Automedica, 1992, Vol. 14, pp. 297–310
Reprints available directly from the publisher
Photocopying permitted by license only

MARCHING CUBES: ALGORITHMIC IMPROVEMENTS

MARTIN LIVERSAGE[1,2], GULAB BHATIA[1] and MICHAEL VANNIER[1]

[1]*Mallinckrodt Institute of Radiology, Washington University School of Medicine, 510 S. Kingshighway Blvd., St Louis, MO 63110, and* [2]*Royal Dental College Copenhagen, Department of Pediatric Dentistry, Nørre Allé 20, DK-2200 København, N. Denmark*

(Received November, 1991)

Improvements to the marching cubes algorithm for surface modeling of space filling objects in volumetric data sets have been made. Specifically, optimization and parallelization of the algorithm for generation of non-planar polygons well suited to display in a high-end computer graphics environment (Silicon Graphics power series 4D/340) was done using an extension of the original Lorenson and Cline algorithm. The method was implemented and tested using CT scan data from a child with skull deformity due to Apert's syndrome.

The effect of incoherences introduced by the original marching cubes algorithm and the asymmetries introduced by our extensions were measured and found to be negligible in a CT scan surface model of the head.

Keywords: Marching cubes, three-dimensional computer graphics, surface modeling of biomedical objects, craniofacial deformities

1. INTRODUCTION

The marching cube algorithm developed by Lorenson and Cline [1] generates three-dimensional surface models of space-filling objects from volumetric data sets. It was originally developed for application to computed tomographic 3D image reconstruction [2], and was extended to magnetic resonance imaging. This algorithm has been extended and refined, and comparisons to other 3D surfacing algorithms have been made [3].

Marching cubes is particularly attractive for use with systems that manage and display polygons efficiently. Marching cubes effectively translates volumetric data into a polygonal surface form which can be efficiently rendered with advanced lighting and image synthesis effects using accelerated hardware and parallel processors.

We have added several important improvements to the marching cubes algorithm and implemented it in an advanced parallel processing high-end computer graphics workstation environment (Silicon Graphics 4D/340).

2. THE ALGORITHM

A set of volumetric data can be examined after surfaces of included objects are detected and translated into polygons. The volume is represented by a real valued function, $f(x, y, z)$, defined on a cubic grid.

The goal is to model an isosurface within this volume. This is a surface in space described by a threshold value, T. A point v (also called a voxel) in the volume is

inside (or on) the isosurface if $f(v) \geq T$. Otherwise it is outside. Normally we describe the discrete volume as a point sample of a continuous volume, and will model the isosurface as a discrete surface approximating the continuous isosurface in the original volume.

2.1 *Describing the Surface*

Consider a pair of adjacent voxels connected by an edge. One voxel lies on the surface, and the other is outside. The line segment which connects the centers of the two voxels will penetrate the surface (Figure 1).

The exact position of the point in space where the edge intersects the surface is determined by linear interpolation as follows: Let p be the vector representing the point of intersection and let v_1 and v_2 be the vectors representing the positions of the voxels in space. We have

$$p = \frac{T - f(v_1)}{(f(v_2) - f(v_1))^* (v_2 - v_1)} + v_1.$$

Consider one cube described by eight voxels. If all voxels are on or all voxels are off, the cube will be entirely inside or outside the isosurface. Otherwise the isosurface will cross within the cube as seen in Figure 2. We define a surface segment with respect to a given cube as the part of the isosurface that is enclosed in the cube.

Now, consider another cube adjacent to the first one so that the cubes share a face. These two cubes will share four voxels (Figure 3). In this example one of the voxels is off and the rest are on. This means that the isosurface will intersect the shared face.

First, consider the cube labeled A. The point of intersection of the isosurface between voxels v_1 and v_2 is p_a. This point is calculated as described earlier. Next, consider the cube labeled B. The point of intersection of the isosurface between the same two voxels is here p_a'. The position in space of this point is only determined by voxels v_1 and v_2 so it is obvious that $p_a' = p_a$. The same can be said about p_b and p_b'. They are equal. This means that it is possible to determine the isosurfaces within single cubes and then stack the cubes together. The isosurfaces of adjacent cubes will fit together. The meaning of "fit together" is that the points of intersection of the cube edges and the isosurface are the same in adjacent cubes.

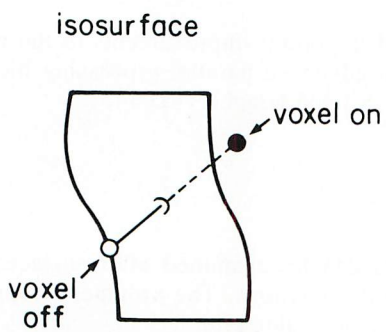


Figure 1 Surface description of an object using an isosurface. A line outline segment connects the centers of two voxels, one within and the other outside the surface. The line segment connecting the centers of the two voxels penetrates the isosurface.

To create the proper surface, the points of intersection of the isosurface and the cube edges have to be connected. The simplest way to do this is to connect these points by straight lines. The surface segment will then be a (generally) nonplanar polygon and may consist of several non-connected surfaces. But how do we connect these polygon vertices? That is a subtle point of the Marching Cubes algorithm; the spatial coherence of extracted surfaces is not assured.

Consider a given surface segment vertex on the edge of a cube. It has to be connected to two other vertices of the surface segment positioned on other edges of the cube. Any edge on a cube will be at an intersection of two faces of the cube and so the surface segment vertex will be adjacent to two faces (Figure 4). The two edges of the surface segment connected to the vertex of the surface segment have to lie in each of the two faces of the cube. As seen from Figure 4 there are three ways each surface segment edge can divide the face. To resolve which way to divide the face we have to study possible candidates for surface segment vertices in the same face to connect to. This is shown in Figure 5 where only one cube face is seen. The first three cases are obvious as there is only one other surface segment vertex in the face, but case D is

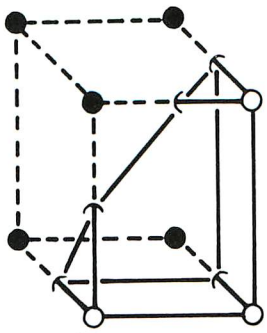


Figure 2 One cube may be represented by eight voxels shown as the corner vertices. A surface may cross the cube as illustrated in this figure. The filled vertices represent portions of the cube which are within the object while the open vertices are outside.

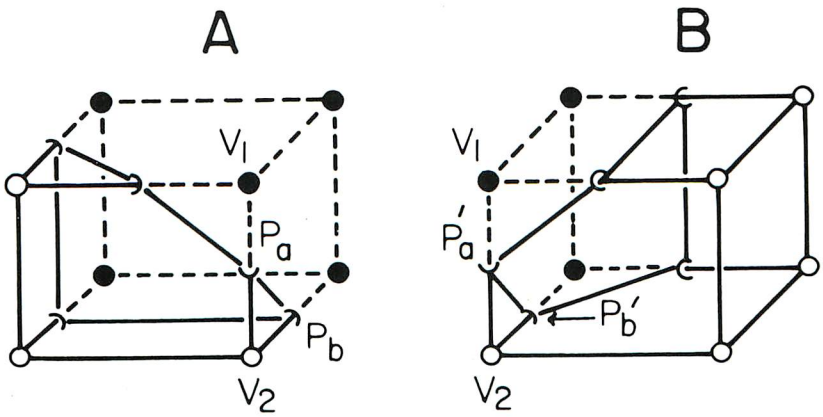


Figure 3 Two adjacent cubes share four voxels. This example shows one of the shared voxels outside the surface while the rest are inside. This indicates that the isosurface must intersect the shared face.

ambiguous. There are three possible vertices to connect to. The vertex on the opposite side of the face is not the right candidate as the surface segment edge needs to divide the face into cube vertices on and cube vertices off. Unfortunately, there are still two candidates left. This is the weak aspect of the Marching Cubes algorithm.

We want to be able to stack cubes along their faces so the polygon edges coincide, so we need to make an arbitrary decision to resolve the ambiguous case. We can either decide to maximize the area of the face inside the isosurface or the area outside the isosurface as A and B in Figure 6. We arbitrarily choose the first option.

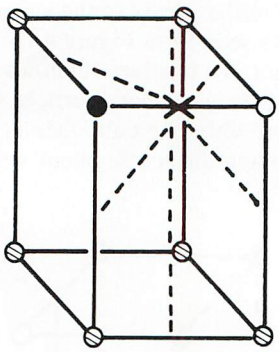


Figure 4 In this case, the surface segment vertex will be adjacent to two faces given a surface segment vertex on the edge of a cube.

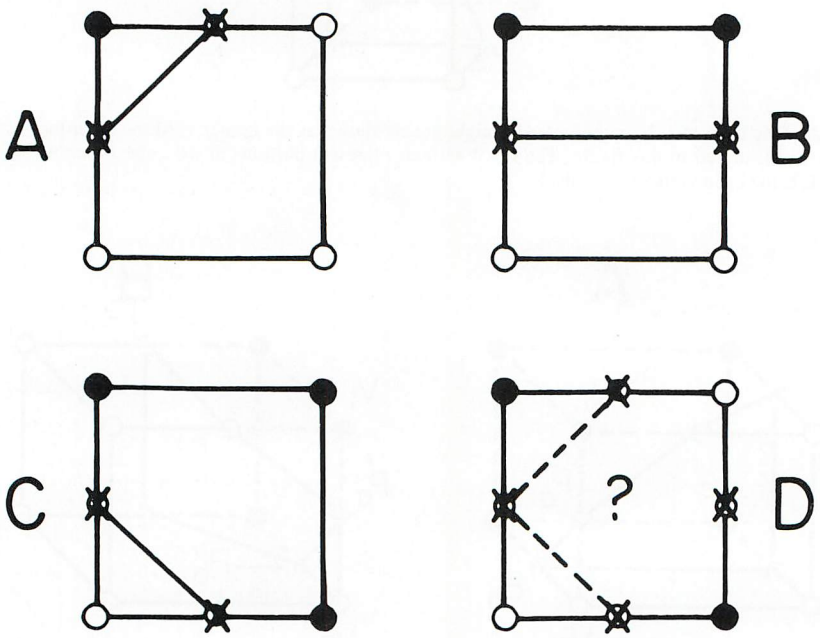


Figure 5 Only one cube face is seen. In the first three cases, A, B, and C, there is only one other surface segment vertex in the face. In case D, the situation is ambiguous. Three possible vertices may be connected. This ambiguity represents a weakness in the marching cubes algorithm.

Now, when we stack cubes together the edges of the surface segment will match exactly. We have already pointed out that the vertices of the surface segment fit together, and as we are connecting surface segment vertices by straight lines using a special rule for resolving ambiguities, these lines will match when cubes are stacked.

2.2 *Exploiting Cube Symmetries*

We define the term "curface" as a cube together with its surface segment. A curface has vertices that can be on or off the isosurface and may contain zero or more polygon surfaces described by the intersection points of the surface and the edges of the cubes, these points being the vertices of the surface segment.

We now introduce the concept of equivalence between curfaces. Two curfaces are strongly equivalent if a vertex in one curface is on and the same vertex in the other curface is on. Figure 7 shows two curfaces that are strongly equivalent. The surface vertices are positioned on the same edges of the cube but the exact position is different.

Two curfaces are weakly equivalent if one cube can be mapped to the other cube in a way that makes them strongly equivalent. The allowed mappings are rotations of the curface along the cubes axis of symmetry. In this paper we only consider three different axes of symmetry in the cube. These are the axes that penetrate the center of each face orthogonal to the face. Figure 8 shows two cubes that are weakly

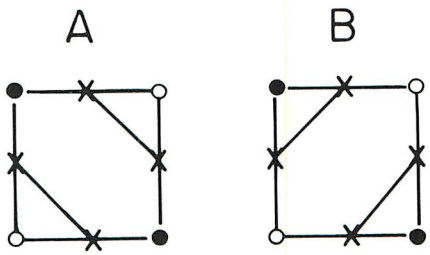


Figure 6 We have chosen to maximize the area of the face inside the isosurface to resolve the ambiguity noted in case D of Figure 5.

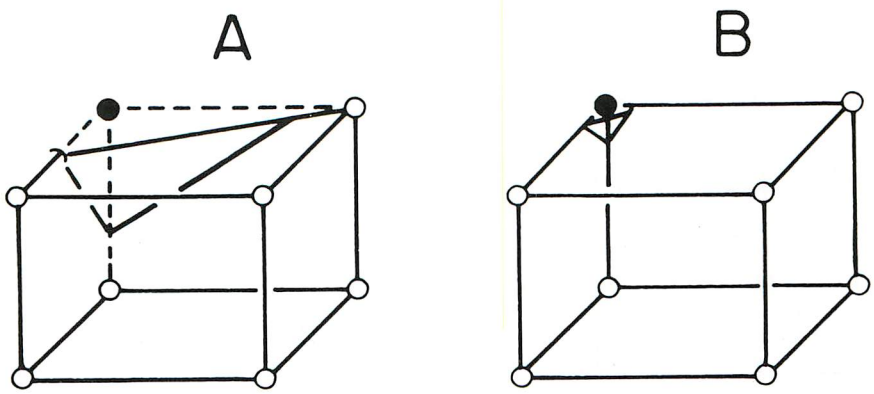


Figure 7 This diagram illustrates strong equivalence of two curfaces.

equivalent. Curface A is rotated 90 degrees along the indicated axis making it strongly equivalent to curface B. Loosely speaking, weakly equivalent means that two curfaces contain the same "kind" of isosurface.

There are several interesting points worth noting. There are 24 ways to map a cube onto itself without mirroring the cube. Any of these mappings can be achieved by no more than four 90 degree rotations along the cube's axis of symmetry. This is a well known result of group theory (specifically the description of the hexaeder group). The concept of equivalence introduces two equivalence relations on the set of all curfaces. There are 256 different classes of strongly equivalent curfaces. This is easily seen as there are eight vertices in a cube and each can have one of two values giving 2^8=256 different classes. Determining the number of weakly equivalent curfaces is more subtle. By expanding the results of Lorenson and Cline [1, 2], we can infer that there must be 23 classes but we have no formal proof of this. By enumeration it is clear that the result is correct, but in order to generalize these results to higher dimensions a more formal approach is needed.

Lorenson and Cline [2] described another concept equivalent to our concept of weak equivalence. But instead of allowing only rotations of the curface they also allow inversion of the on/off attribute of the curface vertices. This reduces the number of equivalence classes to 15 but reintroduces the already resolved ambiguity. There is no guarantee that surface segment vertices on the cube face shown in Figure 9 will be as in case A. It might just as well be as in case B. The final surface constructed from the individual surface segment is generally incoherent as Baker [3] points out. An example of such an incoherent surface constructed with the original algorithm by Lorenson and Cline is shown in Figure 9.

2.3 *Marching the Cubes*

The preceding discussion can be used in describing an algorithm for creating the isosurface. We have made a detailed study of the 23 weakly equivalent curfaces. By rotation we can map these to the 256 strongly equivalent curfaces. Then, by using interpolation the exact position of the polygon surface vertices can be calculated giving the curface.

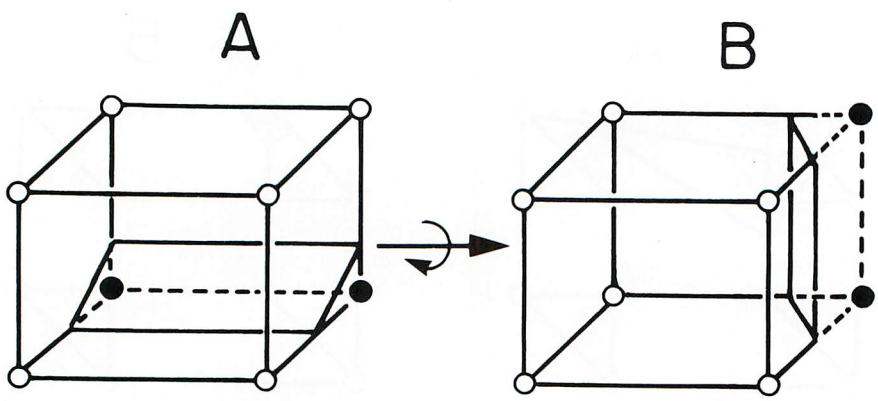


Figure 8 This diagram illustrates a case where two cubes are weakly equivalent.

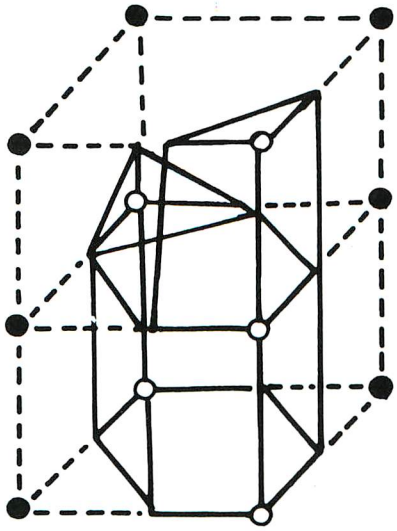


Figure 9 This is an example of an incoherent surface constructed with the original algorithm by Lorenson & Cline.

Our extension to the Marching Cube algorithm is now as follows:

1) Divide the volume into adjacent cubes. Process these cubes one at a time in sequence.

2) Classify the vertices of the cube as being either on or off (inside or outside the desired isosurface).

3) Find the strong equivalence class of the cube. This is easily done by creating an 8-bit binary number from the on/off state of the vertices of the cube. Each vertex corresponds to a single bit in the number. This number will be the number of the equivalence class and is termed the signature of the curface.

4) Get a polygon list from a table describing the surface segment for each strong equivalence class. This list will be zero or more sequences of polygon vertices positioned on the edges of the cube.

5) Calculate the exact position of the surface segment vertices by interpolation as described earlier.

6) For lighting purposes calculate the normal at each surface segment vertex.

This normal vector is the normalized gradient of the volume at that point. This normal vector is orthogonal to the isosurface intersecting that surface segment vertex.

Note that we only need to find the strong equivalence class. To create the table of polygon lists that is used in step 4 we require knowledge about the weak equivalence classes, but once this table has been created we no longer need to consider weak equivalence.

2.4 *Creating the Lookup Table*

The algorithm for constructing the table of polygon intersection list is as follows:

1) Step through a list of the 23 weak equivalence classes one at a time.

2) For each weak equivalence class pick a curface representing the class.

3) Apply all 24 possible rotations (including the identity) to the curface. The result will be a list of 23*24=552 curfaces.

4) Thin the list of curfaces so there is only one curface representing each strong equivalence class. This is done by calculating the signature of the curface.

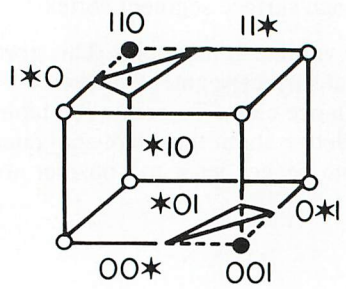
We now have a table of curfaces, each one representing a different strong equivalence class. For each strong equivalence class we will also have a curface. If this were not the case, it would be an indication that there are more than 23 weak equivalence classes. Thus, by performing this algorithm and checking that we get all the strong equivalence classes we prove—in a very inelegant manner (exhaustive enumeration)—that there are no more than 23 weak equivalence classes.

Finally, there is one step in the above algorithm we need to discuss in more detail. How do we rotate a curface? This involves the definition of the representation for curfaces and rotations.

Curfaces consist of a cube with vertices either on or off and a surface segment. First consider the vertices: We require a list of the vertices that are on the surface. This will be a list of length 0 to 8. We represent each vertex as a three-digit binary number. Imagine the cube placed in a three-dimensional right handed Cartesian coordinate system in the all-positive octant, with one vertex located at origin and the symmetry axis of the cube aligned with the x-, y- and z-axis. The scaling of the coordinate axes is such that any vertex will have coordinates of either 0 or 1. The representation of a vertex is now simply the x-, y- and z-coordinate of the vertex combined to create the three-digit binary number: ZYX.

A surface segment is a polygon with the vertices on the edges of the cubes, so we need a scheme to represent edges. This is done by a special three-digit ternary number the digits being 0, 1 and *. There is a special rule that has to be adhered to. The * should appear at one and only one position in the number. The possible count of numbers is $3\times2^2=12$, the number of cube edges. The interpretation of the number is as follows: any cube edge will be aligned with one of the coordinate axis, say the x-axis. Points on this edge can have any x-coordinate between 0 and 1 so X=*, but they have a fixed y- and z-coordinate, say Y=1 and Z=0. The resulting number is then ZYX=01*.

The surface segment is then an ordered list of cube edges. Each edge gives us information on where the surface segment vertex should be positioned and the ordering of the edges gives the sequence in which the vertices are connected. The last



Vertices: 001 110
Edges: 00* *01 0*1 - *10 11* 1*0

Figure 10 A curface and its corresponding ternary representation are shown.

vertex in a surface segment polygon should be connected to the first vertex to close the shape. Also, some surface segments consist of several non-connected surfaces so we need a delimiter to separate these surfaces to insure they are not connected (we use a dash). Finally, note that a surface segment can be void. A curface and its corresponding ternary representation is seen in Figure 10.

As mentioned above, it is possible to do any of the required rotations of the curface by doing no more than four 90 degree rotations around the coordinate axes. In fact, we only need to do right hand rotations. A single right-handed 90 degree rotation around an axis will be represented by the letter of that axis.

Compound rotations are then represented by a sequence of letters. The rotation zx consists of a rotation around the z-axis followed by a rotation around the x-axis. For example the rotation yyyy is the indentity and xxx is a left-hand 90 degree rotation around the x-axis. Also, note that xy and yx are different rotations—rotations generally do not commute.

We now introduce the extended negation operator~acting on the set {0, 1, *}. We define ~0=1, ~1=0, ~*=*. This enables us to describe the rotation of a vertex or edge represented by ZYX. We have x(ZYX)=Y~ZX, y(ZYX)=~XYZ, z(ZYX)=ZX~Y.

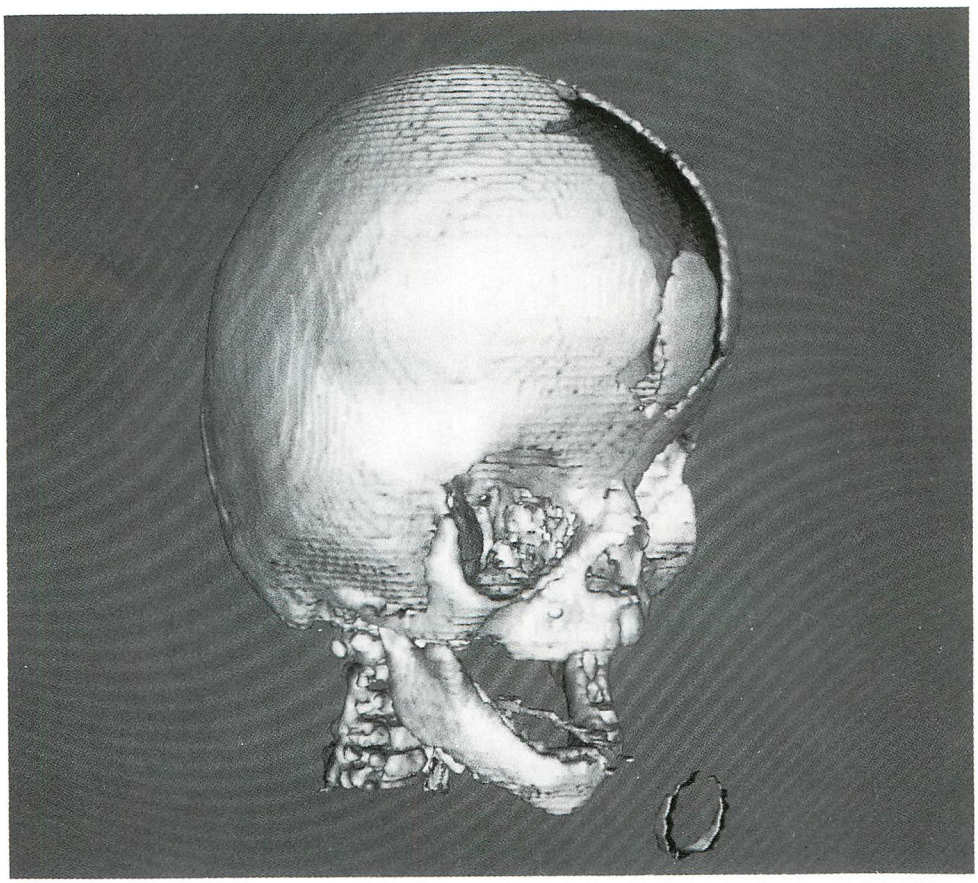


Figure 11 3D surface reconstruction from a CT study of a nine-month-old child with skull deformity due to Apert's syndrome. This is a right frontal oblique bony surface view.

It is now a simple matter to do any rotation of a curface. Simply apply the above described rotation to both the vertices and the edges.

DISCUSSION OF THE IMPLEMENTATION

There are a few additional comments about the implementation of the algorithm worthy of note.

As already mentioned by Lorenson and Cline [2], it is possible to optimize the algorithm by re-using already calculated values of the exact surface segment vertices positions and the normals at these points. This is true because curfaces stacked together will also fit together. Experimentation with several different data sets have indicated that more than 40% of the normals and 75% of the vertices can be re-used.

The algorithm easily lends itself to parallelization. The algorithm examines each cube separately and the results obtained do not depend on any other cube. Probably, the best thing to do is to divide the volume into smaller volumes that are each processed by a separate CPU.

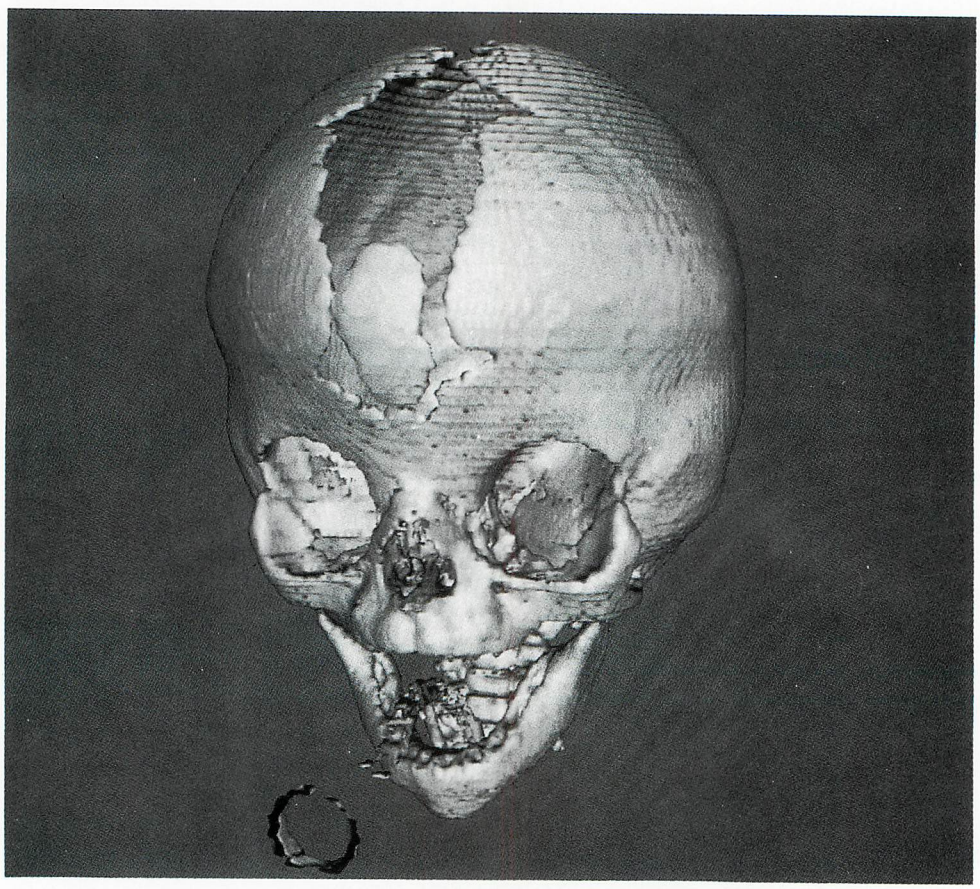


Figure 12 This is a frontal bony surface view from the same patient as Figure 11.

Our algorithm has been designed to generate non-planar polygons that are easily displayed on a Silicon Graphics 4D/340 computer. It is of course possible to triangulate the non-planar polygons to create a triangle representation of the surface and this only needs to be done for each of the 23 weak equivalence classes of curfaces.

Also, the algorithm may create singular polygons with all vertices at the same point. If this is a problem in the continued processing of the data, precautions to avoid output of singular polygons should be taken.

4 RESULTS

We have applied the algorithm to a CT study of the head of a nine-month-old child with a skull deformity due to Apert's syndrome. The 98 axial slices are 2 mm thick, with slice dimension of 256*256 pixels and pixel dimensions of approximately 0.8 mm. Two thresholds were selected to extract the surfaces of the bone and of the soft tissue. In the first case 209,169 polygons were created described by 837,644 vertices and in the second case 144,370 polygons described by 578,306 vertices. We have studied the distribution of the curface weak equivalence classes within the data. The table shows the percentage of curfaces for each weak equivalence class found in the surface. The numbers are rounded to the nearest hundreth of a percent. An ambiguous curface is a curface where the arbitrary choice mentioned earlier affects the description of the surface segment within the curface.

Weak eq.	Bone soft %	Soft tissue %	Ambiguous class
1	13.10	12.50	
2	15.49	15.52	
3	0.12	0.12	yes
4	0.02	0.02	
5	9.54	9.34	
6	0.13	0.13	yes
7	0.00	0.00	yes
8	23.16	23.67	
9	2.93	2.56	
10	0.09	0.11	yes
11	0.14	0.10	
12	0.06	0.12	yes
13	0.00	0.00	yes
14	0.13	0.07	
15	8.95	9.04	
16	0.17	0.21	yes
17	0.00	0.00	yes
18	14.08	14.99	
19	0.09	0.06	yes
20	0.05	0.15	
21	11.74	11.27	

The results show that a certain set of weak equivalence classes are almost never encountered and all the ambiguous curfaces are within this set. Assuming that our study is representative, this implies that incoherencies introduced by the original Marching Cubes algorithm and the asymmetries introduced by our algorithm are negligible in a CT scan surface model of the head.

Figures 11 and 12 show rendering of the bony surface extracted from scan data

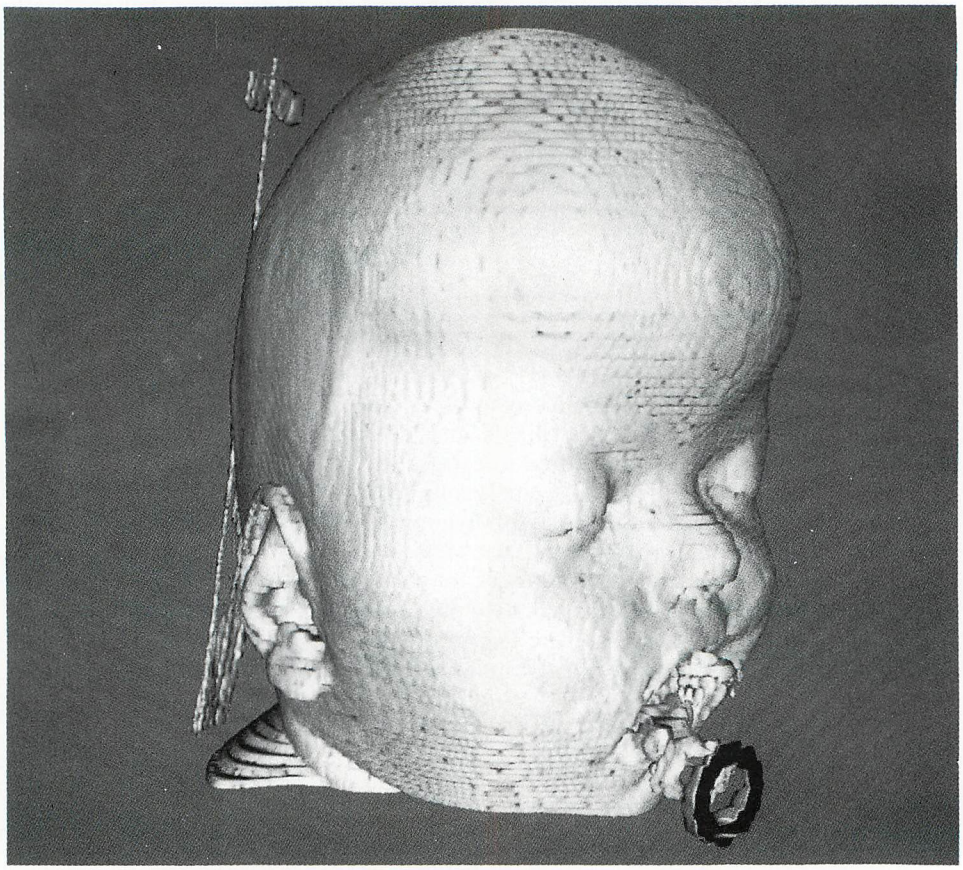


Figure 13 This is the soft tissue surface which corresponds to the bony surface noted in Figure 11. The soft tissues are extracted by lowering the threshold used for isosurface definition.

using the algorithm described. Figures 13 and 14 show rendering of the soft tissue extracted from the same scan data.

5 IMPROVEMENTS

In order to eliminate the asymmetry in our algorithm, we propose the following improvement. It is the decision about which surfaces to create in certain cases (recall Figure 9) that introduces the asymmetry. The decision need not be arbitrary but can instead be based on the voxel values on the face of the cube. As pointed out by Wallin and Gerig [4], this problem can be solved in the following way. The value of the sub-voxel in the center of the face can be calculated by averaging the four voxel values at the face vertices. If this voxel is inside the isosurface, the surface segment will be as Figure 9 case A, and otherwise it will be as in case B. Doing this does not affect the way curfaces stack together. The created surface will still be coherent, but this improvement complicates the algorithm.

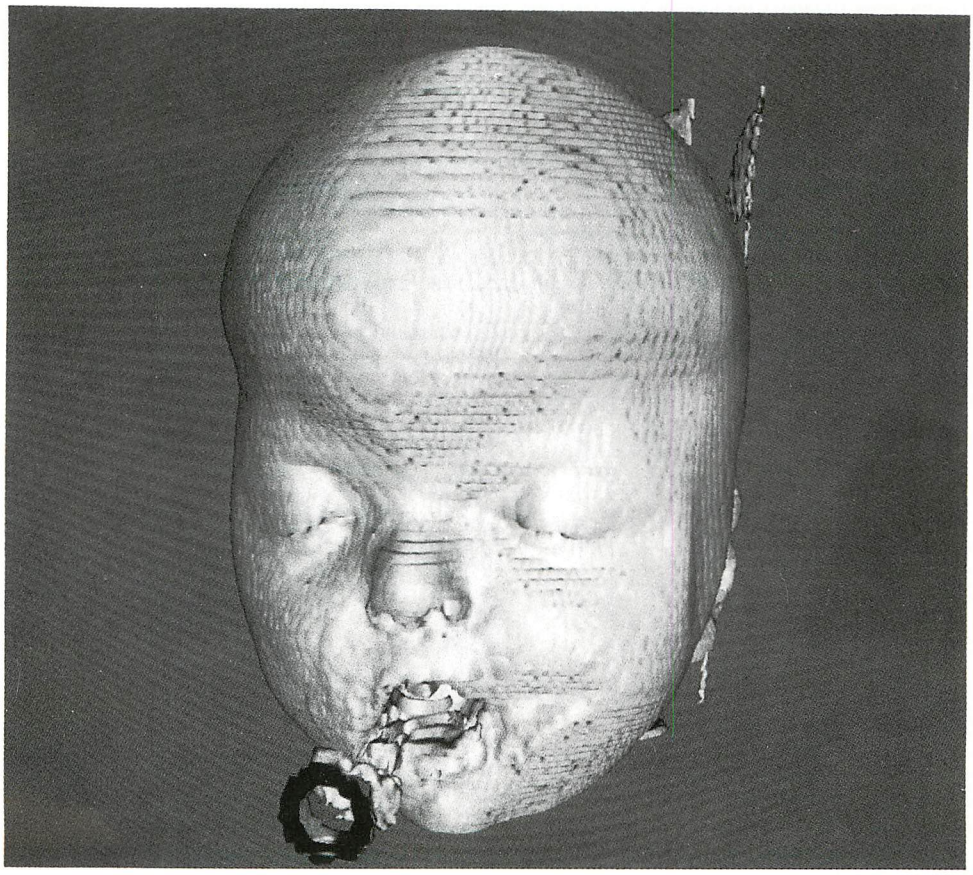


Figure 14 This is the soft tissue surface corresponding to the same orientation as the bony surface shown in Figure 12.

6 CONCLUSIONS

The Marching Cube method of Lorenson and Cline is a simple and efficient algorithm for creating isosurfaces from volume data. As we have shown, the original algorithm by Lorenson and Cline [1, 2] will create incoherent surfaces. This is solved in a way that introduces an asymmetry in the surface description. We consider that both the incoherency and the asymmetry is undesirable, but a case study shows that this has very little impact on the whole surface.

The possibility of generalizing the Marching Cubes algorithm to higher dimensions remains to be explored. Our detailed discussion of the 3D Marching Cubes can be useful in developing this extension.

ACKNOWLEDGEMENTS

The authors wish to thank Dr Sven Kreiborg of the Royal Dental College Copenhagen for his generous support and encouragement of this work. This project was supported, in part, by the National Institutes of

Health, grant DE09890, the Dannin Research Fund, Schoildorn's Research Fund, and the Gongsted Research Fund. Manuscript preparation by Mary Akin is gratefully appreciated.

REFERENCES

1. W. E. Lorenson and H. E. Cline, Marching cubes: A high resolution 3D surface construction algorithm, *Computer Graphics*, **21,** July (1987).
2. H. E. Cline, W. E. Lorenson *et al.*, Two algorithms for the three-dimensional reconstruction of tomograms, *Med. Phys.*, **15,** May/June (1988).
3. H. H. Baker, Building surfaces of evolution: The weaving wall, *Int. J. Computer Vision*, **3,** 51–71 (1989).
4. A. A. Wallin and G. Gerig, Automatic construction of iso-surfaces from volume data, *4th International Symposium on Spatial Data Handling*, Zurich (1990).
5. J. Wilhelms and A. Van Gelder, Topological considerations in isosurface generation, extended abstract, *Computer Graphics*, **24,** 79–86 (1990).