PERGAMON

Computers & Graphics 23 (1999) 583–598

COMPUTERS & GRAPHICS

www.elsevier.com/locate/cag

Technical Section

# Regularised marching tetrahedra: improved iso-surface extraction

G.M. Treece[a,*], R.W. Prager[a], A.H. Gee[a]

[a]*Engineering Department, Cambridge University, Trumpington Street, Cambridge, CB2 1PZ, UK*

**Abstract**

*Marching cubes* is a simple and popular method for extracting iso-surfaces from implicit functions or discrete three-dimensional (3-D) data. However, it does not guarantee the surface to be topologically consistent with the data, and it creates triangulations which contain many triangles of poor aspect ratio. *Marching tetrahedra* is a variation of marching cubes, which overcomes this topological problem, but further degrades the triangle aspect ratios. Improvement in triangle aspect ratio has generally been achieved by *mesh simplification*, a group of algorithms designed mainly to reduce the triangle count. *Vertex clustering* is one of the simplest, but does not necessarily maintain the topology of the original mesh. We present a new algorithm, *regularised marching tetrahedra* (RMT), which combines marching tetrahedra and vertex clustering to generate iso-surfaces which are topologically consistent with the data and contain a number of triangles appropriate to the sampling resolution (typically 70% fewer than marching tetrahedra) with significantly improved aspect ratios. This improvement in aspect ratio greatly enhances smooth shaded displays of the surface. Surface triangulations are shown for implicit surfaces, thresholded medical data, and surfaces created from object cross-sections — implementations of RMT appropriate to each of these situations are available.[1] The application to data sampled on non-parallel planes is also considered. 

*Keywords:* Iso-surface extraction; Marching cubes; Marching tetrahedra; Vertex clustering; Mesh simplification

## 1. Introduction

Extracting and displaying iso-surfaces of volume data is useful in many areas. Three-dimensional (3-D) data is notoriously difficult to visualise, and rendering a surface within this data is one of the main tools (though by no means the only one) available for this purpose. In addition to visualisation, an extracted surface can also be used for measurements such as volume or surface area.

The most common method for describing a generic surface is with connected polygons, triangles being the simplest form. Modern graphics hardware can render such polygonal surfaces very efficiently, and hence most iso-surface extraction techniques have concentrated on this representation. Gouraud or Phong shading can be used to improve the apparent smoothness of the surface by interpolating values estimated at the triangle vertices [4].

A simple method for constructing such *iso*-surfaces, *marching cubes* (MC) [14], was introduced for 3-D medical data. The basic principle is to reduce the problem to that of triangulating a single cube, which is intersected by the surface. Surface intersection points are defined along the edges of the cube, by linear interpolation of the sampled data at each corner. These vertices are joined by between one and five triangles to form a polygonised surface patch. The whole iso-surface can be triangulated by 'marching' this cube through the data and creating surface patches whenever the cube intersects the surface. There are 256 different cases for the triangulation of a cube, based on whether sampled data at the corners is 'inside' or 'outside' the surface; however this can be reduced to only 15 by symmetry. The triangulations for

*Corresponding author. Tel.: +44-1223-332754; fax: +44-1223-332662.

*E-mail address:* gmt11@eng.cam.ac.uk (G.M. Treece)

[1] http://svr-www.eng.cam.ac.uk/ ~ gmt11/software.html.

0097-8493/99/$ - see front matter 
PII: S0097-8493(99)00076-X

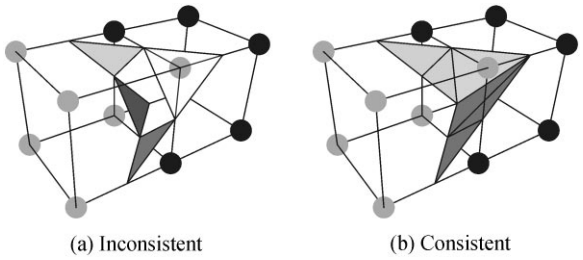


Fig. 1. Marching cubes. (a) The use of complementary symmetry can generate topological inconsistencies. (b) This can be corrected by use of an alternative triangulation for the complementary case. Grey vertices indicate data on one side of the surface, and black on the other.

these cases are stored in a look up table — it is only necessary to compute the symmetry and the case for each cube.

There are two well-known problems with this otherwise very simple and attractive algorithm, connected with the *topology* and the *regularity* of the eventual triangulation.

The original MC algorithm does not produce surfaces which are topologically consistent with the original data. This is due to the arbitrary choice of triangulation for a cube with any face containing two opposite corners which are on one side of the iso-surface, and two which are on the other. If the same triangulation is chosen for both the cubes containing this face, then the surface will contain holes, as shown in Fig. 1. Many methods have been proposed to correct this situation [18,26]. The simplest of these methods employs a different cube triangulation for complementary symmetries of ambiguous faces [15], which results in a bias in the topology towards one side of the surface. More complex methods use more sophisticated interpolation to determine the cube triangulation [17], or incorporate data from outside the cube to be triangulated.

This ambiguity can be removed entirely by tessellating space with tetrahedra rather than cubes. Tetrahedra only have 16 possible triangulations, which reduce to 3 by symmetry. These are shown (save for the case where no triangulation is required) in Fig. 2. A tetrahedral tessellation was originally constructed by dividing each cube into five tetrahedra [7,20]. Unfortunately, this introduces an additional ambiguity since the symmetry of the subdivision of the cube has to alternate between cubes, in order to align the faces of the tetrahedra. There are, therefore, two possible tessellations for a given cubic lattice, which can generate opposed topologies. This ambiguity can only be resolved by using cubic, rather than linear, interpolation, thereby allowing the iso-surface to intersect the tetrahedral edges more than once [27]. However, this is much more complex, and the resulting look up table has 59 cases — many more than the original MC method.

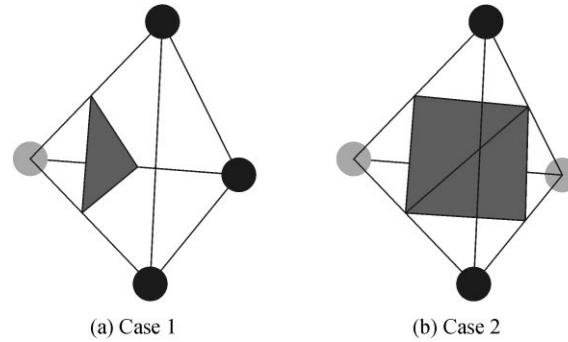


Fig. 2. Marching tetrahedra. The two cases where triangulation is required, and the resulting triangulations. (b) Case 2 can be rendered as a quadrilateral rather than two triangles, since the surface is guaranteed to be planar.

Recently, Chan and Purisma have suggested a tessellation of space into tetrahedra based on a body-centred cubic lattice [2], which we will call *marching tetrahedra* (MT). The lattice can also be formed from a higher-resolution cubic lattice. However, in contrast to the subdivided cube method, the resulting tetrahedra are all identical, and there is no ambiguity in the tessellation. The tetrahedra are also more regular than those formed by subdividing a cube. The resulting triangulation is more uniform, and has fewer of the sharp edges associated with tetrahedral decompositions. Using a body-centred cubic lattice also results in better sampling efficiency compared with the cubic lattice [2]. The main disadvantage of tetrahedral schemes is that they create an even larger number of triangles than MC for a given data set. This aggravates the second problem with MC, that of the *regularity* of the resulting surface triangulation.

The MC algorithm creates on average three triangles per cube, more if different complementary case triangulations are used. In addition, where the cubic lattice just intersects the surface, these triangles can be arbitrarily small or thin. In typical medical applications this often results in more than 500,000 triangles — which leads to substantial storage and rendering requirements. There is a large body of literature on methods of post-processing triangulated surfaces to reduce this number of triangles whilst maintaining the 'quality' of the surface. These *mesh simplification* methods have recently been reviewed in [3].

The driving force behind much of the work on mesh simplification is the reduction of storage and rendering times, often with complex virtual reality scenes in mind. However, badly proportioned triangles can also result in poor quality renderings when any form of interpolated shading is used to make the surface appear smooth. This is particularly the case for Gouraud shading, which is the simplest form and has the most hardware support. Specular reflections using this technique highlight the

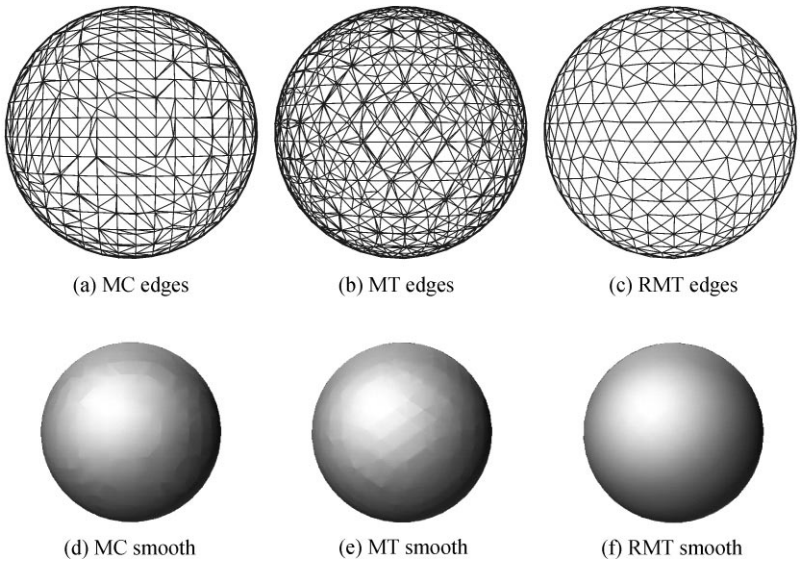


Fig. 3. Effect of triangle condition on interpolated shading. The triangles generated by *marching cubes* (MC) and *marching tetrahedra* (MT) cause sharp edges and points which are no longer apparent when using *regularised marching tetrahedra* (RMT).

original mesh — very thin triangles tend to appear as sharp edges and very small triangles as sharp points. Fig. 3 demonstrates this effect for a sphere — (a) and (d) show the iso-surface formed using MC and (b) and (e) show that formed using MT. In addition, mesh simplification algorithms do not take advantage of the regular way in which the iso-surface is created, since they are mostly designed to operate on arbitrary (although usually closed and oriented) triangle meshes.

### *1.1. Regularised marching tetrahedra*

The main idea of our approach is to combine the two fastest and most simple algorithms, MT for iso-surface extraction, and vertex clustering for mesh simplification, using each to improve the performance of the other. Vertex clustering can be used to produce a regularised triangle set from MT, and basing the clustering around the original tetrahedral lattice allows the preservation of the original topology, which is otherwise difficult to achieve in clustering methods. The result is a triangulated surface consisting of near regular triangles which can be generated in nearly the same time as a surface using MT alone, as in Fig. 3(c) and (f).

To summarise, the RMT algorithm presents the following benefits:

- Simplification using vertex clustering is performed *before* triangulation, taking advantage of the original sampled data.
- Unlike other vertex clustering approaches, the original topology is preserved.
- The additional time required for clustering is offset by the reduction in points and triangles stored, resulting in a fast algorithm.
- Triangle condition is dramatically improved, which results in superior interpolated shading.
- Triangle count is typically reduced by 70% compared to that obtained from MT, and by 40% compared to MC.
- A method is presented to estimate gradients and curvature for a given tetrahedral edge (and hence surface vertex), using only the data from the vertices of the tetrahedra containing that edge.

## 2. Related work

### *2.1. Iso-surface extraction*

A new marching method for triangulating an iso-surface is presented in [9]. Rather than marching through a volume of data, the surface is gradually triangulated by progressing a polygon front across it. The method is more suited to implicit surfaces, since a nearest point calculation is required at each stage. However, the eventual triangulation is very similar to that produced by the RMT method, since a hexagonal grid is used to form the basis for the marching front.

Iso-surface extraction has applications in three different (though overlapping) areas:

*Implicit surfaces:* These are surfaces which can be defined by a mathematical function. As has been noted in [9], this can also include discrete surfaces so long as it is

possible to *implicitise* them with an appropriate function.

*Thresholded data:* Particularly in medical applications, displaying the surface of a thresholded data set can be a useful tool for understanding the anatomy.

*Surface from cross-sections:* Rather than extracting an iso-surface from data directly, a distance function can be created which represents some form of distance to an object cross-section. The iso-surface of this function often gives better results than that of the original function. Shape-based interpolation [10], maximal disc guided interpolation [24], 3-D distance transforms [21], offset surfaces [20] and surface reconstruction from unorganised points [11] are all examples of this.

A data threshold can be used to create the cross-sections in the latter case. Equally, the display of thresholded surfaces can be improved by iso-surfacing a distance function derived from the threshold. However, thresholded data often has much more complex topology, particularly in medical applications, hence the distinction.

### 2.2. Mesh simplification

The aim of mesh simplification is to reduce the number of polygons in a given mesh whilst retaining the main features. The methods are generally iterative, and many are suitable for generating meshes of various resolutions, which can then be used for efficient rendering at different distances. There are many algorithms available for this purpose, recently reviewed in [3]. One of the simplest and fastest groups is *vertex clustering*. Here vertices in some region are merged to form one new vertex, and polygons containing more than one of the merged vertices can then be removed [1,23]. *Edge collapsing*, where an edge is reduced to zero length and two triangles removed, can be seen as a subset of *vertex clustering*, in which only two vertices are clustered.

The main differences between these methods are how they are iterated, and what cost function is used to drive the vertex clustering and determine whether clustering is allowed in any one case. A fairly sophisticated example is *quadric error metrics* [5], which calculates an error for each vertex based on the sum of squared distances from the planes that originally contained that vertex.

We are only aware of two published instances where the ideas of vertex clustering are applied during, rather than after, iso-surface construction. Both of these employ tetrahedral lattices. In [8], vertices which are near to the corners of the tetrahedra (within 5% of the tetrahedra edge length) are 'snapped' to the tetrahedra corner locations, thus eliminating any triangles with edges shorter than this distance. In [6,7], *surface perturbation* is optionally allowed, whereby positive values sampled at the tetrahedral vertices are set to zero, thus ensuring all new vertices are positioned at the tetrahedral vertices, and the triangulation consists entirely of tetrahedral faces.

### 2.3. Surface from cross-sections

Iso-surface extraction methods in combination with some form of distance function calculation can be seen as a solution to the 'surface from cross-sections' problem. Traditionally, this has been approached by placing vertices on each cross-section, then triangulating pairs of cross-sections by joining these vertices. The problem is complicated particularly when the cross-sections are highly concave, or when there are a different number of contours on each plane. Recently, approaches which estimate intermediate cross-sections, in order to simplify these cases, have been proposed [19]. This can be seen as one step towards estimating the *whole* intermediate surface by some function, and then using an iso-surface extraction method to triangulate this estimated surface. This sort of approach (albeit using surface point display methods rather than a triangulation) has been used in [13,16], as well as in [24].

## 3. Algorithm

The terminology adopted in this description is as follows:

*Sample points:* The corners of the tetrahedral lattice, at which the data is sampled.

*Edge:* A line joining two sample points contained by one tetrahedron.

*Surface intersection:* The point of intersection of the iso-surface with an edge, calculated by linear interpolation of the sample points at each end of the edge.

*Vertices:* The points used in the eventual triangulation of the surface.

The first step is to construct a suitable lattice to define the sampling of the data. As in [2], a body-centred cubic lattice is used — this gives better sampling resolution than the conventional cubic lattice, and can be subdivided into a set of identical and nearly regular tetrahedra. Each of these tetrahedra have one pair of opposite edges of 2 units and two pairs of $\sqrt{3}$ units. This lattice is shown in Fig. 4(a), along with the cube on which it is based (rendered semi-transparent to show the hidden edges). Sample points are connected to the nearest neighbours to form four tetrahedra per cube face.

In [2], the planes containing the sample points are aligned with the faces of the cube, as in Fig. 4(b). The orientation can also be chosen such that the planes lie along the diagonal of two cube faces, as in Fig. 4(c). The first orientation leads to sample points spaced on a square grid on each plane, which is more suitable if the original data is arranged in this manner. However, the tetrahedra formed *intersect* these planes, such that the space between two planes cannot be exactly tessellated by them. In contrast, the second orientation leads to

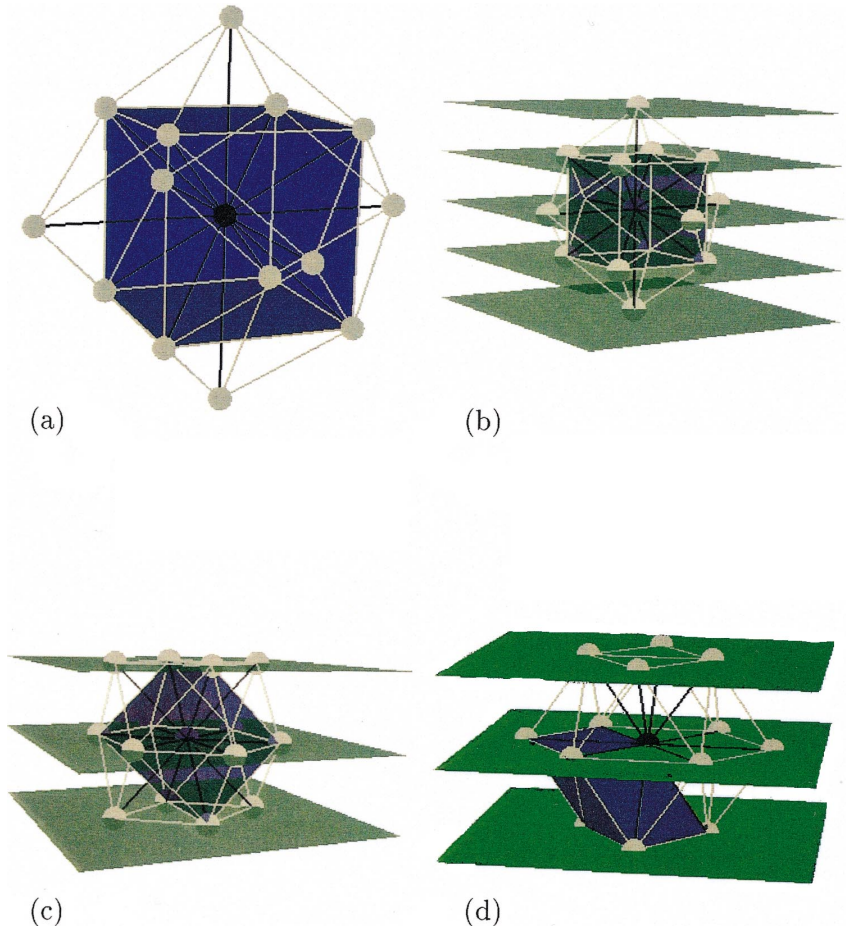


Fig. 4. Body centred cubic sampling. (a) shows a body-centred cubic lattice. (b) and (c) are two possibilities for defining a sample plane through (a). (b) has a square distribution of planar points whilst (c) aligns the tetrahedral faces with the plane. (d) shows the six tetrahedra associated with each point for the alignment of (c).

non-square (though still rectangular) spacing of data points on a plane, but with no tetrahedra intersecting it — this orientation is assumed in the following description. In this case, the sample points within the plane are spaced 2 units apart in one direction and $\sqrt{2}$ in the orthogonal direction. This need not be a problem if the sampled data already has to be interpolated from the original, as is the case, for instance, for distance functions based on a segmentation of the original data.

There are three main processing steps:

*Vertex marking:* Each sample point has 14 edges radiating from it, shown in black in Fig. 4(a) for the central point. Since each edge is associated with two sample points, only seven edges need be examined for each new sample point. Each of the 14 edges is assigned a reference in the form of a bit position. If any edge examined intersects the iso-surface (i.e. the sampled data at each end point is of opposite sign) then the appropriate reference is marked in a bit field for the *nearest* sample point on that edge. Hence, each sample point contains a bit field indicating the existence of any *near* surface intersections. The numbering used is given in Appendix A.

*Vertex clustering:* The first step in the second pass is to determine whether the near surface intersections for each sample point can be combined into new vertices, whilst preserving the topology. The positions of these new vertices can then be calculated from the sampled data and the locations of the sample points. Vertices thus created are stored in a simple list, along with an estimated surface normal. Each edge containing one of the original surface intersections is then marked with a reference to the appropriate new vertex. This means that, in general, several edges may be marked with the same vertex reference.

*Triangulation:* Each sample point is surrounded by 24 tetrahedra, and each tetrahedron is associated with

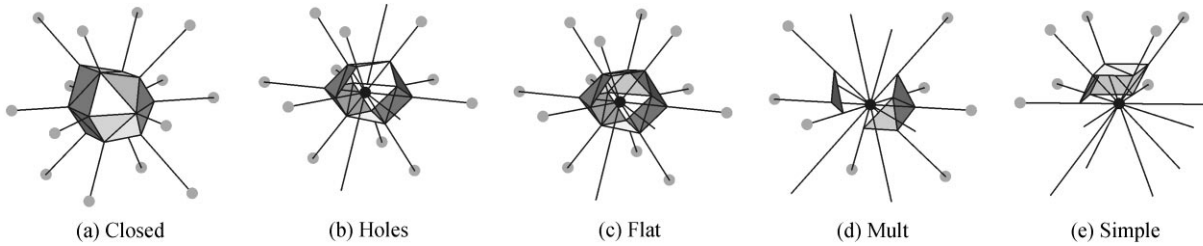


Fig. 5. Topology preservation. (a) to (e) show the possible topological cases considered during vertex clustering. (a), (b) and (c) cannot be clustered. (d) and (e) can be, with one new vertex per surface. Clustering of the vertices in (c) might lead to non-manifold surfaces.

4 sample points. Therefore there are six new tetrahedra associated with each sample point, as in Fig. 4(d). Each of the tetrahedra with surface intersections on any edges is triangulated using either case (a) or (b) from Fig. 2, as appropriate. However, any triangles which contain two or more vertex references which are identical are discarded before storage.

### 3.1. Topology preservation

There are a variety of situations where clustering the surface intersections near a sample point would lead to a change of topology in the resulting surface. The possible cases, shown in Fig. 5, are:

(a) *Closed surface:* The sample point has a value opposite in sign from all the surrounding points. Clustering the surface intersections to a single vertex would result in the elimination of this surface, so the original surface intersections become the new vertices.

(b) *Hole:* There is a single surface, but with a hole in it. Clustering the surface intersections would close up the hole, so once again the original surface intersections remain.

(c) *Flat surface:* There is a single surface with no hole, but the surface is concave such that clustering the surface intersections might result in 'flattening' the surface, leading to two edges or two triangles being back to back. Again, the original surface intersections remain.

(d) *Multiple surfaces:* The surface intersections form more than one separate surface, so one new vertex is required for each of these surfaces.

(e) *Simple surface:* The simple (and most common) case where there is only one surface can be clustered to a single new vertex with no change in surface topology.

Since the surface intersections are all marked in one bit of a bit field, the determination of the topological case can be performed simply by using logical operations and a table of nearest edges for each tetrahedral edge, included in Appendix A. For example, finding all the surface intersections which belong to one surface is performed by selecting an arbitrary edge with an intersection, then including any of the neighbouring edges that also contain surface intersections in a list of edges to check. This list then determines the next edge to check. The process is iterated until there are no more edges to check.

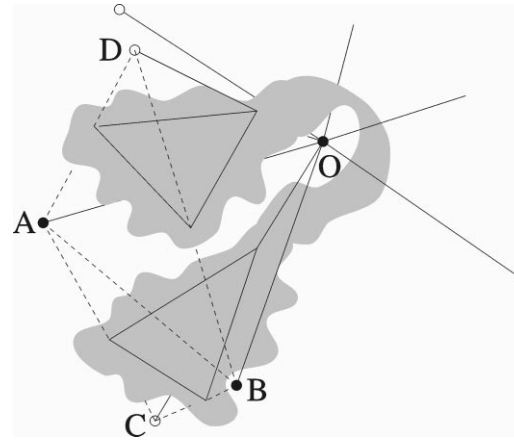


Fig. 6. Criterion for a 'flat surface'. $O$ is the current sample point, $A \ldots D$ are some of the neighbouring sample points. The existence of edges $OA$ and $OB$ with no near surface intersection, and edges $OC$ and $OD$ with near surface intersections, will lead to a collapsed surface if it is connected around $O$, and the surface intersections on edges $AD$ and $AC$ are both near $A$, or those on $BC$ and $BD$ are both near $B$.

The criterion for a 'flat surface', shown in Fig. 6, is a little more complicated. Each of the 36 outer edges (i.e. those joining the surrounding sample points, as $AB$ in the figure) are examined to see if they fall into the category described in the figure. The surface intersections surrounding such holes cannot all be clustered, as this would result in the closing up of portions of the surface, potentially making it non-manifold. Once again, this test can be performed efficiently by use of a bit table for each edge $AB$, giving the nearest neighbours $C$ and $D$. These tables are also included in Appendix A.

### 3.2. Triangle regularisation

Once a collection of surface intersections near to the sample point have passed the topology checks, they can be clustered to form a single new vertex. The reference for this vertex is then marked in each of the edges containing the original surface intersections. When tetrahedra are triangulated, these references are used to form the vertices of the triangles — if the triangles have repeated references, they are discarded before storage.

The simplest method of calculating the position of the new vertex is to take the average location of each of the surface intersections. Fig. 7 shows an example of an object whose surface has been reconstructed by calculating a distance function based on the object contours in Fig. 7(a). The triangulation formed by averaging the surface intersection positions, (d) and (g), is much more regular than the original MT triangulation, (c) and (f), and this results in a smoother rendering when using interpolated shading. In almost all cases, all surface intersections near a sample point can be clustered, so the triangulation ends up with roughly one vertex for each sample point which was near the surface. The evenness of the distribution is a direct result of the evenness of the original sample lattice.

### 3.3. Gradient and curvature calculation

Using the average of the surface point locations works well for smooth surfaces, but not as well for sharp corners or regions of high curvature. This can be seen in Fig. 7(g), where the front edge of the 'stand' has a sequence of specular reflections on it, due to the triangulation not following the actual edge of the surface. This can be overcome by using a data curvature estimate to weight the original surface point positions before averaging.

The previous technique involving tetrahedral lattices [2] used orthogonal estimates for gradient calculations, taking an average of these estimates over the nearest points to the sample point. We require a gradient and curvature estimate for each *edge*, since this is where the surface intersections will lie. The natural region of influence for this calculation is those tetrahedra which share this edge. The geometry is shown in Fig. 8. There are two types of edge, of length 2 and $\sqrt{3}$ units, the former surrounded by four tetrahedra as in (a), and the latter surrounded by six as in (c). In each case, the gradient and curvature are first calculated for each of the planes containing a pair of triangles. Fig. 9(a) shows a pair of triangles for an edge of length $\sqrt{3}$ units.

If the sampled values at each of the points $o$, $a$, $b$ and $c$ are $d_o \ldots d_c$, then the angle which the data surface in each triangle makes with $oa$ is given by

$$\theta_b = \arctan\left[\frac{\sin(\phi_b)}{(d_o - d_b)oa/(d_o - d_a)ob - \cos(\phi_b)}\right],$$

$$\theta_c = \arctan\left[\frac{\sin(\phi_c)}{(d_o - d_c)oa/(d_o - d_a)oc - \cos(\phi_c)}\right] \quad (1)$$

and the curvature, expressed as an angle which is 180° for a flat surface, is therefore

$$\alpha = |\theta_b| + |\theta_c|. \quad (2)$$

This curvature estimate requires further adjustment, since the plane in which it is calculated may be at a shallow angle to the actual surface, as in Fig. 9(b). If $\alpha$ is the original estimate, and the actual surface normal makes an angle $\gamma$ with a plane orthogonal to the original plane and the line $oa$, then the adjusted estimate, $\beta$ is given by

$$\frac{1}{\tan^2(\beta/2)} = (1 - \cos^2\gamma)\left(\frac{1}{\sin^2(\alpha/2)} - 1\right). \quad (3)$$

We are interested in the maximum curvature for each surface intersection, hence the minimum value of $|\tan(\beta/2)|$ in each plane is used as the curvature measure. The location of the new vertex $p_o$ is then calculated from the weighted locations of each surface intersection $s_i$ in the set of intersections to be clustered $S$, with the following equation:

$$\omega_i = \frac{1}{|\tan(\beta/2)|_{\min}}, \; p_o = \frac{1}{\sum_{i \in S} \omega_i} \sum_{i \in S} \omega_i s_i. \quad (4)$$

Note that an estimate of the actual surface normal for each surface intersection is also required for Eq. (3), in addition to being useful for shading the surface. $1/\tan\theta$ gives the projection of the normal in each plane orthogonal to the line $oa$ and scaled for a unit distance of $oa$. In the case of Fig. 8(a), vector addition of these projections in each of the two planes plus a unit vector along $oa$ gives an estimate of the surface normal. For the case in Fig. 8(c), the surface normal is given by $\frac{2}{3}$ of the vector addition of the projections in each plane, plus a unit vector along $oa$. This is because there are three planes at 60° to each other and hence *not* orthogonal.

### 3.4. Implementation details

For the lattice orientation of Fig. 4(c), the whole process can be performed by storing one plane of information, then progressing this plane through the data. The locations of each of the sample points on neighbouring planes alternate, so that two planes taken together form a rectangular lattice of sample points with half the spacing of an individual plane. One data structure is required for each of these points, with the following information:

- The location and the value of the current sample point.
- The location and the value of the last sample point at this lattice position (i.e. the plane before last).
- An unsigned integer indicating which of the 14 edges radiating from this point contain a near surface intersection.

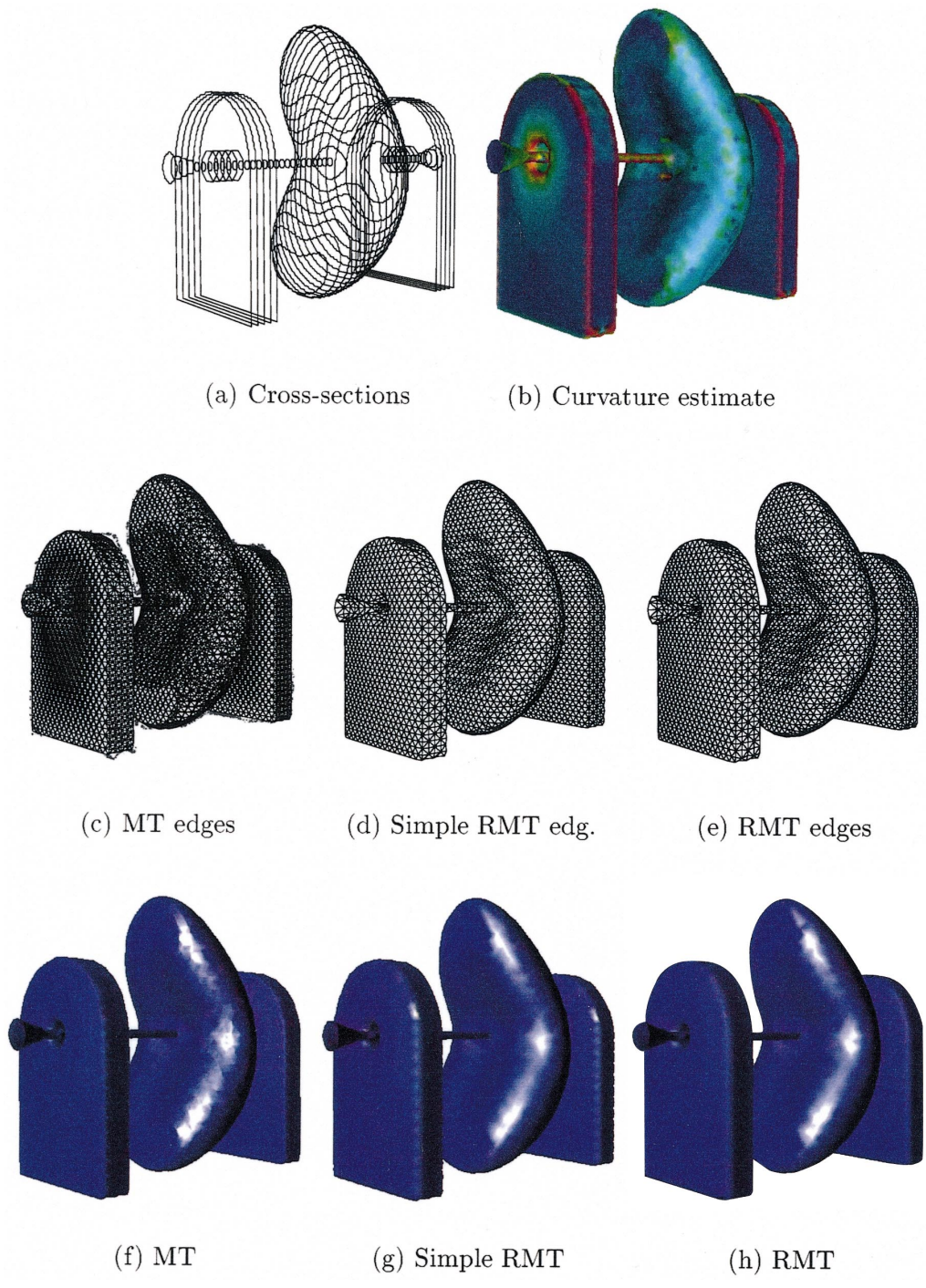


Fig. 7. Example iso-surface. (a) shows the object cross-sections from which the distance function was derived. (c) and (f) show the result of using MT, (d) and (g) include vertex clustering, and (e) and (h) also include weighting of the vertex position using curvature. (b) shows the curvature estimate mapped to colours — blue indicates low curvature through green then red indicating high curvature.

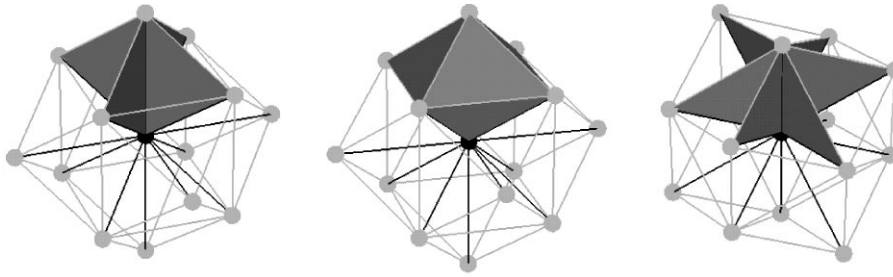


Fig. 8. Gradient and curvature calculation. An edge can be contained in (a) four or (c) six faces. Calculations of gradient and curvature projections within each of the (a) two or (c) three planes are combined to give the gradient and curvature estimates. (b) shows the region used to estimate values for the central edge in case (a).

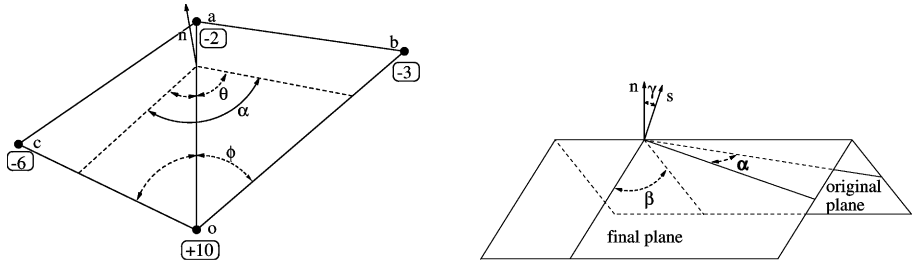


Fig. 9. Gradient and curvature calculation. (a) Gradients and curvatures for an edge are calculated in each plane using the two triangles neighbouring the edge. (b) Curvatures must then be adjusted for the orientation of the plane with respect to the calculated surface normal.

- A reference for each of the seven edges associated with this point, indicating the vertex created for this edge, if any.

In the first pass through a plane, the location and value of the current sample point are calculated (and the previous values updated), and any near surface intersections are marked in the appropriate sample point. In the second pass, the clustering and marking of the new vertices thus created in the appropriate edge reference, is performed. Triangulation can then be performed for each of the six tetrahedra associated with this point, based on the vertex references. For grids which are entirely orthogonal (i.e. not like the ultrasound example in Section 4.3), the locations for each point can be inferred from the lattice indices.

The triangles and vertices can be stored in two simple lists, since it is not necessary to search either list at any point in the process. Each triangle contains a (32-bit) reference to each of three vertices. Each vertex contains a location (a 16-bit integer was used for each direction) and a surface normal (again using 16-bit integers). Each vertex is stored once only, but can be referenced by several triangles.

## 4. Results and discussion

The effects of RMT on vertex and triangle count, volume and surface area measurements, triangle aspect ratio and interpolated shading of the surface have been investigated, for a variety of surfaces. Comparisons were made with MT and with MC at a resolution such that the inter-plane spacing was the same in all cases. As can be seen from Table 1, both the number of vertices and triangles are reduced by about 70% compared to MT and 40% compared to MC. This reduction is affected by the complexity of the surface at the sampling resolution. The skull, which has the highest complexity, is reduced slightly less, since no vertex clustering can be performed for the 'closed surface', 'multiple holes' and 'flat surface' topological cases. In all the other examples, and indeed in most practical situations, there are very few occurrences of topological cases which prevent clustering, and such occurrences are often an indication of the lack of appropriate filtering.

The times quoted in Table 1 include interpolation of the original data (or function evaluation for the implicit surfaces) in addition to iso-surface extraction from that data. The software was written in C and run on a Silicon

Table 1
Triangulation results. 'Example' is from Fig. 7, 'Peaks' from Eq. (5) and 'Closed surface' from Eq. (6). The topological cases indicated for each sample point are cs: Closed surface, ms: Multiple surfaces, mh: Multiple holes, fs: Flat surface. The processing time is for RMT

| Object | Vertices | | | Triangles | | | Topological cases | | | | Time |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | MC | MT | RMT | MC | MT | RMT | cs | ms | mh | fs | (s) |
| Sphere | 1,161 | 2,480 | 660 | 2,320 | 4,956 | 1,316 | 0 | 0 | 0 | 0 | 0.40 |
| Example | 8,646 | 20,924 | 5,563 | 17,450 | 41,796 | 11,102 | 0 | 20 | 12 | 0 | 1.04 |
| Peaks | 3,150 | 8,415 | 2,199 | 6,130 | 16,498 | 4,238 | 0 | 1 | 0 | 0 | 1.60 |
| Closed | 4,113 | 9,368 | 2,500 | 8,196 | 18,744 | 5,008 | 0 | 6 | 0 | 0 | 1.31 |
| Skull | 78,103 | 169,750 | 50,543 | 156,528 | 339,942 | 101,573 | 19 | 974 | 258 | 445 | 11.28 |
| Bladder | 15,053 | 28,524 | 7,828 | 30,213 | 56,848 | 15,556 | 0 | 0 | 0 | 0 | 5.76 |

Table 2
Volume and surface area results. The objects are as in Table 1. The values shown are not calibrated, except for the skull and bladder, in which case the volume is in *ml*, and the surface area in $cm^2$

| Object | Volume | | | Surface area | | |
|---|---|---|---|---|---|---|
| | MT | RMT | % | MT | RMT | % |
| Sphere | 1.0159 | 1.0121 | 99.63 | 4.8905 | 4.8793 | 99.77 |
| Example (simple RMT) | 0.7728 | 0.7699 | 99.62 | 8.8393 | 8.7428 | 98.91 |
| Example (RMT) | 0.7728 | 0.7716 | 99.84 | 8.8393 | 8.7799 | 99.33 |
| Peaks | — | — | — | 9.2656 | 9.2482 | 99.81 |
| Closed surface | 0.8196 | 0.8145 | 99.38 | 7.9559 | 7.8811 | 99.06 |
| Skull | 362.84 | 361.95 | 99.75 | 1614.8 | 1602.9 | 99.26 |
| Bladder | 317.60 | 317.51 | 99.97 | 249.37 | 249.01 | 99.86 |

Graphics workstation.[2] It was designed and optimised for *non-parallel* lattices (such as the bladder in Fig. 13), and hence the processing times for the other examples are significantly more than could be achieved with a design specifically for an orthogonal lattice. Despite this, triangles are typically generated at 4000 to 10,000 per second, dependent on the complexity of the interpolation or underlying function.

The effect of the iso-surface extraction method on the volume and surface area of the triangulation is compared to that for standard MT in Table 2. In all cases, there were approximately 100 sample points spanning the objects in each direction. This leads to a 1% error in distance, 2% error in area and 3% error in volume measurement. By comparison, all of the measurements on the objects (with the exception of that made on the example with no curvature information, i.e. simple RMT) are within 1% of the measurements calculated with MT. The volume measurements were calculated from the triangulation by a method derived from Gauss' theorem and given in [12].

The improvement in triangle aspect ratios can be seen from the triangulations shown in Figs. 3, 7, 10, 11 and 13. This is quantified for the last example in the graph of Fig. 12. Here, the triangles for MC, MT and RMT have been sorted in order of increasing aspect ratio, and normalised by the total number of triangles. The first peak on the graph represents near equilateral triangles, and the second smaller peak near right-angled isosceles triangles. These two types of triangle represent those formed from intersecting the four tetrahedra around a 2 unit edge, and the six tetrahedra around a $\sqrt{3}$ unit edge, respectively.

The aspect ratio improvement is reflected in the improved shaded surface displays,[3] particularly evident for the sphere in Fig. 3 and the example in Fig. 7. These differences are even more apparent on modern 24-bit colour displays, where the sharp edges and points caused by poor aspect ratio triangles become quite pronounced.

[2] SGI Indigo 2 R10000 CPU, Silicon Graphics Incorporated, Mountain View, California.

[3] Rendering was performed using Geomview, http://www.geom.umn.edu/software/geomview/.

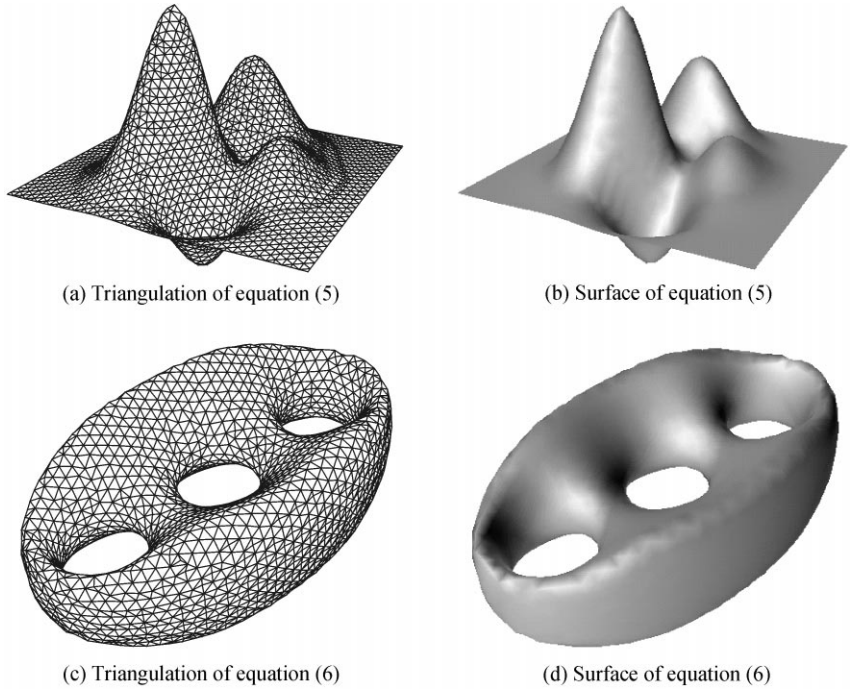


Fig. 10. Iso-surfaces of implicit functions. (a) and (b) are the peaks function included with Matlab v5.0. (c) and (d) are a closed surface as defined in [9].

They are also very apparent when the surfaces are rotated. Unfortunately, it is not possible to fully encapsulate these effects on paper. The displayed surfaces are Gouraud shaded with diffuse and specular reflection, in order to accentuate any sharp edges or points.

### 4.1. Implicit surfaces

The triangulations of two implicit iso-surfaces are shown in Fig. 10. Here, the data at the sample points is taken directly from a function. The surfaces shown are for $f(x, y, z) = 0$. Fig. 10(a) and (b) are from the well-known 'peaks' function provided with Matlab v5.0:[4]

$$f(x, y, z) = (3 - 3x)^2 e^{-x^2-(y+1)^2} - 10\left(\frac{x}{5} - x^3 - y^5\right)e^{-x^2-y^2} - \frac{1}{3}e^{-(x+1)^2-y^2} - z. \quad (5)$$

Fig. 10(c) and (d) are from a closed surface used in a recent paper [9]:

$$f(x, y, z) = \left(1 - \left(\frac{x}{6}\right)^2 - \left(\frac{y}{3.5}\right)^2\right) \times ((x - 3.9)^2 + y^2 - 1.44)(x^2 + y^2 - 1.44) \times ((x + 3.9)^2 + y^2 - 1.44) - z^2. \quad (6)$$

### 4.2. Thresholded surfaces

Iso-surfaces of thresholded data, particularly medical data, are usually topologically complex compared to implicit functions or surfaces from cross-sections. The skull in Fig. 11 is an example of such a surface[5] [25]. Here, a high-resolution (0.41 mm by 0.41 mm by 1 mm) computed tomography (CT) scan of a child's skull has been thresholded using the appropriate coefficient for bone, and shape-based interpolation [10] has been used to improve the surface normals. The resolution of the

[4] (c) Copyright 1984-96 The MathWorks, Inc., http://www.mathworks.com.

[5] From CHILD.IM0, provided with 3DViewnix v1.1.1 (c) 1993–1996 M I P G University of Pennsylvania, All Rights Reserved.

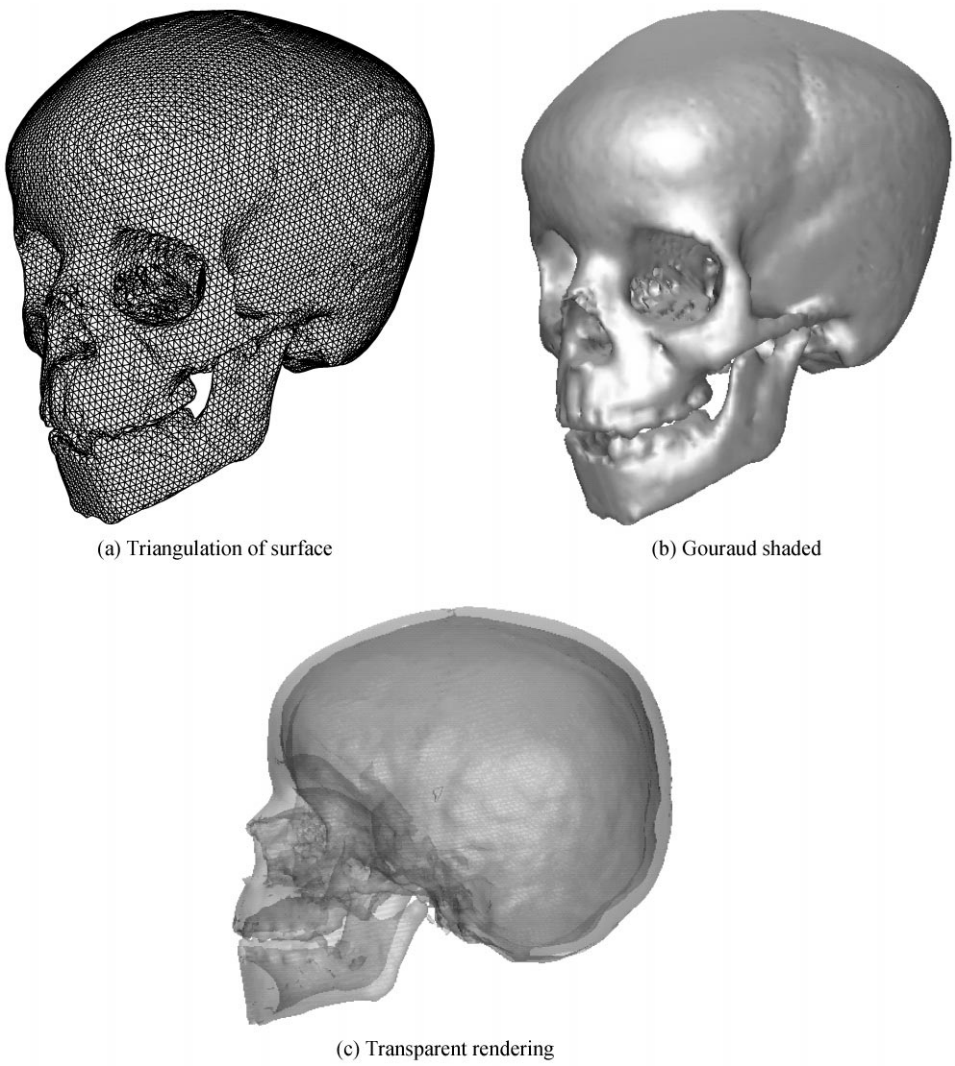


Fig. 11. Iso-surfaces for thresholded data. The data is from a CT scan of a child's skull, the sample point spacing being 2 mm. (c) Semi-transparent surface rendering of both the inner and outer surfaces of the skull, showing the skull thickness.

triangulation is much less than that of the data — there are only 50,000 triangles in the outer surface. Both the inside and outside skull surfaces are extracted in addition to many internal structures which are impossible to view from only one direction. The resulting surface can also be rendered semi-transparent as in Fig. 11(c) to reveal detail such as the skull thickness.

### *4.3. Surface from non-parallel cross-sections*

Iso-surface techniques, unlike many 'surface from cross-sections' triangulation techniques, can very simply be extended for data from non-parallel cross-sections. This is particularly useful for freehand 3-D ultrasound data [24]. In this case, the lattice on which the samples are taken is no longer orthogonal, but is allowed to align itself with the planes containing the original data. The algorithm then proceeds sequentially from plane to plane, generating an iso-surface of the interpolated data between one pair of planes at a time.

On any given plane, the lattice is defined exactly as in Fig. 4(c). If the planes are separated anywhere by a distance greater than the in-plane sample distance, additional sample points are created between these planes, by projecting an average normal to each of the original planes from each in-plane sample point, and splitting this into an appropriate number of segments. The lattice for the next plane is similarly calculated by projecting the

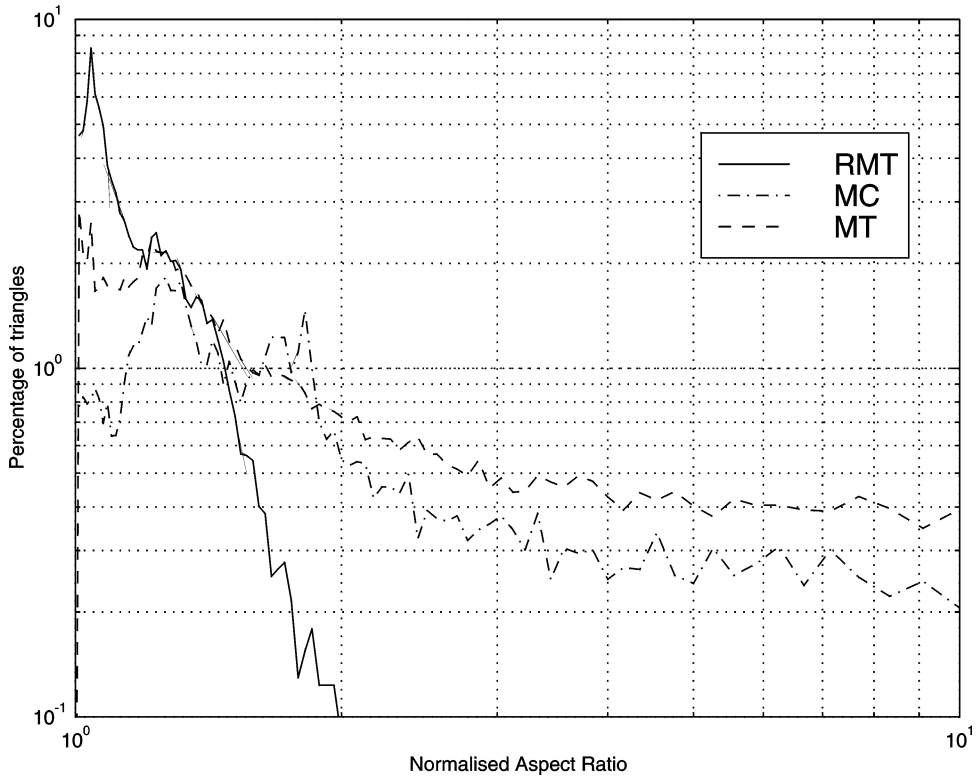


Fig. 12. Effect of RMT on triangle aspect ratio for a human bladder. The aspect ratio is defined by the ratio of the radius of the circumscribed circle, to the radius of the inscribed circle. This is normalised by the aspect ratio for an equilateral triangle.

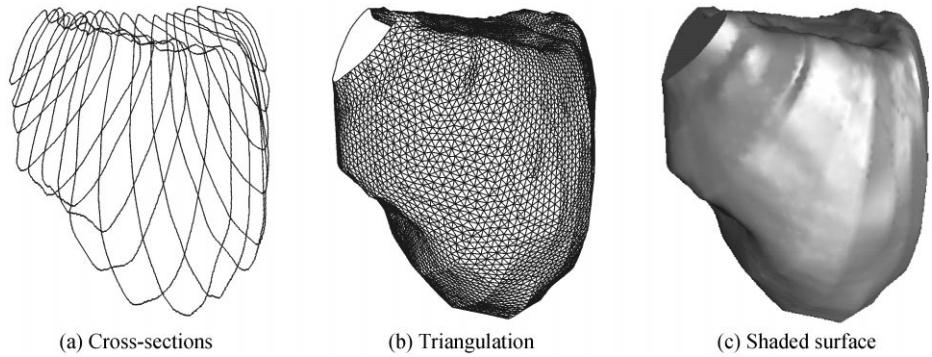


Fig. 13. Surface from cross-sections. The data is from a freehand 3-D ultrasound scan of a bladder. The probe was moved using a fanning action, generating the cross-sections (after segmentation) in (a). (b) and (c) show the surface after maximal disc guided interpolation and RMT iso-surface extraction. Planar polygons have been fitted to the first and last cross-sections to close the surface.

lattice of the previous plane along the average normal. This ensures that lattices between each pair of planes meet at the in-plane sample points.

Once the sample point locations have been determined, the processing proceeds as previously described, save that the locations of the vertices need to be calculated carefully, since they are no longer on a simple rectangular grid. In addition, the data interpolation scheme must also be capable of handling non-parallel data planes.

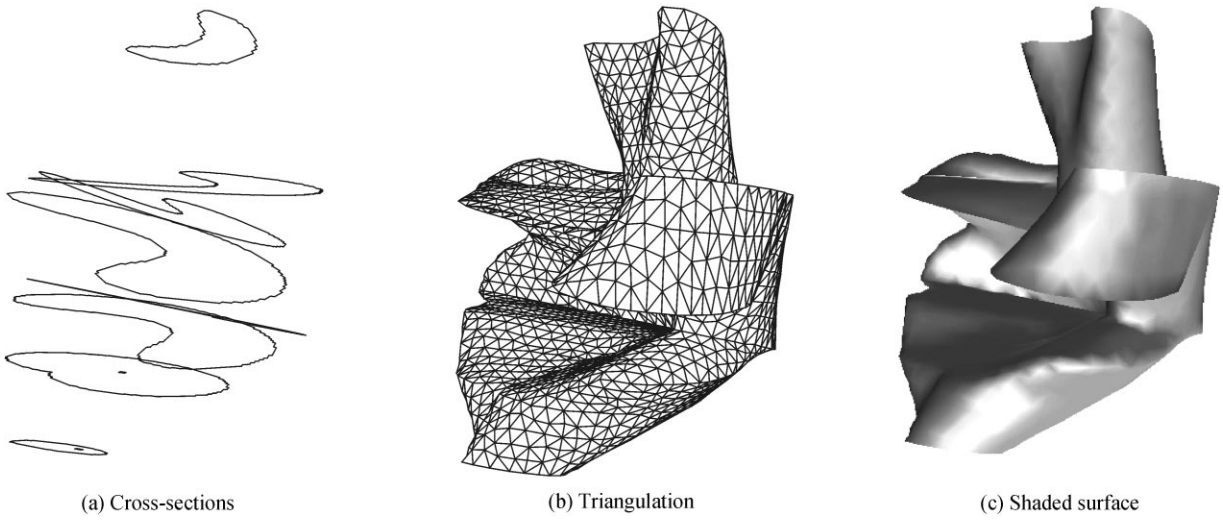


Fig. 14. Surface from cross-sections. The contours have been highly mis-registered such that the planes in which they are defined overlap each other. The resulting surface is self intersecting, but consistent with the mis-registered contours.

An example of this process is shown in Fig. 13, which is a segmented ultrasound scan of a bladder acquired using Stradx v5.0[6] software [22]. Interpolation was performed using maximal disc guided shape based interpolation, described in [24]. This interpolation technique, and RMT, are included in the latest version of Stradx.

It is even possible to handle data which is from overlapping, non-parallel planes, as in Fig. 14. A few modifications are required for handling this case. Firstly, the direction of sweep of the sequential planes affects the orientation of the triangles. Unfortunately, this imposes two opposite requirements — for correct surface shading, all triangles should be oriented consistent with outside/inside criteria. For correct volume measurement, when the sweep direction changes, the triangle orientation should flip such that the volume is then reduced as a result. If volume calculation is performed after triangulation, an additional flag is required per triangle, indicating the sweep direction of the planes which the triangle lies between.

Secondly, surface intersection points which are on planes at the extremity of a sweep direction, are only allowed to move within that plane. This ensures that the edge of the surface will still be contained within the original plane after vertex clustering. Thirdly, sample points which are near to the intersection of two overlapping planes are moved on to the line of intersection. This ensures that there are no tetrahedra which are themselves intersected by this line, and hence would not be correctly triangulated.

## 5. Conclusions

*Regularised marching tetrahedra* is a fast iso-surface extraction method suitable for implicit or discrete data. The volume and surface error differences, compared to MT, are much less than the error inherent in the sampling resolution. However, the number of triangles is reduced by 70% compared to MT, and by 40% compared to MC. Interpolated shading of the surface, particularly Gouraud shading, is greatly improved by the better aspect ratio of the triangles. Unlike other vertex clustering methods, the topology is consistent with the sampled data.

For applications where the resolution can be reduced, the tetrahedral grid can be formed from a sub-set of the original orthogonal grid. For applications which already require interpolation of the original data, there is no disadvantage in using a tetrahedral lattice for the interpolated data. Iso-surface extraction at the highest resolution of the original data is only possible if additional points are interpolated from this data.

## 6. Further work

The vertex clustering method is not optimal in terms of minimising either surface area, volume or surface curvature errors. Other methods involving principal curvature calculations [7] would lead to more accurate positioning

[6] http://svr-www.eng.cam.ac.uk/~rwp/stradx/.

of the amalgamated vertex, though at the cost of increased complexity and processing time. It is also of interest to consider how RMT may affect subsequent triangle reduction steps, and whether the method can be adapted to generate triangle strips, or to work from unstructured tetrahedral meshes. Our thanks are due to the reviewers for these suggestions.

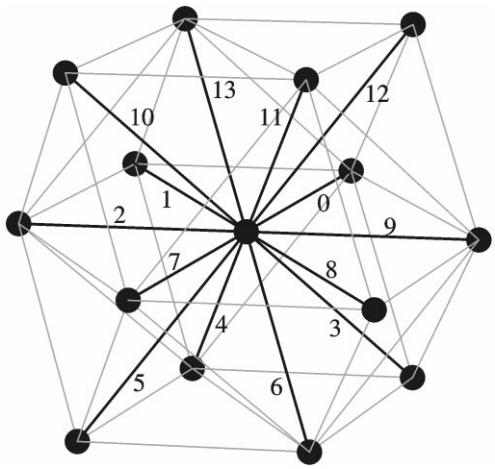


Fig. 15. Edge labels for topological tables.

**Acknowledgements**

The authors would like to thank Laurence Berman, of the University of Cambridge Department of Radiology, Addenbrooke's Hospital, for performing the scanning and segmentation of the human bladder in Figure 13. Graham Treece would also like to thank his wife, Sarah Treece, for her continuing encouragement and support. Graham Treece is supported by an EPSRC studentship and a Newton Trust award from the Cambridge University Department of Engineering.

## Appendix A. Tetrahedral edge tables for topology preservation

The 14 edges associated with one sample point are given the labels 0 … 13, as in Fig. 15. Edges 0 … 6 are the 7 new edges introduced when the sample point is traversed for the first time. It is convenient to label the opposite set of edges by adding 7 to each of these labels.

Each of the edges is assigned a bit number equivalent to the label, such that only a single 16-bit integer is required to mark all of them. Gathering points together which form a single surface can be done using the hexadecimal masks in Table 3, as follows:

Table 3
Nearest neighbour masks. If the edges 0 … 13 in Fig. 15 are assigned bit positions 0 … 13, then the hexadecimal masks below give the nearest edges. If the current edge is 2 units, there are four neighbours, else if $\sqrt{3}$ units there are six neighbours

| Edge | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Neighbours | 6 | 4 | 6 | 4 | 6 | 4 | 6 | 6 | 4 | 6 | 4 | 6 | 4 | 6 |
| Near mask | 321A | 2015 | 24B2 | 0251 | 006F | 00D4 | 03B8 | 0D64 | 0AC0 | 1949 | 2884 | 3780 | 2A01 | 1C07 |

Table 4
'Flat surface' topology masks. The hexadecimal masks give the vertices of all the outer edges, i.e. those marked in grey in Fig. 15. The first check for a 'flat surface' is that there are intersections on both of the edges in the 'adjacent' mask, on neither of the edges in the 'opposite' mask

| Edge | 0–1 | 0–3 | 0–4 | 0–9 | 0–12 | 0–13 | 1–2 | 1–4 | 1–13 | 2–4 | 2–5 | 2–7 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Adjacent mask | 0003 | 0009 | 0011 | 0201 | 1001 | 2001 | 0006 | 0012 | 2002 | 0014 | 0024 | 0084 |
| Opposite mask | 2010 | 0210 | 000A | 1008 | 2200 | 1002 | 2010 | 0005 | 0005 | 0022 | 0090 | 0420 |
| Edge | 2–10 | 2–13 | 3–4 | 3–6 | 3–9 | 4–5 | 4–6 | 5–6 | 5–7 | 6-7 | 6–8 | 6–9 |
| Adjacent mask | 0404 | 2004 | 0018 | 0048 | 0208 | 0030 | 0050 | 0060 | 00A0 | 00C0 | 0140 | 0240 |
| Opposite mask | 2080 | 0402 | 0041 | 0210 | 0041 | 0044 | 0028 | 0090 | 0044 | 0120 | 0280 | 0108 |
| Edge | 7–8 | 7–10 | 7–11 | 8–9 | 8–11 | 9–11 | 9–12 | 10–11 | 10–13 | 11–12 | 11–13 | 12–13 |
| Adjacent mask | 0180 | 0480 | 0880 | 0300 | 0900 | 0A00 | 1200 | 0C00 | 2400 | 1800 | 2800 | 3000 |
| Opposite mask | 0840 | 0804 | 0500 | 0840 | 0280 | 1100 | 0801 | 2080 | 0804 | 2200 | 1400 | 0801 |

```
MarkedEdges = {all marked edges}
Surfaces = 0
while ∃MarkedEdges
  DoneEdges = ∅
  ToDoEdges = {one edge from MarkedEdges}
  while ∃ToDoEdges
    i = {one edge from ToDoEdges}
    add i to DoneEdges
    add (MarkedEdges∩NearMask(i)) to ToDoEdges
    remove DoneEdges from ToDoEdges
  remove DoneEdges from MarkedEdges
  Surface(Surfaces) = DoneEdges
  increment surfaces
```

Here the operations have been described in set notation, but each set can be represented by a single integer, and the set operations replaced by logical operations on such integers. At the end of the operation, *Surface* is an array of *Surfaces* integers, which each contain edges from *MarkedEdges* which are part of the same surface.

The first step in the check for the 'flat surface' topology described in Section 3.1 can similarly be done using Table 4, as follows:

```
for i = 0 ... 35 do
  if (AdjacentMask(i)∩SurfaceEdge) ≡ ∅
    if (OppositeMask(i)∪SurfaceEdge) ≡ SurfaceEdge
      Further checks for a flat surface ...
```

This check is repeated for each of the separate surfaces; *SurfaceEdge* contains the associated edges for the surface. The further checks are described in Fig. 6.

## References


[1] Algorri M-E, Schmitt F. Mesh simplification. Computer Graphics Forum 1996;15(3):C–77–C–86.

[2] Chan SL, Purisima EO. A new tetrahedral tesselation scheme for isosurface generation. Computers and Graphics 1998;22(1):83–90.

[3] Cignoni P, Montani C, Scopigno R. A comparison of mesh simplification algorithms. Computers and Graphics 1998;22(1):37–54.

[4] Foley JD et al. Computer graphics: principles and practice, 2nd Edition, Addison-Wesley Systems Programming Series. Reading, MA: Addison-Wesley, 1996.

[5] Garland M, Heckbert PS. Surface simplification using quadric error metrics. In SIGGRAPH '97 Proceedings, Computer Graphics, August 1997. p. 209–16.

[6] Guéziec A, Dean D. The Wrapper: A surface optimization algorithm that preserves highly curved areas. In Visualization in Biomedical Computing, SPIE 1994;2359: p. 631–42

[7] Guéziec A, Hummel R. Exploiting triangulated surface extraction using tetrahedral decomposition. IEEE Transactions on Visualization and Computer Graphics 1995;1(4):328–42.

[8] Hall M, Warren J. Adaptive polygonization of implicitly defined surfaces. IEEE Computer Graphics and Applications 1990;10(6):33–40.

[9] Hartmann E. A marching method for the triangulation of surfaces. The Visual Computer 1998;14(2):95–108.

[10] Herman GT, Zheng J, Bucholtz CA. Shape-based interpolation. IEEE Computer Graphics and Applications 1992; 12(3):69–79.

[11] Hoppe H, DeRose T, Duchamp T, McDonald J, Stuetzle W. Surface reconstruction from unorganized points. Computer Graphics 1992;26(2):71–8.

[12] Hughes SW, D'Arcy TJ, Maxwell DJ, Saunders JE, Ruff CF, Chiu WSC, Sheppard RJ. Application of a new discreet form of Gauss' theorum for measuring volume. Physics in Medicine and Biology 1996;41:1809–21.

[13] Liu Y-H, Sun Y-N, Mao C-W, Lin C-J. Edge-shrinking interpolation for medical images. Computerized Medical Imaging and Graphics 1997;21(2):91–101.

[14] Lorensen WE, Cline HE. Marching cubes: a high resolution 3D surface construction algorithm. Computer Graphics 1987;21(4):163–8.

[15] Montani C, Scateni R, Scopigno R. A modified look-up table for implicit disambiguation of Marching Cubes. The Visual Computer 1994;10:353–5.

[16] Montanvert A, Usson Y. Discrete distances applied to 2D granulometry and 3D reconstruction. In 8th Scandinavian Conference on Image Analysis 1993. p. 1153–1160.

[17] Natarajan BK. On generating topologically consistent isosurfaces from uniform samples. The Visual Computer 1994;11:52–62.

[18] Ning P, Bloomenthal J. An evaluation of implicit surface tilers. IEEE Computer Graphics and Applications 1993;13(6):33–41.

[19] Oliva J-M, Perrin M, Coquillart S. 3D reconstruction of complex polyhedral shapes from contours using a simplified generalized Voronöi diagram. Computer Graphics Forum 1996; 15(3):C–397–C–408.

[20] Payne BA, Toga AW. Surface mapping brain function on 3D models. IEEE Computer Graphics and Applications 1990;10(5):33–41.

[21] Payne BA, Toga AW. Distance field manipulation of surface models. IEEE Computer Graphics and Applications 1992;12(1):65–71.

[22] Prager RW, Gee AH, Berman L. Stradx: real-time acquisition and visualisation of freehand 3D ultrasound. Medical Image Analysis 1999;3(2):129–40.

[23] Reddy M. SCROOGE: perceptually-driven polygon reduction. Computer Graphics Forum 1996;15(4):191–203.

[24] Treece GM, Prager RW, Gee AH, Berman L. Fast surface and volume estimation from non-parallel cross-sections, for freehand 3-D ultrasound. Medical Image Analysis 1999;3(2):141–73.

[25] Udupa JK, Odhner D, Samarasekera S, Goncalves R, Iyer K, Venugopal K, Furuie S. 3DVIEWNIX: an open, transportable, multidimensional, multi-modality, multiparametric imaging software system. In SPIE Proceedings 1994;2164: p. 58–73.

[26] Wilhelms J, Gelder AV. Topological considerations in isosurface generation — extended abstract. Computer Graphics 1990;24(5):79–86.

[27] Zhou Y, Chen W, Tang Z. An elaborate ambiguity detection method for constructing isosurfaces within tetrahedral meshes. Computers and Graphics 1995;19(3):355–64.