle made with the x-axis. Figure 12 shows the analytical result along with the prediction of the unsmoothed DICER mesh. The DICER mesh is accurate at the edge nearest to the applied stress but shows large fluctuations in stress towards the left end of the rim. This is due to the stepped edges, meaning some nodes hang "freely" and experience very little stress (see nodes 8, 11 and 15 in Figure 8(c)). Smoothing this mesh with a coefficient of 0.8 clearly improves results (see Figure 13), most notably at the aforementioned nodes. Note that the peak stress at node 15 corresponds to the peak stress plotted in Figure 10.

These results indicate that while mesh smoothing can substantially improve global accuracy and often also local accuracy, the ensuing distorted elements can increase local errors in certain areas of the mesh, although by an amount that is less than in an unsmoothed mesh. The general conclusions which can be made from these results are that for static, small displacement analyses, a smoothed voxel mesh with reduced integration formulation elements yields superior results to an unsmooth voxel mesh. These results would suggest that by using reduced integration elements, optimal results can be obtained with a smoothing coefficient of approximately 0.9.

4. Discussion and Conclusions

In common with previous VHM systems, the system just described offers the ability to automatically, rapidly and robustly generate highly anatomically accurate FE meshes direct from medical images. New features, namely an improved boundary smoothing algorithm and the ability to retain anatomic detail from image sets of higher resolution than could be used with previous VHM systems make the new system a significant improvement over those previously

published. We suggest that VHM should not be viewed as a method in itself but rather as an initial step in mesh generation, providing a convenient way to reliably discretise a complex object into a well connected mesh of elements. Additional techniques such as plane-merging can then be used to modify that mesh and remove many of the elements likely to become highly distorted by the smoothing process.

The smoothing algorithm operates correctly on all mesh configurations assuming they have passed through the connectivity improvement algorithms. This is an improvement over earlier algorithms that failed to perform correctly on certain combinations of curvature in the different directions. By considering different configurations of boundary nodes differently, the algorithm compromises between mesh distortion and surface quality. This is enabled by the classification of nodes into a small number of simple groups. By using a simultaneous equation approach, nodes automatically move correctly relative to their neighbours. This avoids the problem of earlier algorithms where the degree of smoothing possible was constrained by the necessity to ensure that no node would move too far relative to its neighbours. By using a linear equation approach, very fast solver algorithms which exist can be used, meaning it executes in a matter of seconds even for meshes with approximately 100000 nodes.

Plane-merging is a novel technique that significantly reduces the number of highly distorted elements, has a minimal effect on surface description and reduces the number of degrees of freedom in the model. By reducing the number of degrees of freedom, it becomes possible to use segmented image sets of significantly higher resolution. This is of particular relevance for objects, such as the skull, that have some very thin sections. If the image resolution must be reduced too much, thin sections can be lost from the original images. As planes will only be merged in regions that are suitable, the algorithm introduces, for the first time in VHM, the ability for mesh refinement. Areas of the mesh with fine detail will retain a small element size, whereas in simpler, thicker regions, multiple plane-merges will be performed, greatly reducing mesh density. Currently, the plane-merging algorithm attempts to merge planes uniformly throughout the mesh, stopping only in regions where merging would lead to poor nodal connectivity. It is envisaged that a more intelligent algorithm that merges planes selectively by assessing where the greatest reduction in distorted elements would be achieved would make this technique even more powerful. The other processes such as internal smoothing and connectivity improvement, all serve to produce meshes of a higher quality than previously seen with VHM systems, yet are still entirely automated.

The results presented here demonstrate the geometric quality of meshes produced by the system. It must also be stated that they show that highly distorted elements still render the local accuracy of VHM meshes less than satisfactory for some applications. This suggests that finding a compromise between boundary smoothing and element distortion is not sufficient to produce reliable local accuracy. This places the emphasis for future work on techniques such as plane-merging that attempt to modify the configuration rather than just the shape of elements in the mesh. The MATLAB code described in this paper is freely available for non-commercial research purposes from the corresponding author.

References

Benzley SE, Perry E, Merkley K, Clark B, Sjaardama G, 1995. A comparison of all hexagonal and all tetrahedral finite element meshes for elastic and elastoplastic analysis. Proceedings of the 4th International Meshing Roundtable; Sandia National Laboratories, New Mexico.

Blacker TD, Meyers RJ, 1993. Seams and wedges in plastering: A 3D hexahedral mesh generation algorithm. Engineering with Computers. 2(9):83-93.

Camacho DL, Hopper RH, Lin GM, Myers BS, 1997. An improved method for finite element mesh generation of geometrically complex structures with application to the skullbase. J. Biomechanics, 30(10):1067-1070.

Canny J, 1986. A computational approach to edge detection. IEEE Trans. Pattern Anal. Mach. Intell. 8(6):679-698.

Chan TF, Vese LA, 2001. Active contours without edges. IEEE Trans. Image Process. 10(2):266-278.

Dempster A, Laird N, Rubin D, 1977. Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society. 39(1):1-38.

DiMasi F, Marcus J, Eppinger R, 1991. 3D anatomic brain for relating cortical strains to automobile crash loading. Proceedings of the 13th International Technical Conference on Experimental Safety Vehicles, Paper No. 91-S8-O-11.

Duprey S, Bruyere K, Verriest J-P, 2007. Human shoulder response to side impacts: A finite element study. Computer Methods in Biomechanics and Biomedical Engng. 10(5): 361-370.

Gelaude F, Vander Sloten J, Lauwers B, 2006. Semi-automated segmentation and visualisation of outer bone cortex from medical images. Computer Methods in Biomechanics and Biomedical Engng. 9(1):65-77.

Gonzalez RC, Woods RE, 2002. Digital image processing. 2nd ed. Prentice Hall.

Grosland NM, Brown TD, 2002. A voxel-based formulation for contact finite element analysis. Computer Methods in Biomechanics and Biomedical Engng. 5(1):21-32.

Guldberg RE, Hollister SJ, Charras GT, 1998. The accuracy of digital image-based finite element models. J. Biomechanical Engineering. 120(2):289-295.

Haralick RM, Shanmugam K, Dinstein I, 1973. Textural features for image classification. IEEE Trans. Syst., Man, Cybern. 3(6):610-621.

Horgan TJ, Gilchrist MD, 2003. The creation of three-dimensional finite element models for simulating head impact biomechanics. Int. J. Crashworthiness. 8(4):353-366.

Horgan TJ, Gilchrist MD, 2004. Influence of FE model variability in predicting brain motion and intracranial pressure changes in head impact simulations. Int. J. Crashworthiness. 9(4):401-418.

Kaminsky J, Rodt T, Gharabaghi A, Forster J, Brand G, Samii M, 2005. A universal algorithm for an improved finite element mesh generation Mesh quality assessment in comparison to former automated mesh-generators and an analytic model. Medical Engng & Physics. 27:383-394.

Kang H, Willinger R, Diaw BM, 1997. Validation of a 3D anatomic human head model and replication of head impact in motorcycle accident by finite element modeling. Proceedings of the 41st Stapp Car Crash Conference, pp. 329-338.

Kass M, Witkin A, Terzopoulos D, 1988. Snakes: Active contour models. Int. J. Computer Vision 1(4):321-331.

Keyak JH, Fourkas MG, Meagher JM, Skinner HB, 1993. Validation of an automated method of three-dimensional finite element modelling of bone. J. Biomedical Engineering. 15(6):505-509.

Kleiven S, von Holst H, 2002. Consequences of size following trauma to the human head. J. Biomechanics. 35(2):135-160.

Krabbel G, Muller R, 1996. Development of a finite element model of the head using the visible human data. Proceedings of The Visible Human Project Conference. National Institutes of Health, Bethesda, Maryland.

Kwon GH, Chae SW, Lee KJ, 2003. Automatic generation of tetrahedral meshes from medical images. Computers and Structures. 81(8-11):765-775.

Lie WN, 1995. Automatic target segmentation by locally adaptive image thresholding. IEEE Trans. Image Process. 4(7):1036-1041.

Muller-Hanneman M, 1998. Hexahedral mesh generation by successive dual cycle elimination. Proceedings of the 7th International Meshing Roundtable, Sandia National Laboratories, New Mexico.

Owen SJ, 1998. A survey of unstructured mesh generation technology. Proceedings of the 7th International Meshing Roundtable, Sandia National Laboratories, New Mexico.

Ramos A, Simoes, JA, 2006. Tetrahedral versus hexahedral finite elements in numerical modeling of the proximal femur. Medical Engng & Physics. 28:916-924.

Roylance D, 1996. Mechanics of materials. New York: J. Wiley & Sons.

Tautges TJ, Blacker T, Mitchell SA, 1996. The whisker weaving algorithm: A connectivity-based method for constructing all-hexahedral finite element meshes. Int. J. Numer. Methods in Engineering. 39(19):3327-3349.

Taylor WR, Roland E, Ploeg H, Hertig D, Klabunde R, Warner MD, Ho Ba Tho MC, Rakotomanana L, Clift SE, 2002. Determination of orthotropic bone elastic constants using FEA and modal analysis. J. Biomechanics. 35(6):767-773.

Tesuo A, Naoki K, Takeshi T, 1998. An algorithm for finding an interval with maximum interclass variance and its extension to higher dimensions. Information Processing Society of Japan, Special Interest Group Notes, 065-001.

Verhoeve RSJM, 1998. Computer model of the human head to assess mechanical brain loading in car collisions [dissertation]. [Stan Ackermans Institute, Eindhoven]: The Netherlands.

Vincent L, Soille P, 1991. Watersheds in digital spaces: An efficient algorithm based on immersion simulations. IEEE Trans Pattern Anal. Mach. Intell. 13(6):583-598.

Zhang L, Yang KH, King AI, 2001. Comparison of brain responses between frontal and lateral impacts by finite element modeling. J. Neurotrauma. 18(1):21-30.

FIGURES CAPTIONS

Figure 1 - A sample image during various stages of image processing. (A) shows the original image, an axial slice through the Visible female data set. (B) shows the same image following thresholding. (C) shows the image following initial morphological opening and closing operations to improve connectivity. (D) shows the image following resolution reduction. Plane-merging removes the need to reduce the resolution of the image to such an extreme extent. (E) shows the final image following opening by reconstruction and closing. The large, white area in the region of the sinus is due to the anatomy of the skull in the images immediately above the 2D slice shown here.

Figure 2 - Flow chart showing flow of information through DICER from input of image set to the output of a high quality mesh.

Figure 3 - (A) Two elements that are only joined by an edge. This scenario is modified by adding two elements in the empty locations shown. (B) The central node in this 2x2x2 block is highly exposed. Again, the two empty locations are filled with elements. (C) The front most element has two corners which are unconnected to any other elements. The smoothing algorithm will leave this element highly deformed. Any instances of this kind of element are deleted.

Figure 4 - (A) shows a typical mesh arrangement that following boundary smoothing can lead to highly deformed elements as seen for the labeled elements in (B). (C) shows the same elements after the two planes have been merged into one. (D) shows that following an equivalent smoothing operation there is significantly less element deformation but smoothing results are equal or better to the unmerged case.

Figure 5 - The six unique arrangements of elements surrounding a boundary node that are permitted by the smoothing algorithm. The inverse and rotations of each of these arrangements is also permitted but are considered to be identical by the algorithm.

Figure 6 - High-resolution mesh of a skull containing 135315 elements. B shows the same mesh following one plane-merge in each direction. There is an almost negligible change in the geometry of the mesh yet the element numbers have been reduced to 25489. C shows a mesh without plane-merging but built at a low resolution so the element numbers are very similar to those of mesh B. It can be seen that the level of detail in C is much lower than in B and that the cross section has had to be thickened to prevent losing entire sections of the mesh.

Figure 7 - A cutaway view of a two material mesh based on the Visible Female Dataset. The bone elements are shown in red and the soft tissue elements in blue. The model contains a total of 168543 elements and 180,528 nodes.

Figure 8(a): Semi-infinite plate with a circular hole (not to scale), loaded axially by remote stress, σ_0 and three corresponding FE meshes (b)-(d). The plate needs to be significantly larger relative to the hole size, to simulate an infinite plate. The stress is examined along edge A and circumferentially around the hole between points A and B. Figure 8(b): Patran Mesh Detail; 8(c): Unsmoothed Voxel Mesh Detail; 8(d): Smoothed Voxel Mesh Detail (smoothing coefficient = 0.8).

Figure 9 - Plot of σ_x stress along edge A of the PATRAN mesh (c.f. Figure 8(a)). The predicted stress increases from σ_0 remote from the hole to $3\sigma_o$ at the edge of the hole. The theoretical values range from σ_0 to $3\sigma_0$. As such, the PATRAN results agree closely with the theoretical result.

Figure 10 - Plot of σ_x-stress along the edge of the unsmooth mesh (see Figure 8(c)). The stress increases to $2.3\sigma_0$ at the edge of the hole. Note that the results are essentially identical to those predicted by the PATRAN mesh from the edge of the plate to within one element from the edge of the hole.

Figure 11 - Variation in peak axial stress as a function of the smoothing coefficient. The plot shows a clear improvement as the coefficient increases; the stress approaches the true value of $3\sigma_0$. Above a smoothing coefficient of 0.8, however, the result rapidly deteriorates.

Figure 12 - Variation of axial stress circumferentially around rim of hole as predicted using the unsmoothed voxel mesh of Figure 8(c). The error is greatest at nodes 8, 11 and 15.

Figure 13 - Variation in axial stress circumferentially around rim of hole predicted using the smoothed voxel mesh of Figure 8(d) with a smoothing coefficient of 0.8. Improvements at nodes 11 and 15 have yielded a significant improvement in the overall stress distribution.

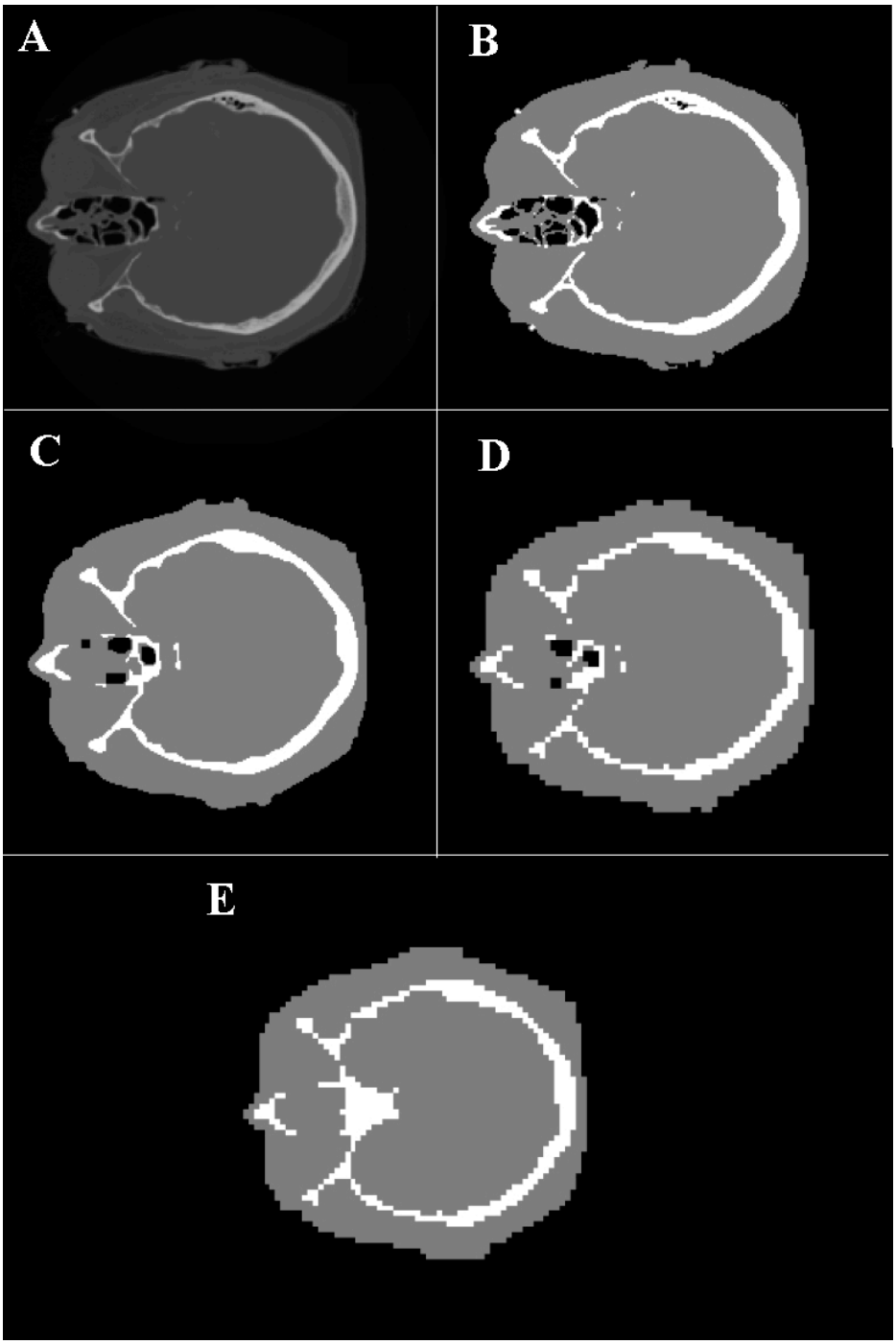


Figure 1 - A sample image during various stages of image processing. (A) shows the original image, an axial slice through the Visible female data set. (B) shows the same image following thresholding. (C) shows the image following initial morphological opening and closing operations to improve connectivity. (D) shows the image following resolution reduction. Plane-merging removes the need to reduce the resolution of the image to such an extreme extent. (E) shows the final image following opening by reconstruction and closing. The large, white area in the region of the sinus is due to the anatomy of the skull in the images immediately above the 2D slice shown here.

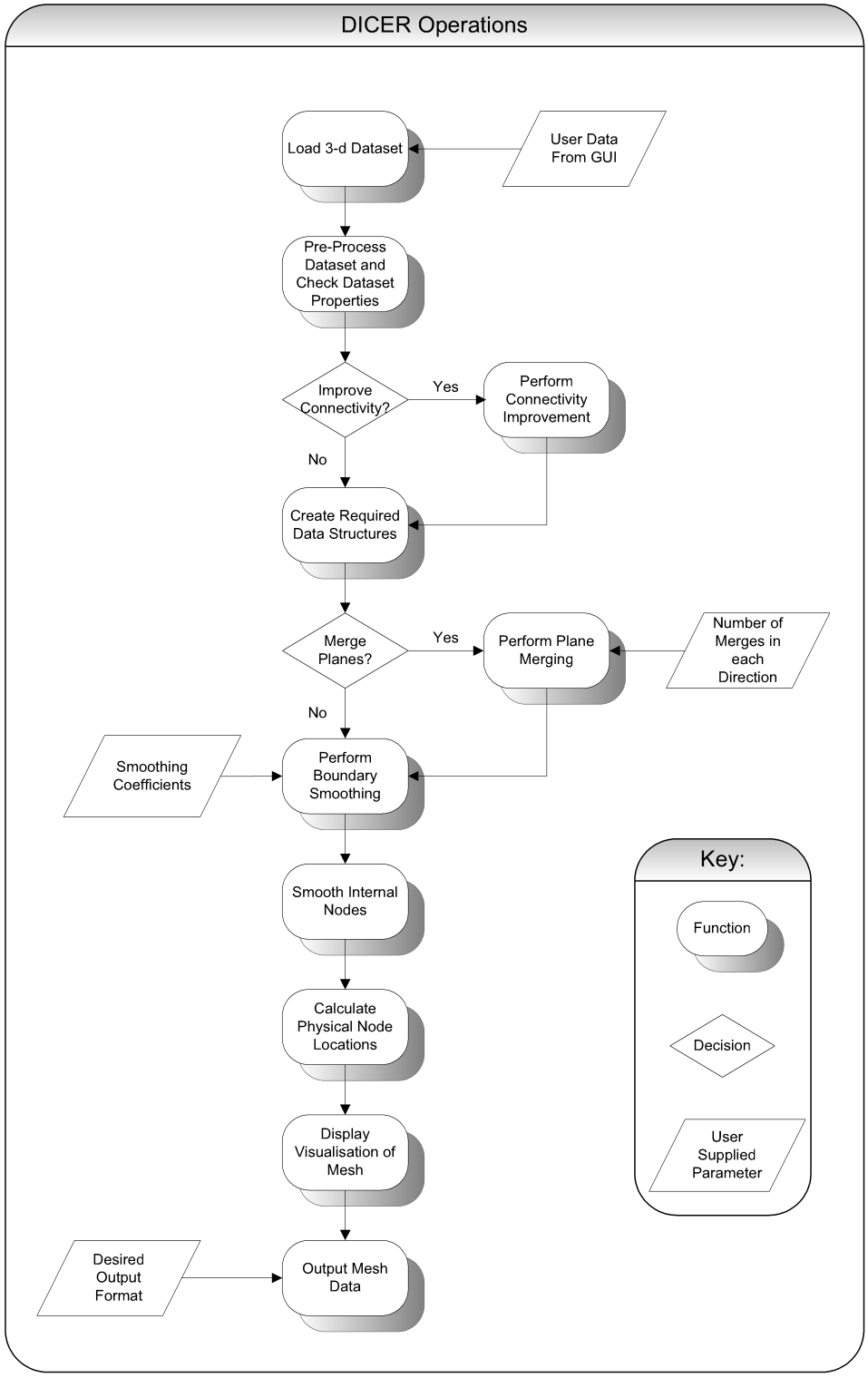


Figure 2 - Flow chart showing flow of information through DICER from input of image set to the output of a high quality mesh.

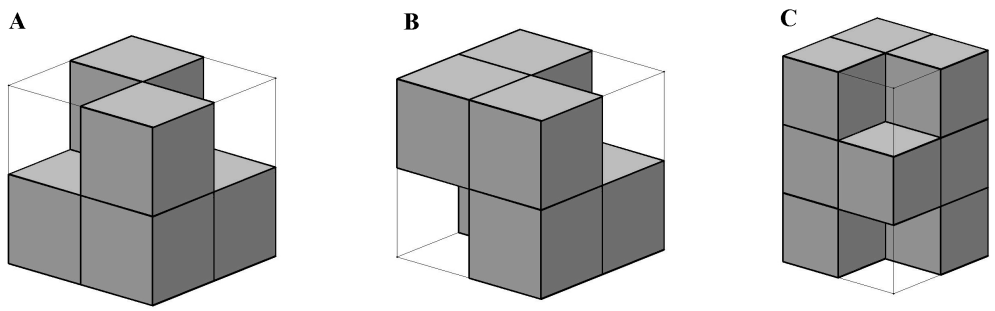


Figure 3 - (A) Two elements that are only joined by an edge. This scenario is modified by adding two elements in the empty locations shown. (B) The central node in this 2x2x2 block is highly exposed. Again, the two empty locations are filled with elements. (C) The front most element has two corners which are unconnected to any other elements. The smoothing algorithm will leave this element highly deformed. Any instances of this kind of element are deleted.

A B

C D

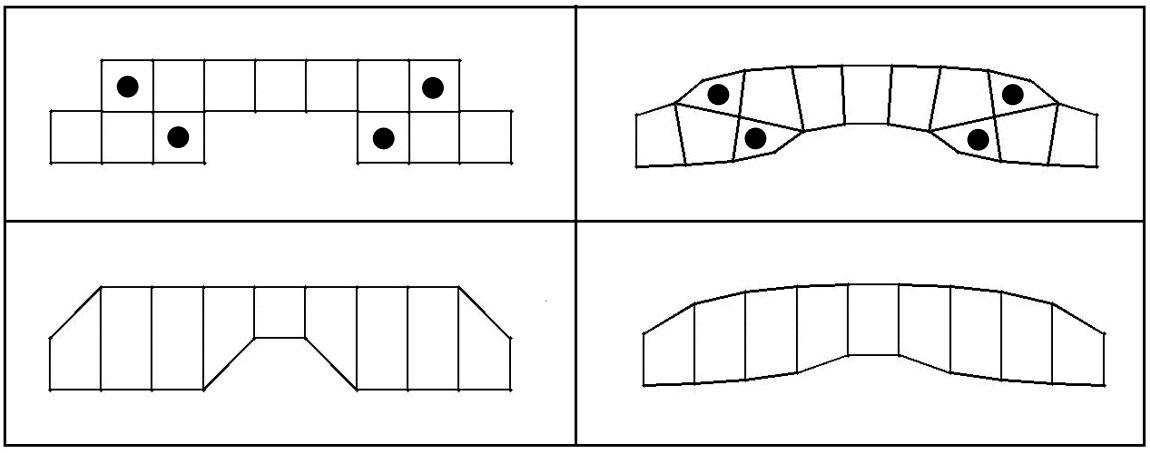


Figure 4 - (A) shows a typical mesh arrangement that following boundary smoothing can lead to highly deformed elements as seen for the labeled elements in (B). (C) shows the same elements after the two planes have been merged into one. (D) shows that following an equivalent smoothing operation there is significantly less element deformation but smoothing results are equal or better to the unmerged case.

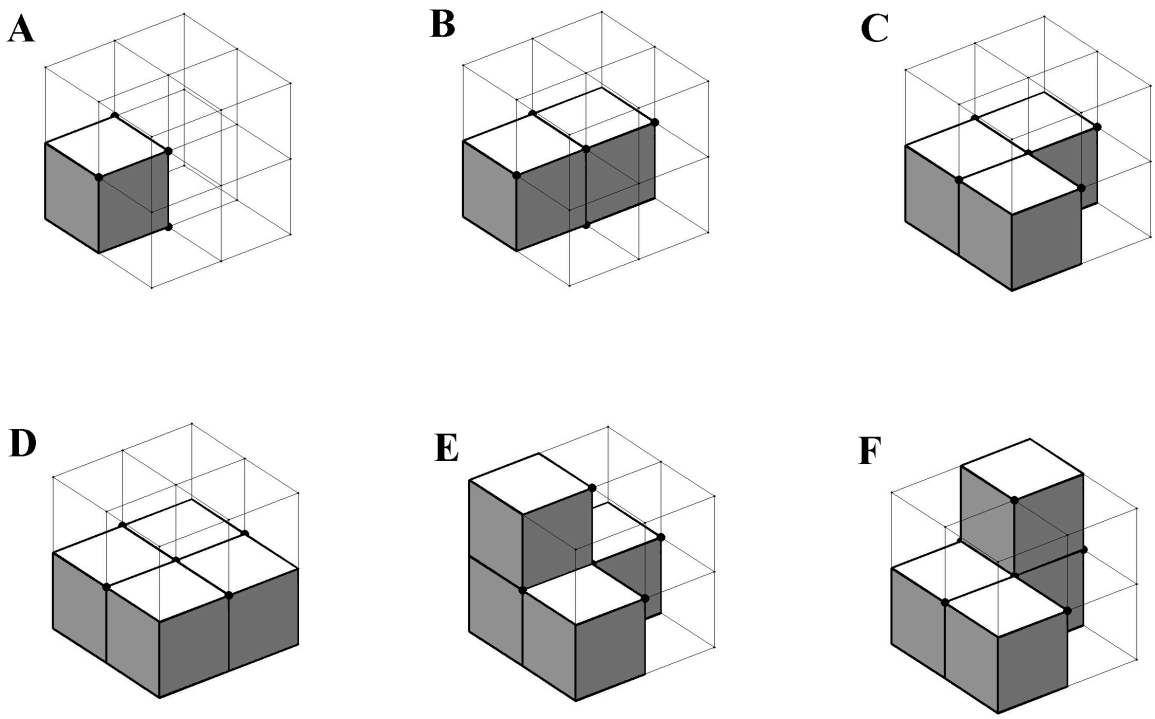
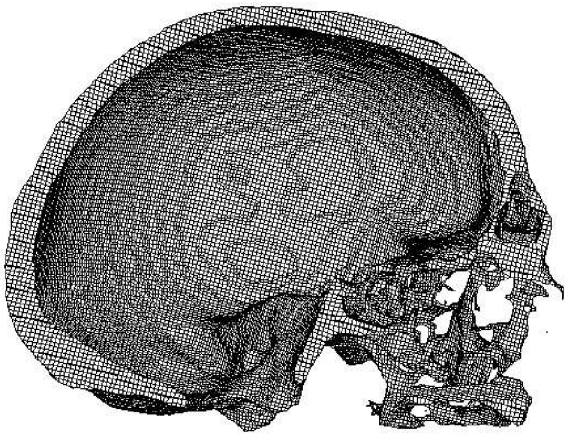
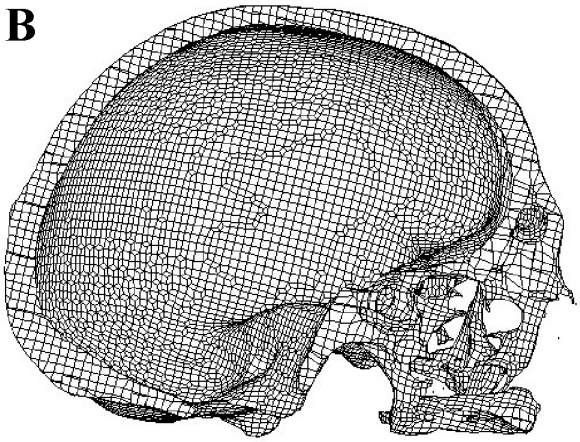


Figure 5 - The six unique arrangements of elements surrounding a boundary node that are permitted by the smoothing algorithm. The inverse and rotations of each of these arrangements is also permitted but are considered to be identical by the algorithm.

A



B



C

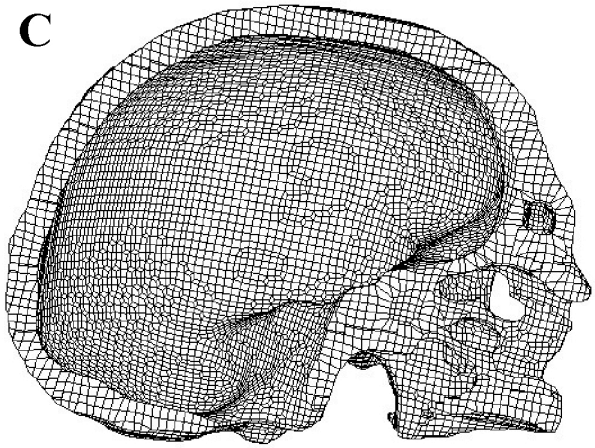


Figure 6 - High-resolution mesh of a skull containing 135315 elements. B shows the same mesh following one plane-merge in each direction. There is an almost negligible change in the geometry of the mesh yet the element numbers have been reduced to 25489. C shows a mesh without plane-merging but built at a low resolution so the element numbers are very similar to those of mesh B. It can be seen that the level of detail in C is much lower than in B and that the cross section has had to be thickened to prevent losing entire sections of the mesh.

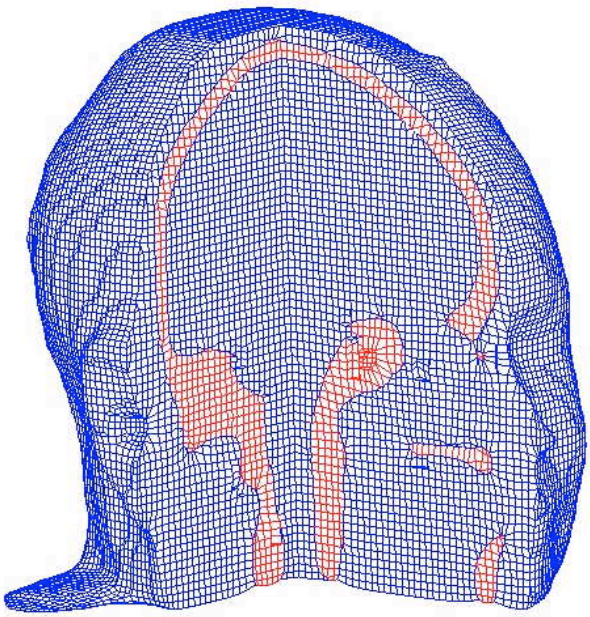


Figure 7 - A cutaway view of a two material mesh based on the Visible Female Dataset. The bone elements are shown in red and the soft tissue elements in blue. The model contains a total of 168543 elements and 180528 nodes.

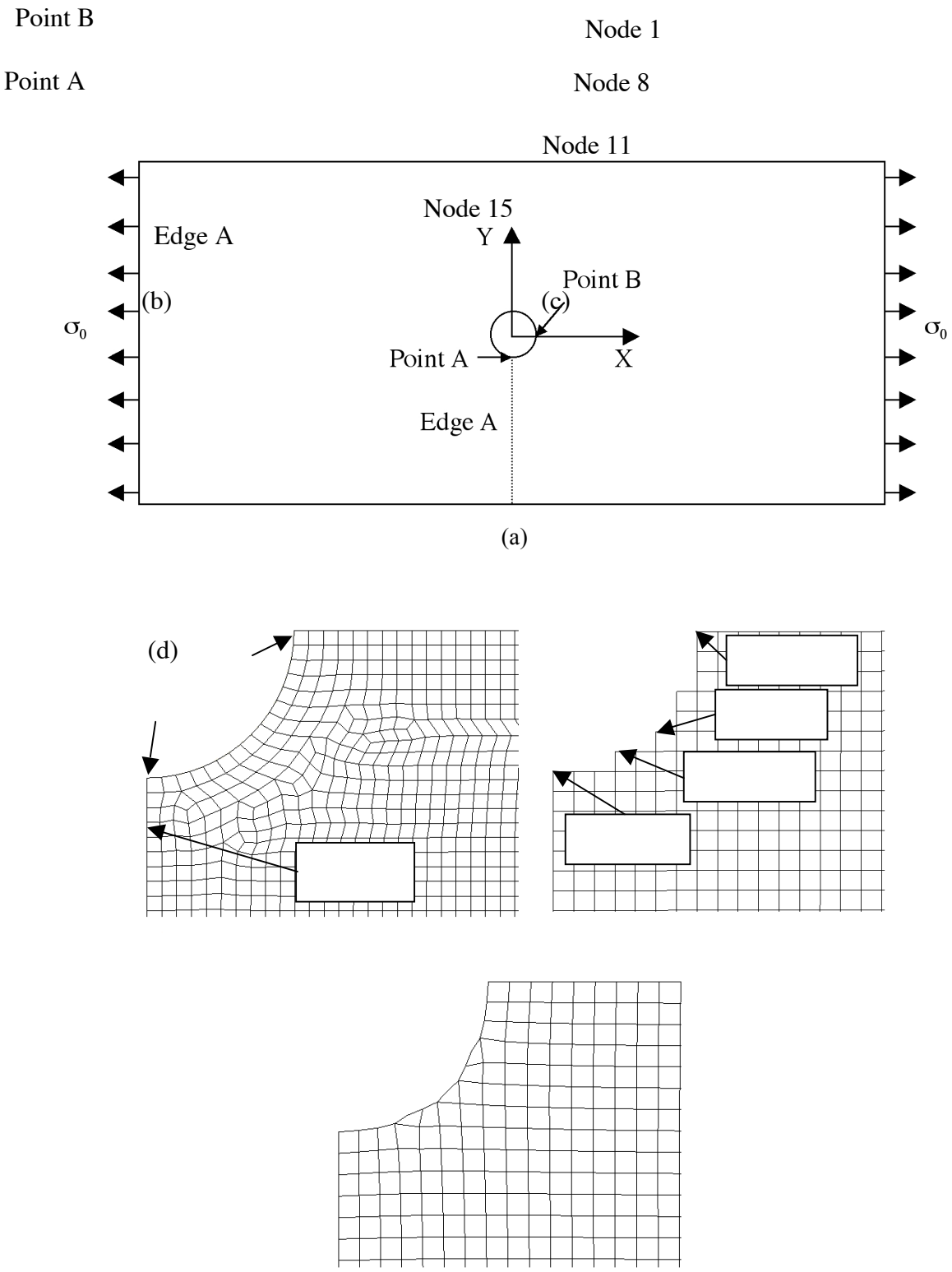


Figure 8(a): Semi-infinite plate with a circular hole (not to scale), loaded axially by remote stress, σ_0 and three corresponding FE meshes (b)-(d). The plate needs to be significantly larger relative to the hole size, to simulate an infinite plate. The stress is examined along edge A and circumferentially around the hole between points A and B. Figure 8(b): Patran Mesh Detail; 8(c): Unsmoothed Voxel Mesh Detail; 8(d): Smoothed Voxel Mesh Detail (smoothing coefficient = 0.8).

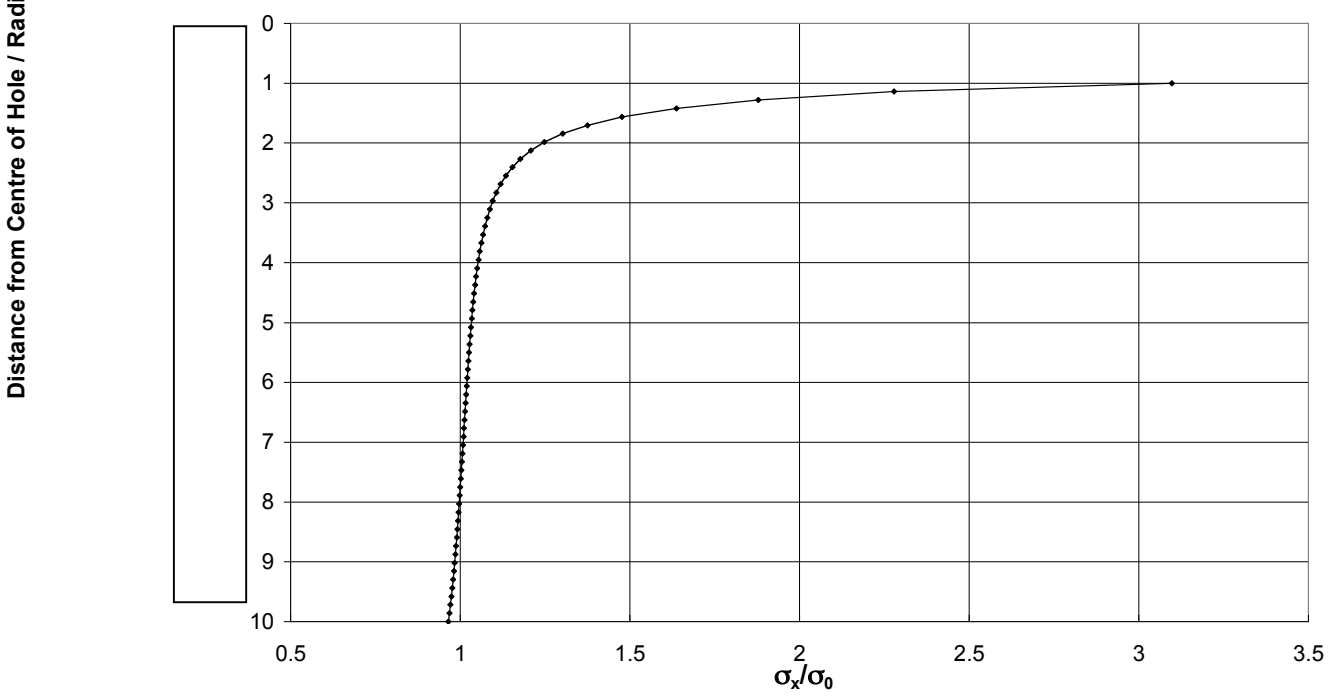


Figure 9 - Plot of σ_x stress along edge A of the PATRAN mesh (c.f. Figure 8(a)). The predicted stress increases from σ_0 remote from the hole to $3\sigma_o$ at the edge of the hole. The theoretical values range from σ_0 to $3\sigma_0$. As such, the PATRAN results agree closely with the theoretical result.

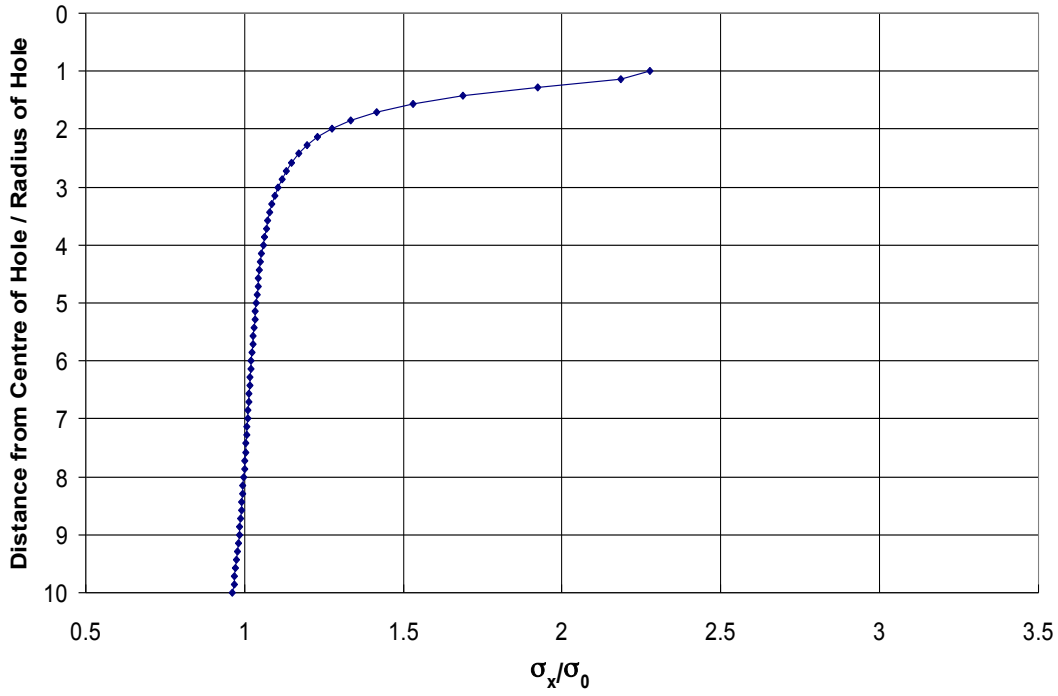


Figure 10 - Plot of σ_x-stress along the edge of the unsmooth mesh (see Figure 8(c)). The stress increases to $2.3\sigma_0$ at the edge of the hole. Note that the results are essentially identical to those predicted by the PATRAN mesh from the edge of the plate to within one element from the edge of the hole.

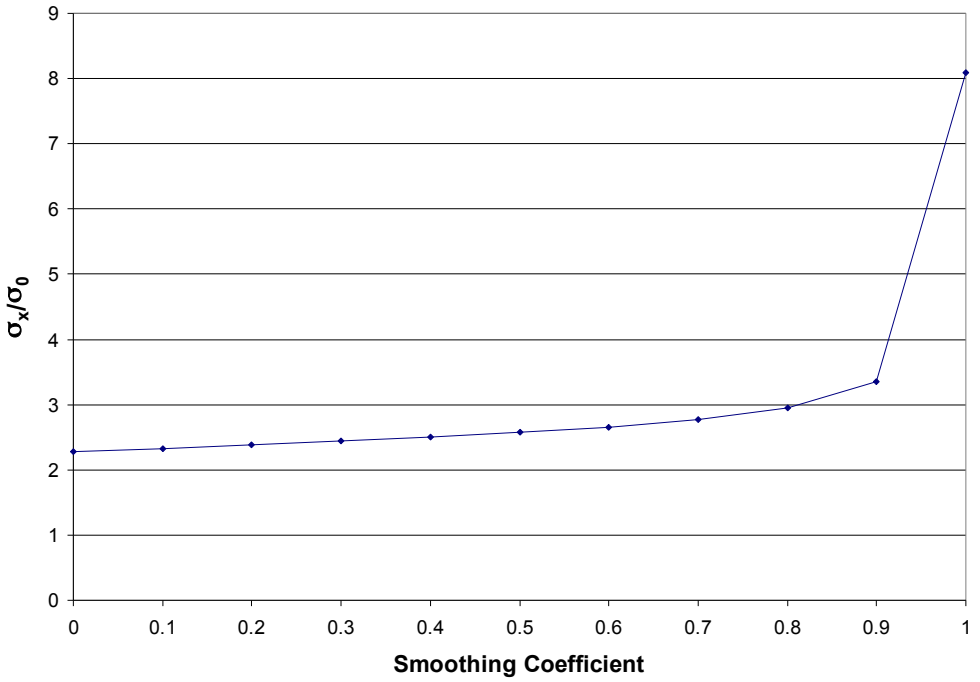


Figure 11 - Variation in peak axial stress as a function of the smoothing coefficient. The plot shows a clear improvement as the coefficient increases; the stress approaches the true value of $3\sigma_0$. Above a smoothing coefficient of 0.8, however, the result rapidly deteriorates.

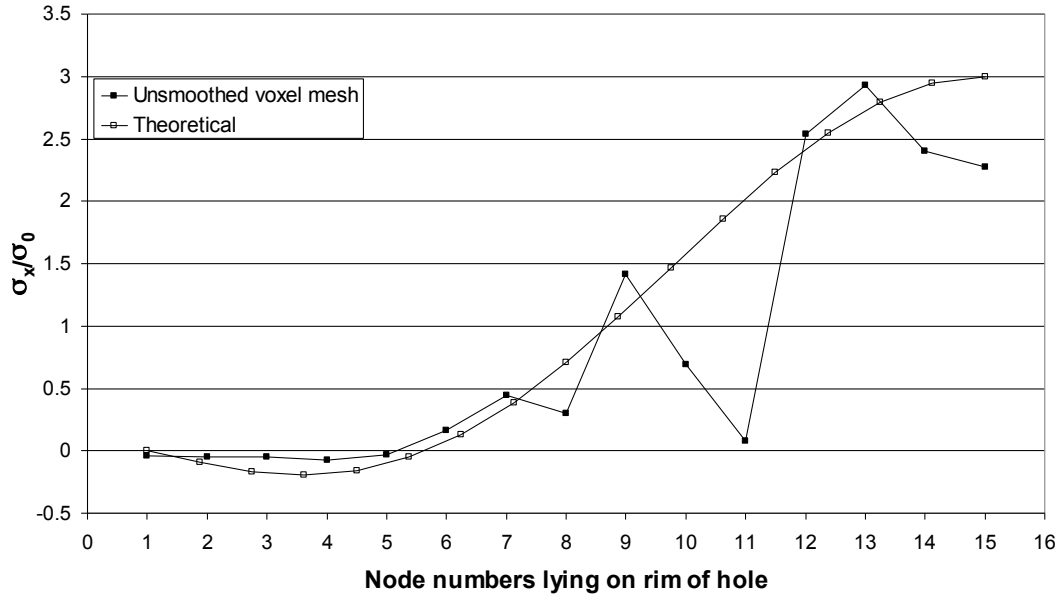


Figure 12 - Variation of axial stress circumferentially around rim of hole as predicted using the unsmoothed voxel mesh of Figure 8(c). The error is greatest at nodes 8, 11 and 15.

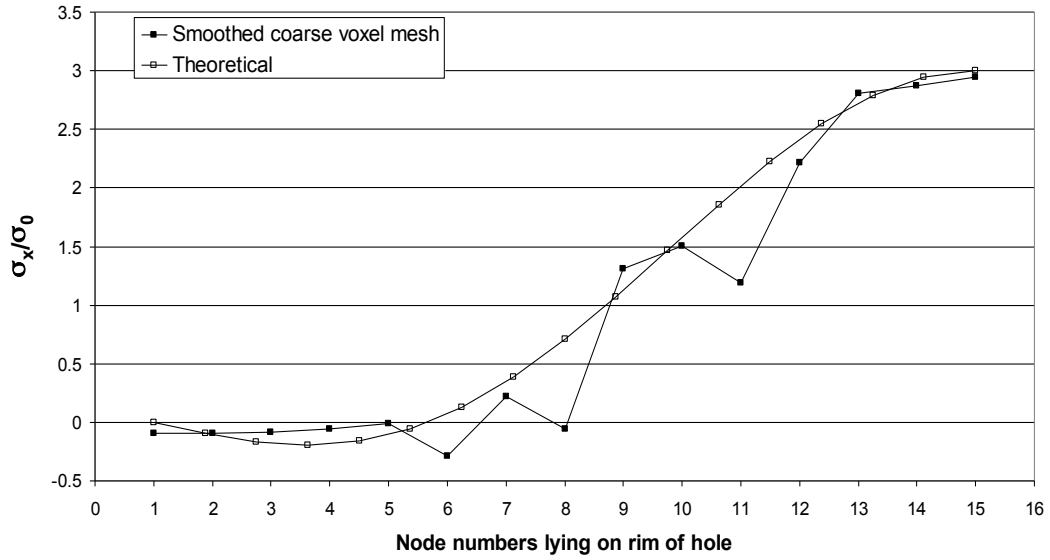


Figure 13 - Variation in axial stress circumferentially around rim of hole predicted using the smoothed voxel mesh of Figure 8(d) with a smoothing coefficient of 0.8. Improvements at nodes 11 and 15 have yielded a significant improvement in the overall stress distribution.

BOX 1

Type A mesh connectivity improvement

1. This function is separately called for each different material class in the mesh. During each function call, all elements of other material classes are ignored. All voxels of the specific class are given a value of 1 and all others are set to 0, creating a binary dataset. It is assumed that there is no preferential order in which to apply this algorithm to the different material classes.
2. Search 2x2 block of voxels on each 2D plane in the voxel-set for instances of pairs of voxels of value 1 which are only diagonally connected, i.e., they are arranged diagonally in the 2x2 block and the two remaining voxels are of value 0. This can be rapidly performed using the `bwmorph.m` function in MATLAB, which convolves the image with a specific filter and compares the results to a lookup table determined by the arrangement of voxels which are to be modified.
3. Wherever an instance of diagonal connection is found, the `bwmorph.m` function adds in a voxel so the voxels are connected by an edge rather than just by a vertex.
4. If the mesh is 3D, repeat these steps on both of the other sets of 2D planes.
5. For all voxels in the original dataset that have a voxel value of 1 in the binary dataset, set their value to that of the material classification being considered in the current function call.

Type B mesh connectivity improvement

1. This function is separately called for each different material class in the mesh. During each function call, all elements of other material classes are ignored. All voxels of the specific class are given a value of 1 and all others are set to 0, creating a binary dataset. It is assumed that there is no preferential order in which to apply this algorithm to the different material classes.
2. Due to the arrangement of elements to be corrected, this algorithm is only necessary for 3D datasets.
3. For every possible 2x2x2 block of voxels in the dataset, sum the values of all 8 voxels. Do this by creating index vectors that reference the 7 relevant voxels relative to each voxel in the dataset. This is accomplished rapidly by generating a vector [n, 1, 2, 3, …] for a dataset of size nxnxn. These numbers index the last, first, second, third, etc. voxels along a given dimension. This uses toroidal boundary conditions and indexes the last voxel in a given direction as preceding the first, the first voxel as preceding the second voxel and so on. Using combinations of this index vector in all three voxel indices rapidly searches all possible 2x2x2 blocks.
4. Find all instances where the sum calculated in the previous step is six, implying six voxels of value 1 in the block.
5. Assemble an array (NEIGHBOURS) where each row corresponds to an instance of a six-voxel block and each column holds the voxel value for one of the eight voxels in that 2x2x2 block.
6. Depending on the order of the columns in NEIGHBOURS, there exist four pairs of columns that correspond to voxels that are arranged diagonally in the block.
7. For each row of NEIGHBOURS, find the indices of the two zero voxels and compare these to the four diagonal voxel pairs. Wherever there is a match, an instance has been found of the voxel arrangement to be corrected: store an index to this 2x2x2 block in an array (CORNERTOCORNERS).
8. Return to the original dataset and set all voxels that are indexed in CORNERTOCORNERS to the value of the material class being operated on.

Type C mesh connectivity improvement

1. This algorithm searches for and replaces elements that have two nodes that are connected to each other and that both have just three boundary neighbours. Nodes of this type will cause the element they form to become highly distorted following boundary smoothing. Due to the nature of the element configuration, this algorithm is only necessary for 3D meshes.
2. Follow the boundary smoothing algorithm (Box 3) up to step 5.
3. Set all column entries of NODEADJ corresponding to nodes with three boundary neighbours equal to zero.
4. Compare the sum of the rows of NODEADJ before and after this operation. For those nodes with three boundary neighbours that have altered values of this row sum, an instance of the configuration to be removed has been found.
5. Find every element that contains one of these nodes from the element list.
6. For each of these elements, find which material class to which the majority of its neighbouring elements belong.
7. In the segmented voxel set, change the material class of each of the elements to this majority classification.

BOX 2

Plane Merge Algorithm

1. User specifies the number of plane merges to perform in each direction (SI/AP/LR). The merges in different directions are performed separately but all follow the steps described below. If more than two planes are to be merged into one plane, begin by merging the top two planes in the group to be merged. Then merge the newly formed plane of elements with the next plane and so on.
2. Extract node locations (NODE) from master data structure.
3. Divide nodes into groups based on which plane perpendicular to the merge direction they lie on (NODEPLANE). This division groups nodes with the same index in the merge direction. The index is found from NODE.
4. Generate a list of planes from NODEPLANE that must be deleted (DELETEPLANES). Node planes to be deleted are defined by each pair of adjacent element planes that are to be merged. From this, generate a list of all nodes lying on planes to be deleted (POSSIBLES).
5. Find all edges parallel to the merge direction (VERTEDGES). Each edge is defined by the two nodes it connects. The node closest to the origin is listed first, thereby associating a "top" and "bottom" with each edge.
6. Search each edge in VERTEDGES to find nodes at the top of one edge and the bottom of another. Take the intersection of this list with POSSIBLES. This list (DELETE) contains all nodes that will be deleted.
7. The set difference of DELETE with POSSIBLES yields those nodes that must be retained in DELETEPLANES. For these nodes, determine whether they lie at the top or bottom of an edge. This is necessary so these nodes can be assigned to the correct plane for subsequent merges, as the index in the merge direction is not sufficient to specify which plane a node belongs to once merges have occurred.
8. For each node in DELETE find the nodes by which it can be replaced. These are the nodes it shares an edge with in VERTEDGES. This list of possible replacements is called REPLACE.
9. Generate new reference numbers for all nodes that will be retained.
10. Replace all nodes in delete with the correct replacement in REPLACE. The correct replacement is found by checking whether the node to be deleted is at the top or bottom of an edge. When it is at the top of an edge, replace with the node with which it forms the bottom of an edge and vice versa. This creates two duplicate elements that are the sum of two adjacent elements in the planes to be merged.
11. Extract the unique elements to form a trial element list (TRIALELEM). From this, form a trial nodal connectivity matrix (TRIALNODEADJ).
12. Use TRIALNODEADJ to check for over-connected nodes. These are nodes connected to more than six other nodes.
13. If no over-connected nodes exist, TRIALNODEADJ and TRIALELEM become NODEADJ and ELEM. Update all other data structures and end.
14. If over-connected nodes exist, find which planes they lie on and remove those plane indices from DELETEPLANES. Return to step 4 and continue until no over-connected nodes exist.

BOX 3

Boundary Smoothing Algorithm

1. Extract node locations (NODE), nodal connectivities (NODEADJ) and nodal classifications (BOUNDARYNODES) from master data structure.
2. User specifies smoothing coefficients. From these, calculate the smoothing coefficient appropriate for each class of node (see text).
3. Check if mesh is 2D or 3D and enter corresponding section of code. Both function similarly.
4. Remove all nodes except BOUNDARYNODES from NODEADJ, leaving a 2D graph (or 1D if the mesh is 2D) of boundary nodes.
5. Classify boundary nodes by the number of boundary edges they form. This forms lists (THREES, FOURS, FIVES & SIXES). In a 2D mesh all nodes form two boundary edges.
6. Further classify those with four edges into groups where the node is shared by two or four elements. This classifies the nodes as being either class B or D nodes (see text). In a 2D mesh there are just two corresponding classes.
7. Modify NODEADJ to become matrix A in the boundary smoothing equation. Multiply all entries on the rows corresponding to THREES by –C/3, those corresponding to FOURS by –C/4 and so on, where C is the smoothing coefficient for nodes with the given number of boundary edges. Then set all main diagonal entries to 1. For 2D meshes all rows should be multiplied by –C/2.
8. Modify the columns of NODE to become the vector B in the boundary smoothing equation. Multiply entries of NODE corresponding to THREES by (1-C). Repeat for rows corresponding to the nodes indexed by FOURS, FIVES & SIXES.
9. Obtain smoothed positions by solving Ax=b for each dimension of the dataset, where x_i are the new locations of the nodes in the i^{th} dimension.
10. Update NODE and pass back to the master structure and end.