Approximating Normals for Marching Cubes applied to Locally Supported Isosurfaces

Gregory M. Nielson
Adam Huang
Steve Sylvester

Computer Science and Engineering
Arizona State University, Tempe AZ 85287-5406

Abstract

We present some new methods for computing estimates of normal vectors at the vertices of a triangular mesh surface approximation to an isosurface which has been computed by the marching cube algorithm. These estimates are required for the smooth rendering of triangular mesh surfaces. The conventional method of computing estimates based upon divided difference approximations of the gradient can lead to poor estimates in some applications. This is particularly true for isosurfaces obtained from a field function, which is defined only for values near to the isosurface. We describe some efficient methods for computing the topology of the triangular mesh surface, which is used for obtaining local estimates of the normals. In addition, a new, one pass, approach for these types of applications is described and compared to existing methods.

Additional Keywords: isosurface, normal vectors, marching cubes, triangular mesh, topology, Gouraud shading, approximation

1. INTRODUCTION AND MOTIVATION

Isosurfaces are very useful and important tools for the visualization and analysis of volume data. Typically, samples or measurements over a regular 3D lattice of a field function, $F(x,y,z)$, are known and the isosurface, S_T, is defined implicitly by

$$S_T \equiv \{(x,y,z): F(x,y,z)=0\}.$$

It is often the case that isosurfaces are approximated by a surface comprised of triangles. The Marching Cubes algorithm [6] is a very popular and effective means to compute this type of approximation. In addition, many rendering and post processing techniques require estimates of the normal vector of the isosurface at the vertices of the triangulated surface approximation. In this paper we present some new methods for computing these estimates. In this present section, we cover some background material and motivate the new techniques presented in Section 2 and Section 3. Examples of our new methods are presented in Section 4.

IEEE Visualization 2002 Oct. 27 - Nov. 1, 2002, Boston, MA, USA
0-7803-7498-3/02/$17.00 © 2002 IEEE

In general, there are two approaches to computing estimates of the normal, N(x,y,z), of an isosurface.

1. The first is based upon the fact that the normal vector is the gradient of the field function. That is

$$N(x,y,z)=\nabla F(x,y,z)=\begin{pmatrix}\frac{\partial F}{\partial x}(x,y,z)\\ \frac{\partial F}{\partial y}(x,y,z)\\ \frac{\partial F}{\partial z}(x,y,z)\end{pmatrix}. \quad (1)$$

Numerical differentiation approximations are then used to approximate the partial derivatives of (1). The method suggested in [6] is based upon a well known, second order, approximation of Zucker. First, the following approximation to the gradient is used to compute approximations to the normals at the lattice points:

$$N(i,j,k)=\begin{pmatrix}\frac{F(i+1,j,k)-F(i-1,j,k)}{2\Delta x}\\ \frac{F(i,j+1,k)-F(i,j-1,k)}{2\Delta y}\\ \frac{F(i,j,k+1)-F(i,j,k-1)}{2\Delta z}\end{pmatrix},\ \bar{N}(i,j,k)=\frac{N(i,j,k)}{\|N(i,j,k)\|} \quad (2)$$

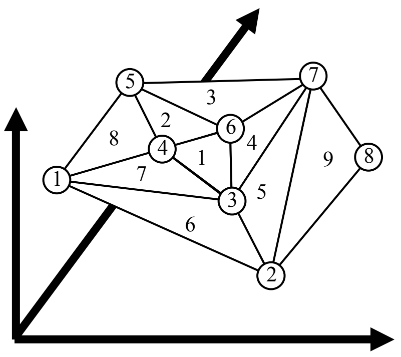
Next, linear interpolation along edges is used to obtain approximations to the normals at the vertices of the triangular mesh isosurface. We should point out that even though the approximation of (2) only involves two values of the field function, it is really a second order approximation. It is obtained by passing a quadratic function through three consecutive points and then taking the derivative of this quadratic at the center point. Due to the symmetry, the middle term drops out except on the boundaries. The appropriate formulas for the boundaries are easily obtained by the quadratic interpolation procedure:

$$N(1,n,k)=\begin{pmatrix}\frac{-F(3,n,k)+4F(2,n,k)-3F(1,n,k)}{2\Delta x}\\ \frac{F(1,n-2,k)-4F(1,n-1,k)+3F(1,n,k)}{2\Delta y}\\ \frac{F(1,n,k+1)-F(1,n,k-1)}{2\Delta z}\end{pmatrix}$$

Based upon the above ideas, it is very easy to implement a single pass marching cubes algorithm that produces or immediately renders a list of triple vertices with normal vectors. This is a standard simple approach, which avoids computation of the topology of the triangular mesh isosurface**.**

2. The second approach is more general and appeals more directly to the definition that a normal vector is perpendicular to

the surface. Approximations are based upon the normal vectors of local planar surface approximations. This approach requires the topology of the triangular mesh surface. A triangular grid structure as defined in Fig. 1 is a popular and convenient method for representing the topological information for a triangular mesh surface. An estimate of the normal at a vertex can be computed as a weighted average of the normals of all the triangles which involve this vertex. The weights may be chosen to be uniform or based upon the area of the associated triangle or even the subtended angle.



Triangles

n_i	n_j	n_k
3	4	6
4	5	6
6	5	7
3	6	7
2	3	7
2	1	3
1	4	3
1	5	4
2	7	8

Fig. 1. An example that serves to define a triangular grid structure for representing a triangular mesh surface.

Now we proceed with the motivation for the new methods that we will eventually describe in Sections 2 and 3. When implementing a mc algorithm, it is a relatively easy matter to include normal estimates based upon the gradient approach and the divided difference approximations given in (2). This approach does not require computing the topology of the triangular grid structure. But, in some applications, these types of approximations may be rather poor. The types of applications that have motivated the methods of this paper are those where the field function is only defined at the lattice points that are very close to the vertices of the isosurface. Here, the isosurface or points on or near the isosurface are given and the field function is determined so as to yield the isosurface. Some methods of point cloud fitting would fall into this category [4]. Also applications that use a signed distance function (or modifications thereof) to represent surfaces have the potential of having this locally supported property. In these types of applications, it is possible for the gradient estimates to involve lattice points, which are not necessary to precisely define the isosurface, and so these values may not even be defined by the algorithm producing the field function. We illustrated this problem further in the two-dimensional case with the diagram of Fig. 2.

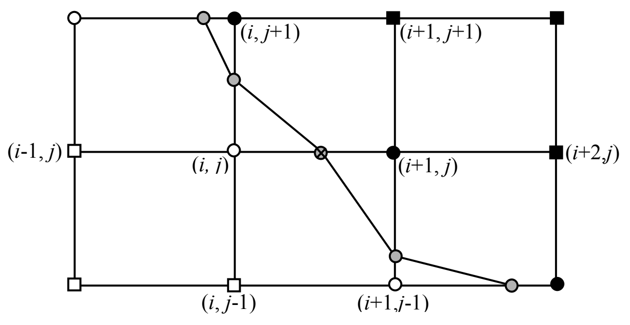


Fig. 2. Diagram showing problems with gradient estimate methods.

The values of the field function at the lattice points marked with white and black circles are selected so as to obtain an approximation to a given isocurve. The value of the field function at the lattice points marked with black and white squares are not necessarily specified and may simply take on some default value such as +1 or –1 or even zero in some applications. The labelled lattice points affect the approximation of the normal at the isocurve point between (i, j) and $(i+1, j)$:

$$N(x, j) \cong (i+1-x)\begin{pmatrix} \dfrac{F(i+1, j)-F(i-1, j)}{2} \\ \dfrac{F(i, j+1)-F(i, j-1)}{2} \end{pmatrix} + (x-i)\begin{pmatrix} \dfrac{F(i+2, j)-F(i, j)}{2} \\ \dfrac{F(i+1, j+1)-F(i+1, j-1)}{2} \end{pmatrix} \quad (3)$$

As can be observed from (3), the values of the field function at $(i+1, j+1)$, $(i+2, j)$, $(i-1, j)$ and $(i, j-1)$ do not affect the isocurve, yet they would be used in the Zucker/gradient calculation of the normal.

2. COMPUTING THE TRIANGULAR GRID TOPOLOGY AS PART OF THE MARCHING CUBES ALGORITHM

We use the notation defined in Fig. 3 where $V_0 = (0,0,0)$, $V_4 = (0,0,1)$, $V_1 = (1,0,0)$, $V_5 = (1,0,1)$, $V_2 = (0,1,0)$, $V_6 = (0,1,1)$, $V_3 = (1,1,0)$, $V_7 = (1,1,1)$.

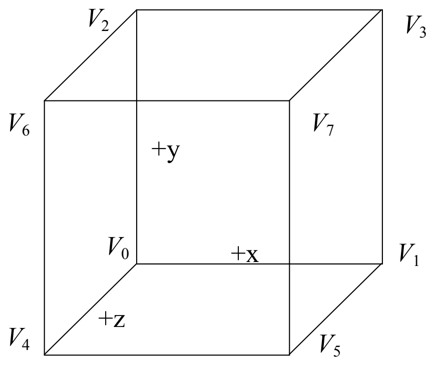


Fig. 3. The unit cube and notation.

The general approach of the mc algorithm is to first characterize a collection of representative configurations to which all possible cases can be mapped. A fixed triangulation is determined for each unique representative configuration and this triangulation is inferred on an arbitrary case by the inverse mapping. A decision has to be made as to what type of maps are allowed and what properties are to be used for characterizing representatives. We use only rotation maps. In the interest of reducing the number of representatives, some authors have used a richer collection of maps including also mirroring maps and complementation (with respect to above or below the isosurface threshold). While this is possible, special caution must be used so as not to produce erroneous results as was the case for "hole" problem of the original mc method (see [6] and [7]).

If the value at one vertex on an edge is above the threshold and the value the other endpoint of the edge is below the threshold, then we know by linear interpolation that there is a point on the isosurface on this edge. This leads to a total of 256 (= 2^8) cases. Many of these cases are equivalent in the sense that one can be rotated to the other. Under the operation of rotation maps these 256 cases collect themselves into a much smaller set of equivalence classes. There are 23 distinct equivalence classes. A representative of an equivalence class is called a *configuration*. In Fig. 4 we show triangulations of these 23 configurations. Here, we have arbitrarily taken all ambiguous faces to be separated [7].

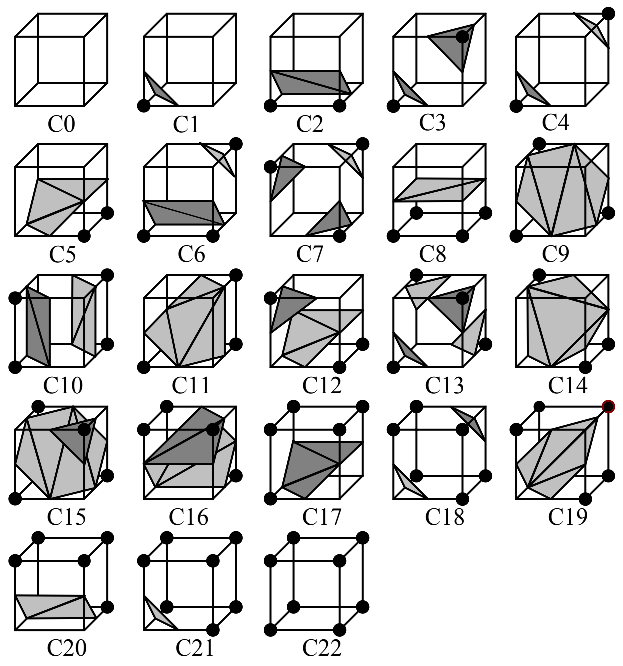


Fig. 4. Triangulations of the configurations of the mc algorithm.

There are 24 distinct rotations of a cube that map the 8 corner vertices onto themselves. In the following table we give the vertex permutations for each of these 24 rotations. The vertex permutations denote which vertex gets mapped to which position. For example, with R6, V_0 maps to V_1, V_1 maps to V_5, … , V_7 maps to V_6. These rotation maps form a group. Any composition (one rotation followed by another) of two of these rotations will result in one of the other rotations and any rotation has an inverse.

Table 1

Rotation group for the cube

Index	**7 6 5 4 3 2 1 0**	Index	**7 6 5 4 3 2 1 0**
0	7 6 5 4 3 2 1 0	12	1 3 5 7 0 2 4 6
1	3 2 7 6 1 0 5 4	13	0 4 2 6 1 5 3 7
2	1 0 3 2 5 4 7 6	14	5 1 7 3 4 0 6 2
3	5 4 1 0 7 6 3 2	15	4 6 0 2 5 7 1 3
4	6 2 4 0 7 3 5 1	16	6 7 2 3 4 5 0 1
5	2 3 0 1 6 7 4 5	17	7 5 3 1 6 4 2 0
6	3 7 1 5 2 6 0 4	18	7 3 6 2 5 1 4 0
7	5 7 4 6 1 3 0 2	19	3 1 2 0 7 5 6 4
8	4 5 6 7 0 1 2 3	20	1 5 0 4 3 7 2 6
9	6 4 7 5 2 0 3 1	21	4 0 5 1 6 2 7 3
10	2 0 6 4 3 1 7 5	22	0 2 1 3 4 6 5 7
11	0 1 4 5 2 3 6 7	23	2 6 3 7 0 4 1 5

Each entry into the case/configuration table consist of a configuration which serves as the representer of the equivalence class to which this case belongs along with the rotation which makes them equivalent. For example, the entry for case 68 (0100 0100) would consist of rotation R13 and configuration C2. This means that case 68 will be rotated to configuration C2 with rotation R13. The triangles produced by Case 68 would be the same as those produced by configuration C2 except that the indices are replaced by the vertex permutations. That is, rather than the two triangles ($P_{4\text{-}6}$, $P_{0\text{-}4}$, $P_{1\text{-}5}$) and ($P_{4\text{-}6}$, $P_{1\text{-}5}$, $P_{5\text{-}7}$) the triangles ($P_{6\text{-}4}$, $P_{7\text{-}6}$, $P_{3\text{-}2}$) and ($P_{6\text{-}4}$, $P_{3\text{-}2}$, $P_{2\text{-}0}$) are produced because of the vertex replacement rules given by the vertex permutations of Rotation R13. This is done in the following manner. When we see $P_{1\text{-}5}$, for example, we ask what got mapped to V_1 (answer is V_3) and what got mapped to V_5 (answer is V_2) and so we replace $P_{1\text{-}5}$ with $P_{3\text{-}2}$. Here is another example. If we apply R4 to case 148, we get configuration C7. Configuration C7 says to produce the triangles: ($P_{4\text{-}6}$, $P_{6\text{-}7}$, $P_{2\text{-}6}$), ($P_{4\text{-}5}$, $P_{1\text{-}5}$, $P_{5\text{-}7}$), and ($P_{1\text{-}3}$, $P_{2\text{-}3}$, $P_{3\text{-}7}$). But rather, we use the vertex permutation rules and produce the triangles: ($P_{0\text{-}2}$, $P_{2\text{-}6}$, $P_{3\text{-}2}$), ($P_{0\text{-}4}$, $P_{5\text{-}4}$, $P_{4\text{-}6}$), and ($P_{5\text{-}7}$, $P_{3\text{-}7}$, $P_{7\text{-}6}$).

Table 2

Case/configuration table

msb|lsb	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0000	0:0	3:1	5:1	3:2	2:1	15:2	2:3	16:5	14:1	19:3	17:2	15:5	2:2	17:5	3:5	3:8
0001	0:1	4:2	20:3	6:5	10:3	7:5	11:7	0:9	0:4	20:6	22:6	0:11	16:6	7:14	3:12	10:17
0010	6:1	3:3	6:2	0:5	2:4	15:6	6:6	0:14	6:3	2:7	23:5	4:9	2:6	17:12	9:11	11:17
0011	0:2	5:5	4:5	0:8	11:6	6:11	4:12	2:17	0:6	5:12	4:14	14:17	9:10	1:16	8:16	1:20
0100	1:1	4:3	1:4	5:6	13:2	20:5	13:6	7:11	1:3	5:7	17:6	15:12	8:5	3:9	3:14	1:17
0101	7:2	18:5	7:6	6:14	19:5	7:8	19:12	21:17	10:6	18:12	0:10	9:16	14:11	22:17	12:16	9:20
0110	7:3	8:7	21:6	0:12	18:6	20:12	1:10	14:16	0:7	1:13	23:12	4:15	8:12	3:15	4:16	0:19
0111	12:5	1:9	1:11	13:17	12:14	9:17	23:16	14:20	12:12	1:15	19:16	18:19	3:16	12:19	3:18	8:21
1000	8:1	3:4	12:3	3:6	18:3	19:6	1:7	16:12	14:2	23:6	9:5	9:14	13:5	3:11	8:9	12:17
1001	0:3	4:6	3:7	6:12	4:7	7:12	0:13	0:15	14:6	3:10	9:12	18:16	13:12	21:16	8:15	7:19
1010	9:2	12:6	22:5	4:11	9:6	2:10	22:12	10:16	21:5	21:12	9:8	19:17	13:14	7:16	18:17	7:20
1011	1:5	1:14	5:9	8:17	1:12	17:16	5:15	1:19	10:11	13:16	20:17	13:20	5:16	1:18	4:19	1:21
1100	1:2	8:6	1:6	7:10	14:5	14:14	14:12	0:16	2:5	2:12	13:11	11:16	2:8	4:17	5:17	0:20
1101	11:5	12:11	11:12	2:16	2:9	23:17	2:15	6:19	2:14	6:16	15:16	2:18	0:17	6:20	3:19	6:21
1110	10:5	10:12	10:14	16:16	2:11	22:16	20:16	0:18	11:9	11:15	7:17	10:19	6:17	20:19	4:20	0:21
1111	1:8	3:17	17:17	2:20	15:17	17:20	19:19	14:21	16:17	2:19	15:20	2:21	3:20	5:21	3:21	0:22

If we are planning to produce a triangular grid structure for the isosurface rather than simply a collection of triple vertices, then the notation used for labeling the edges shown in Fig. 5 and Fig. 6 is helpful. These unique labels serve as pointers to the vertices of the triangular mesh surface. In fact the actual vertices do not need to be computed during this phase of the algorithm. If we use this type of approach, then, rather than the vertex permutations of the rotations, we need the edge mappings as given in Tabel 3.

To see how this works, we look again at one of the previous examples. This is the example where R4 is applied to case 148 to yield configuration C7. Configuration C7 says to produce the triangles: (001Y, 011X, 010Z), (001X, 100Z, 101Y) and (100Y, 010X, 110Z). But rather, we use the edge mapping rules and produce the triangles: (000Y, 010Z, 010X), (000Z, 001X, 001Y) and (101Y, 110Z, 011X) which for an arbitrary voxel *i,j,k* we would get the triangles (*i,j,k*.Y, *i,j*+1,*k*.Z, *i,j*+1,*k*.X), (*i,j,k*.Z, *i,j,k*+1.X, *i,j,k*+1.Y), (*i*+1,*j,k*+1.Y, *i*+1,*j*+1,*k*.Z, *i,j*+1,*k*+1.X).

A portion of voxel grid is shown in Fig. 7 and in Table 4 there is a fragment of the triangular grid for the resulting isosurface from the mc algorithm.

We note that the following triangulations were assumed in creating Table 4 and Fig. 7. C1: (001X, 001Y, 000Z); C5: (100Y, 101Y, 000Y), (101Y, 001X, 000Z), (101Y, 000Z, 000Y); C14: (100Y, 101Y, 001X), (100Y, 001X, 010Z), (001X, 000Z, 010Z), (100Y, 010Z, 010X); C3: (001Y, 000Z, 001X), (011X, 101Y, 110Z).

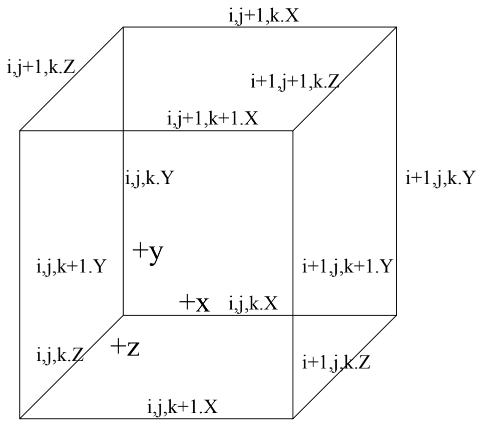


Fig. 5. Edge labeling for voxel i,j,k.

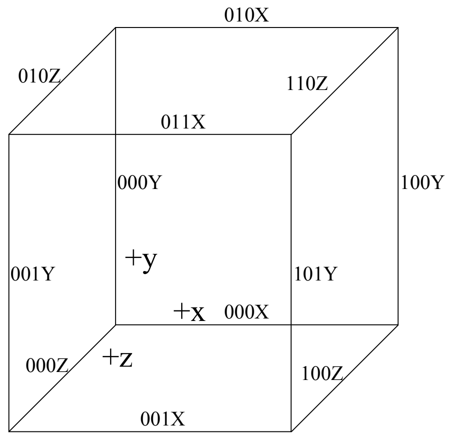


Fig. 6. Edge labeling for generic voxel.

Table 3

Edge Mapping Table

	Edge Mappings											
R0	000X	010X	001X	011X	000Y	100Y	001Y	101Y	000Z	100Z	010Z	110Z
R1	001X	000X	011X	010X	000Z	100Z	010Z	110Z	001Y	101Y	000Y	100Y
R2	011X	001X	010X	000X	001Y	101Y	000Y	100Y	010Z	110Z	000Z	100Z
R3	010X	011X	000X	001X	010Z	110Z	000Z	100Z	000Y	100Y	001Y	101Y
R4	100Z	110Z	000Z	010Z	100Y	101Y	000Y	001Y	000X	001X	010X	011X
R5	001X	011X	000X	010X	101Y	001Y	100Y	000Y	100Z	000Z	110Z	010Z
R6	000Z	010Z	100Z	110Z	001Y	000Y	101Y	100Y	001X	000X	011X	010X
R7	000Y	100Y	001Y	101Y	010X	000X	011X	001X	010Z	000Z	110Z	100Z
R8	010X	000X	011X	001X	100Y	000Y	101Y	001Y	110Z	010Z	100Z	000Z
R9	100Y	000Y	101Y	001Y	000X	010X	001X	011X	100Z	110Z	000Z	010Z
R10	101Y	100Y	001Y	000Y	100Z	110Z	000Z	010Z	001X	011X	000X	010X
R11	011X	010X	001X	000X	110Z	010Z	100Z	000Z	101Y	001Y	100Y	000Y
R12	001Y	000Y	101Y	100Y	010Z	000Z	110Z	100Z	011X	001X	010X	000X
R13	110Z	100Z	010Z	000Z	101Y	100Y	001Y	000Y	011X	010X	001X	000X
R14	010Z	000Z	110Z	100Z	000Y	001Y	100Y	101Y	010X	011X	000X	001X
R15	100Y	101Y	000Y	001Y	110Z	100Z	010Z	000Z	010X	000X	011X	001X
R16	000X	001X	010X	011X	100Z	000Z	110Z	010Z	100Y	000Y	101Y	001Y
R17	000Y	001Y	100Y	101Y	000Z	010Z	100Z	110Z	000X	010X	001X	011X
R18	000Z	100Z	010Z	110Z	000X	001X	010X	011X	000Y	001Y	100Y	101Y
R19	001Y	101Y	000Y	100Y	001X	011X	000X	010X	000Z	010Z	100Z	110Z
R20	010Z	110Z	000Z	100Z	011X	010X	001X	000X	001Y	000Y	101Y	100Y
R21	110Z	010Z	100Z	000Z	010X	011X	000X	001X	100Y	101Y	000Y	001Y
R22	101Y	001Y	100Y	000Y	011X	001X	010X	000X	110Z	100Z	010Z	000Z
R23	100Z	000Z	110Z	010Z	001X	000X	011X	010X	101Y	100Y	001Y	000Y

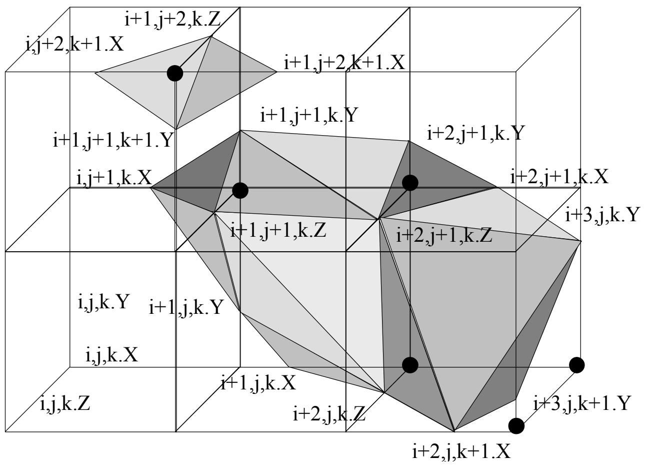


Fig. 7. Portion of voxel grid with isosurface approximation.

Table 4
A fragment of the triangular grid representation of the isosurface of fig. 7

List of triple pointers defining the triangular grid structure			Notes
. . .	. . .	. . .	
i+1,j+1,k.Z	i+1,j,k.Y	i,j+1,k.X	i,j,k; case 8, R14:C1
i+2,j+1,k.Z	i+2,j,k.Z	i+1,j+1,k.Z	i+1, j,k, case 14, R3:C5
i+2,j,k.Z	i+1,j,k.X	i+1,j,k.Y	
i+2,j,k.Z	i+1,j,k.Y	i+1,j+1,k.Z	
i+3,j,k.Y	i+3,j,k+1.Y	i+2,j,k+1.X	i+2, j, k case 39, R0:C14
i+3,j,k.Y	i+2,j,k+1.X	i+2,j+1,k.Z	
i+2,j,k+1.X	i+2,j,k.Z	i+2,j+1,k.Z	
i+3,j,k.Y	i+2,j+1,k.Z	i+2,j+1,k.X	
. . .	. . .	. . .	
i+1,j+2,k.Z	i,j+2,k+1.X	i+1,j+1,k+1.Y	i, j+1, k case 130, R12:C3
i+1,j+1,k.Y	i+1,j+1,k.Z	i,j+1,k.X	
. . .	. . .	. . .	

3. NON-GRADIENT NORMAL ESTIMATES WITHOUT TRIANGULAR GRID TOPOLOGY

The basic idea of this method is to compute an estimate using the vertices of the isosurface that are connected to P through an edge of the triangles of the isosurface, except that we ignore points such as the one of Fig. 8 on the edge joining the voxel vertices $(i+1, j-1, k)$ and $(i+1, j-1, k+1)$ since this join to P is arbitrary and depends upon the particular triangulation of the isosurface used for each configuration. Except on the boundary voxels, there will always be exactly four vertices that connect to P with an edge on a voxel face. We denote these points as Px-, Px+, Py-, Py+. The point Px+ is in the same X-Z plane as P (i.e. has the same y coordinate) and has an x-coordinate that is greater than

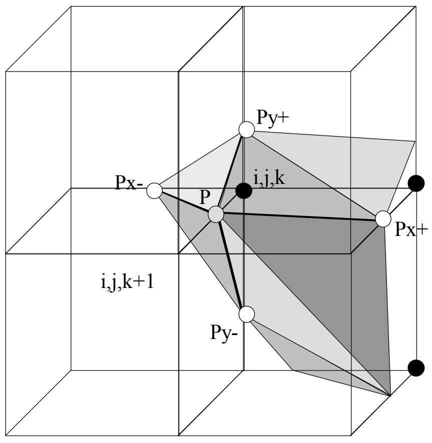


Fig. 8. Notation.

that of P. The point Px- is also in the same X-Z plane as P and has an x-coordinate that is less than that of P. The point Py- has the same x-coordinate as P and a y-coordinate, which is less than P. The point Py+ has a y-coordinate greater than P. These five points are then used to compute estimates of the normal at P. There are several reasonable and viable possibilities. In order to focus the attention on the basic ideas of our new method, we will only include results for the following, which we have found to be a good overall approach:

$$N_{++} = \text{Normal of triangle: (P, Py+, Px+)}$$
$$N_{-+} = \text{Normal of triangle: (P, Px-, Py+)}$$
$$N_{--} = \text{Normal of Triangle: (P, Py-, Px-)}$$
$$N_{+-} = \text{Normal of Triangle: (P, Px+, Py-)}$$
$$N_P = N_{++}/\|N_{++}\| + N_{-+}/\|N_{-+}\| + N_{--}/\|N_{--}\| + N_{+-}/\|N_{+-}\|$$
$$N_P = N_P/\|N_P\|$$

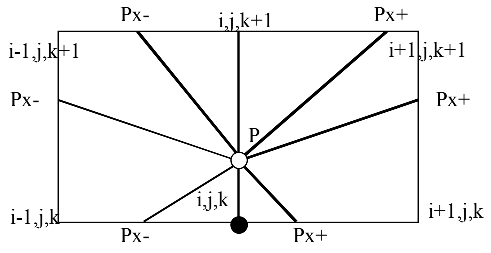


Fig. 9. Illustrating the computation of Px- and Px+.

The vertices Px- and Px+ are computed by linear interpolation along the edge that contains them. The appropriate edge is determined as follows:

```
If $F_{i-1,j,k} < \alpha$ then Px- $\in$ [i,j,k to i-1,j,k]
    else ( if $F_{i-1,j,k+1} < \alpha$ then Px- $\in$ [i-1,j,k+1 to i-1,j,k]
        else Px- $\in$ [i-1,j,k+1 to i,j,k+1])
If $F_{i+1,j,k} < \alpha$ then Px+ $\in$ [i,j,k to i+1,j,k]
    else ( if $F_{i+1,j,k+1} < \alpha$ then Px+ $\in$ [i+1,j,k+1 to i+1,j,k]
        else Px+ $\in$ [i+1,j,k+1 to i,j,k+1] )
```

For the points Py- and Py+ the process is entirely similar except we work in the other plane containing the edge of P; namely the plane with the same y-coordinate as P.

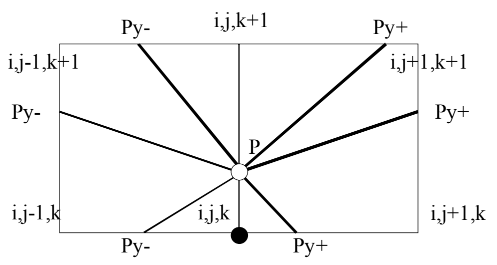


Fig. 10. An aid to the computation of Py+ and Py-.

```
If $F_{i,j-1,k} < \alpha$ then Py- $\in$ [i,j,k to i,j-1,k]
    else ( if $F_{i,j-1,k+1} < \alpha$ then Py- $\in$ [i,j-1,k to i,j-1,k+1]
        else Py- $\in$ [i,j-1,k+1 to i,j,k+1] )
If $F_{i,j+1,k} < \alpha$ then Py+ $\in$ [i,j,k to i,j+1,k]
    else ( if $F_{i,j+1,k+1} < \alpha$ then Py+ $\in$ [i,j+1,k to i,j+1,k+1]
        else Py+ $\in$ [i,j+1,k+1 to i,j,k+1] )
```

In the above discussion we have assumed that $F_{i,j,k} > \alpha$ and $F_{i,j,k+1} < \alpha$. The other case, where $F_{i,j,k} < \alpha$ and $F_{i,j,k+1} > \alpha$, is similar but we simply reverse the role of ">" and "<".

The other two cases where P is on an edge from i,j,k to i+1,j,k and P is on an edge from i,j,k to i,j+1,k are completely analagous to the above. Both lead to four additional points, which along with P are used for the local estimation of the normal vector. The cases where P is on a boundary edge are exceptional and there will only be three or two additional points used for the approximation. With the use of a marking flag and/or more elaborate data structures or labelling (as in Section 2), it is possible to compute only once (or even delay the computation) of the neighboring vertices leading to a more efficient implementation. However, we have found that a naïve implementation that computes vertices on demand often runs sufficiently fast for real time application. We have found this to be the case for field function resolutions < 200^3 on a gigahertz PC.

4. EXAMPLES

Example 1.

Our first example is based upon a point cloud of data values. A surface is fit to this point cloud. In this approach a field function is found so that its zero level isosurface is the approximating surface. We do not describe the method of fitting as the details are not of interest here, but it is similar in spirit to the methods of [1] but with different basis functions [4]. So that we can test our new method, we choose the point cloud from a known surface. The surface is the graph of the function

$$z = G(x,y) = \frac{-5.5xy + 2.5x + 2.5y - 1}{-8xy + 2.5x + 2.5y} \tag{4}$$

The data points and the true normals are shown in Fig. 11.a. In Fig.11.b and 11.c, we show the isosurface of a field function designed so as to yield the graph of (4) as an isosurface. The method is intended to only give an approximation, but in this case the results are quite good and it would be difficult to distinguish from the graph of (4). In Fig. 11.d, we show the results of the new method described in Section 3. The results of the method of Section 2 are indistinguishable from those of Fig. 11.d. In Fig. 11.e we have the results of the Zucker/gradient method described in the introductory section. These results are poor, particularly near the "neck" of the graph. The purpose of the image of Fig. 11.f is to show that the field function is locally supported. We change the threshold a little and the isosurface changes dramatically. The algorithm that produced this isosurface was only concerned with ensuring that the graph of (4) was the zero threshold isosurface and lattice values, which do not affect this, are not of much concern. Since we can compute the true value of the normals from the test function, we can compare the RMS error for the three methods:

Table 5

RMS Errors of true normal verses approximation

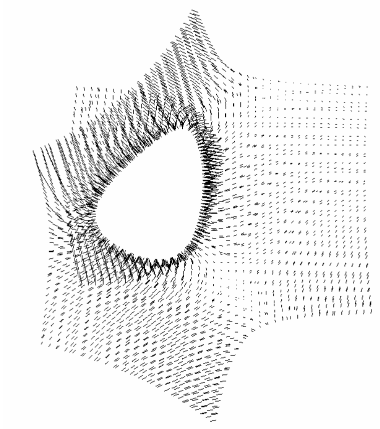
Zucker/gradient Method	0.043
Local average method of Section 2	0.009
New method of Section 3	0.013

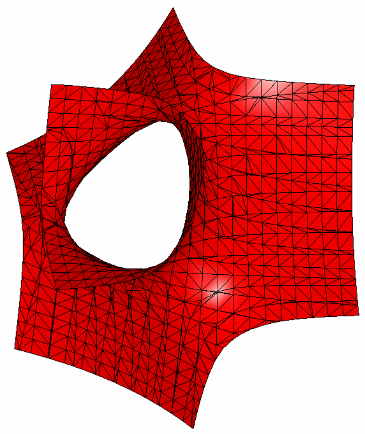
Example 2.

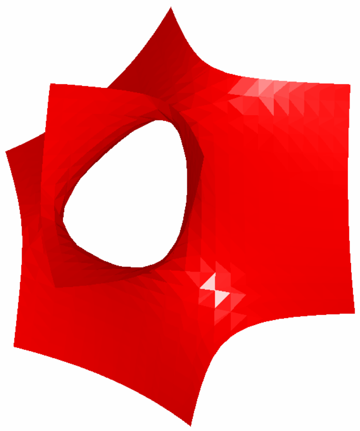
This example is similar to the previous one except that the point cloud is obtained by scanning an object. An LDI scanner was used. It works similar to a conventional digital camera except that we also get a pixel array of distance values. Various scans were taken and later registered. The data is rather noisy due somewhat to the method of registration. In Fig. 12a, a photo of the actual object is shown. In Fig. 12.b the data is shown overlaid on a surface so that it can be perceived better. Fig. 12.c shows a rendering based upon the Zucker/gradient approach to computing normals and in Fig. 12.d we have a rendering based upon the normals computed by the new method of Section 3. Again, we do not show the local average method of Section 2 as it is indistinguishable from the image of Fig. 12.d. As in Example 1, we see that the same dark flaws in the Zucker/gradient method, particularly behind the left ear and on the face of the left bust support. These artefacts are not present in the rendering based upon the new method of Section 3.

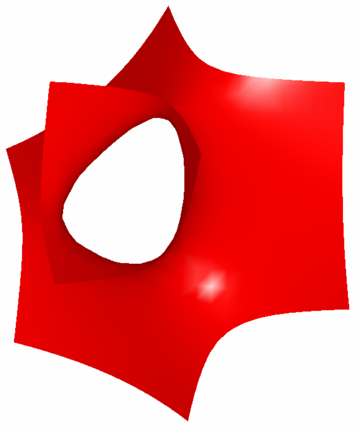
Example 3.

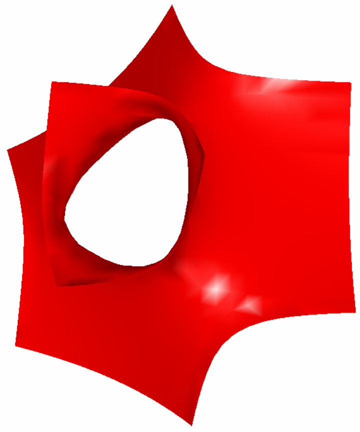
For this example, a surface is given and the field function is determined so as to yield this specified surface as an isosurface. In this application, the field function is the signed distanced to the surface. The field function is only computed at lattice points close to the surface. Once the field function is determined, it is an easy matter to apply triple tensor product wavelet operators to obtain multiresolution approximations. Since it is not clear how to use wavelet approximations directly for general triangular mesh surfaces, this simple approach is rather appealing. Here the Daubechies wavelets, D_4, are used (see [2] and [3]). We do not further explain these approximations because that is not the point of this example in this context. The point is to illustrate how the conventional Zurcker/gradient approach can give bad results for some applications and our new method gives much improved results at approximately the same computational cost. In Fig. 13, we show three levels of approximation to a given surface. The top images use the Zucker/gradient method for computing normals and the bottom images use our new method as described in Section 3. The difference and consequently the improvement is more pronounced at the higher resolution models, but there are significant improvements at all levels.


a. Point Cloud Data with Normals


b. Isosurface Grid


c. Constant Shading


d. Normals by New Method of Sec. 3


e. Zucker/Gradient Method

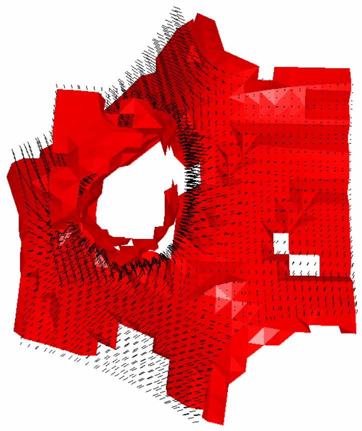
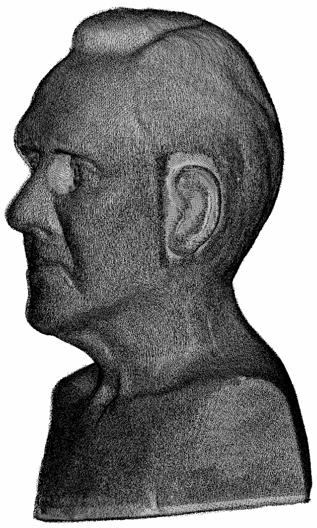
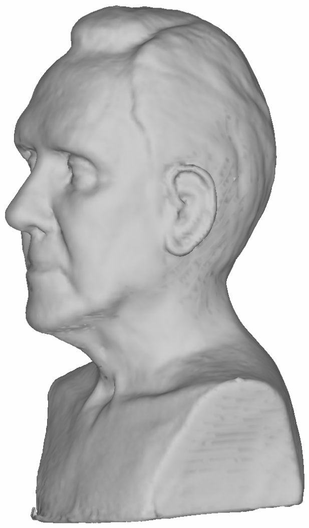

f. Isosurface Threshold on 1.0

Fig. 11. Example 1. An isosurface with local support showing the problems with the Zucker/gradient method and the improvements of the new method of Section 3.

a. Actual Object


b. Data Overlaid on Surface


c. Zucker/Gradient Normals

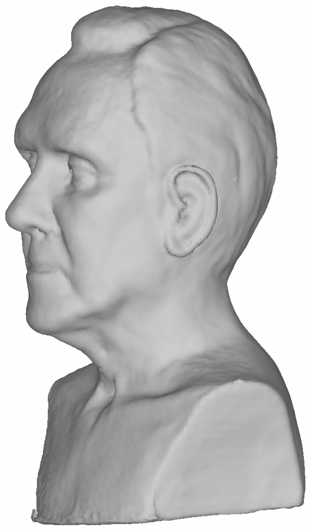

d. New Method for Normals

Fig. 12. Example 2.

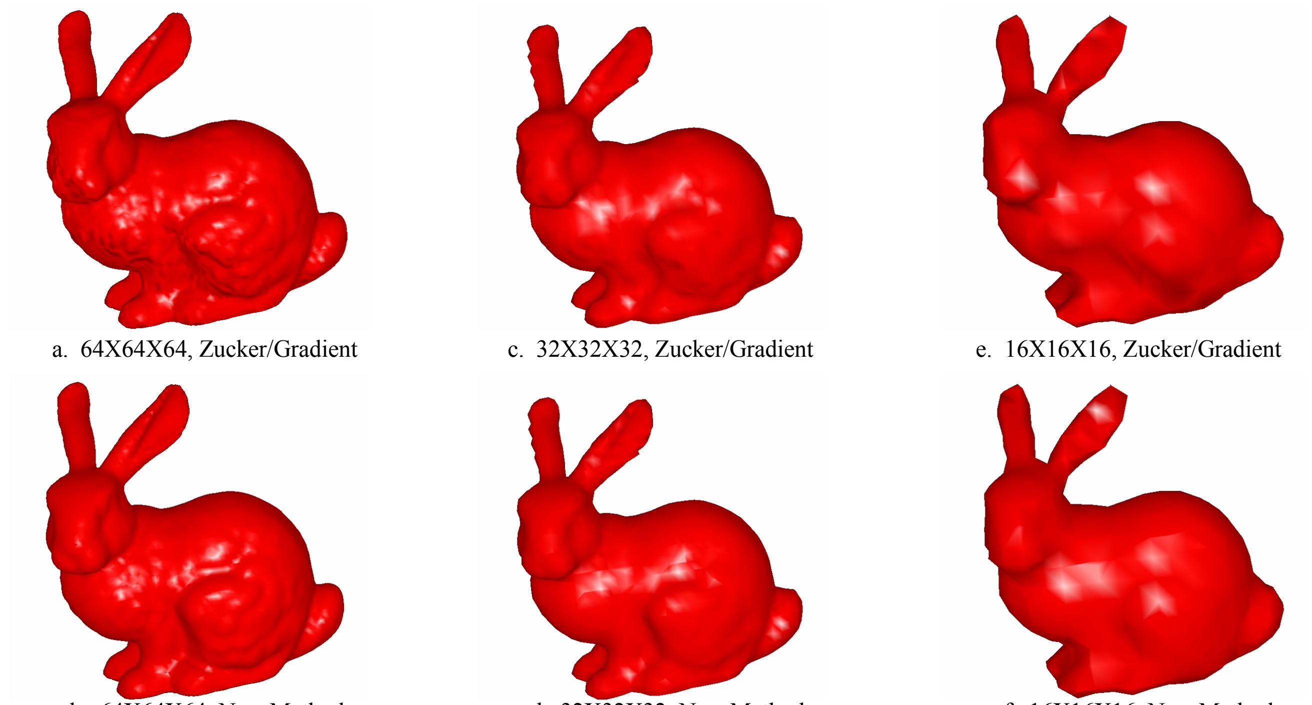

a. 64X64X64, Zucker/Gradient
c. 32X32X32, Zucker/Gradient
e. 16X16X16, Zucker/Gradient
b. 64X64X64, New Method
d. 32X32X32, New Method
f. 16X16X16, New Method

Fig. 13. Example 3. Surface defined as isosurface of signed distance function and wavelet approximations applied to the field function and the subsequent isosurface extracted.

5. CONCLUSIONS

We have presented new methods for computing the estimates of normal vectors for the vertices of a triangular mesh surface approximation to an isosurface computed by the mc algorithm. The methods are designed for applications where the conventional Zucker/gradient methods can give poor results such as the case where the field function is only defined at lattice points near to the isosurface. Our new methods are easy to implement, efficient and give good results. Our new methods only work in the context of the mc algorithm and therefore should not be confused with other methods for estimating normals, for example, methods that estimate the normals of point cloud data [5]. In Section 2, we described a method for computing the triangular grid topology as part of the mc algorithm. In Section 3, we described a method with results similar to that of Section 2, but without the explicitly computing the triangular grid topology. In this paper, we have focused on estimating normals for Gouraud shading and our interest in the topology provided by the triangular grid structure of Section 2 is based upon this. But there is certainly independent interest in computing this topology and so the techniques of Section 2 should have much broader interests than the topic of focus for this paper.

Acknowledgements

We wish to acknowledge the support of the Office of Naval Research (N00014-97-1-0243 & N00014-00-1-0281 & N00014-02-1-0287), the National Science Foundation (NSF IIS-9980166 & ACI-0083609) and DARPA (MDA972-00-1-0027). The bunny data is from Stanford.

References

[1] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum & T. R. Evans, Reconstruction and Representation of 3D Objects with Radial Basis Functions, SIGGRAPH 2001, pp. 67-76.

[2] I. Daubechies, Orthonormal bases of compactly supported wavelets, Comm. Pure Appl. Math., Vol. 41 (1988), pp. 909-996.

[3] M. Gross, Integrated Volume Rendering and Data Analysis in Wavelet Space, in: Scientific Visualization (Nielson, Hagen & Meuller eds.), IEEE Computer Society Press, pp. 149-178, 1997.

[4] H. Hagen, Th. Schreiber and G. Brunnett, Generating Triangular Surface Meshes from Large Sets of Unorganized 3D Points, to be published in Numerisation 3D.

[5] H. Hoppe, et al., Surface reconstruction from unorganized points, SIGGRAPH '92, pp. 123-143.

[6] W. E. Lorensen and H. E. Cline, Marching cubes: A high resolution 3D surface construction algorithm, SIGGRAPH '87 Proceedings, Vol. 21, 1987, pp. 163-169.

[7] G. M. Nielson and B. Hamann, The Asymptotic Decider: Resolving the Ambiguity in Marching Cubes, Proc. IEEE Visualization 1991, pp. 83-91, Oct. 1991.