CHAPTER 2

MARCHING CUBES AND VARIANTS

In the introduction, we mentioned four different approaches to isosurface construction. In this chapter, we describe one of those approaches to isosurface construction, the widely used MARCHING CUBES algorithm by Lorensen and Cline [Lorensen and Cline, 1987a].

The MARCHING CUBES algorithm is based on two ideas. First, the isosurface can be constructed piecewise within each cube of the grid without reference to other grid cubes. Second, the combinatorial structure of each isosurface patch in a grid cube can be retrieved from a lookup table. Since the main operation is retrieving this structure from the lookup table, the algorithm runs in time proportional to the number of grid cubes.

We first present a two-dimensional version of the algorithm, called MARCHING SQUARES, for constructing two-dimensional isocontours. Before discussing the MARCHING SQUARES algorithm, we define some terminology that will be used by the algorithms in this chapter.

2.1 Definitions

Given a regular scalar grid and an isovalue σ, it is convenient to assign "+" and "−" labels to each grid vertex based on the relationship between its scalar value and σ.

Definition 2.1.

- A grid vertex is positive, "+", if its scalar value is greater than or equal to σ.
- A grid vertex is negative, "−", if its scalar value is less than σ.
- A positive vertex is strictly positive if its scalar value does not equal σ.

Since the scalar value of a negative vertex never equals the isovalue, there is no point in defining a similar "strictly negative" term.

Grid edges can be characterized by the labels at their endpoints.

Definition 2.2.

- A grid edge is positive if both its endpoints are positive.
- A grid edge is negative if both its endpoints are negative.
- A positive grid edge is strictly positive if both its endpoints are strictly positive.
- A grid edge is bipolar if one endpoint is positive and one endpoint is negative.

Note that a grid vertex or edge is only positive or negative in relationship to some isovalue.

The definitions given above apply not just to regular scalar grids but also to curvilinear grids. They also apply to the vertices and edges of polyhedral meshes such as tetrahedral and simplicial meshes.

2.2 Marching Squares

2.2.1 Algorithm

Input to the MARCHING SQUARES algorithm is an isovalue and a set of scalar values at the vertices of a two-dimensional regular grid. The algorithm has three steps. (See Figure 2.1.) Read in the isocontour lookup table from a pre-constructed data file. For each square, retrieve from the lookup table a set of

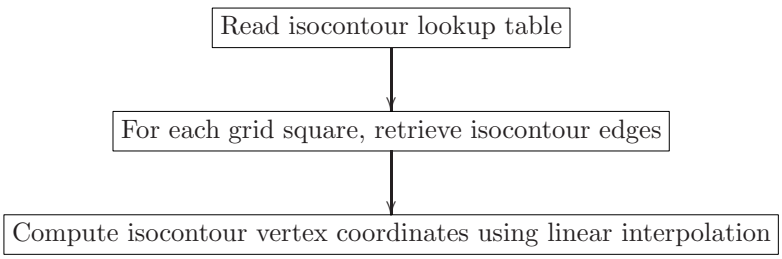


Figure 2.1. MARCHING SQUARES.

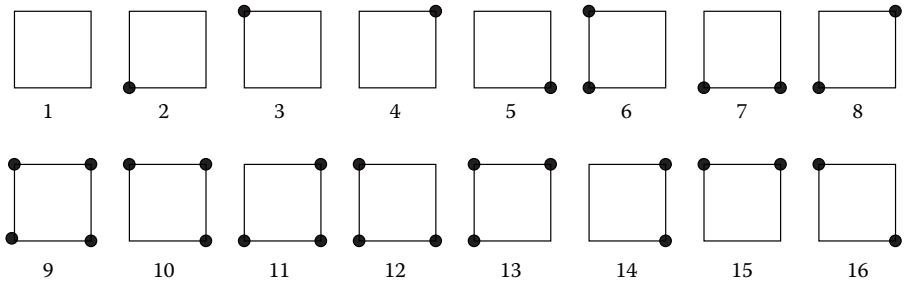


Figure 2.2. Square configurations. Black vertices are positive.

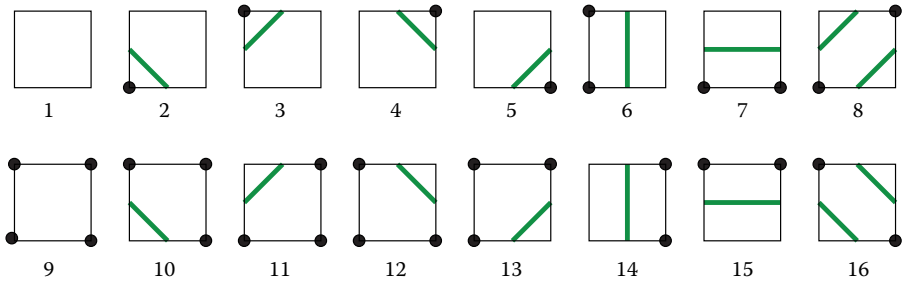


Figure 2.3. Square isocontours. Configurations 1 and 9 have no isocontour. Isocontours for configurations 2–7 and 10–15 are single line segments. Isocontours for configurations 8 and 16 are two line segments.

isocontour edges representing the combinatorial structure of the isocontour. The endpoints of these edges form the isocontour vertices. Assign geometric locations to the isocontour vertices based on the scalar values at the square edge endpoints. We explain the last two steps of the algorithm next.

Each grid vertex is labeled positive or negative as described in Section 2.1. (See Figure 2.4(b) for an example.) Since a square has four vertices, there are $2^4 = 16$ different configurations of square vertex labels. These configurations are listed in Figure 2.2.

The combinatorial structure of the isocontour within each square is determined from the configuration of the square's vertex labels. In order to separate the positive vertices from the negative ones, the isocontour must intersect any square edge that has one positive and one negative endpoint. An isocontour that intersects a minimal number of grid edges will not intersect any square edge whose endpoints are both strictly positive or whose endpoints are both negative.

For each square configuration κ, let $E_\kappa^{+/-}$ be the set of bipolar edges. Note that the size of $E_\kappa^{+/-}$ is either zero, two, or four. Pair the edges of $E_\kappa^{+/-}$. Each such pair represents an isocontour edge with endpoints on the two elements of the pair. Figure 2.3 contains the sixteen square configurations and their

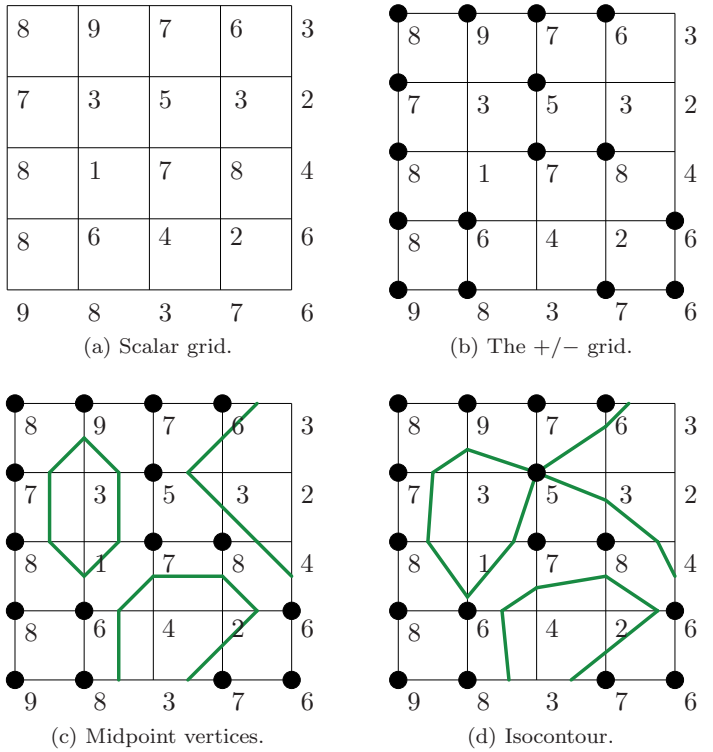


Figure 2.4. (a) 2D scalar grid. (b) Black vertices are positive. Vertex v with scalar value s_v is positive if $s_v >= 5$ and negative if $s_v < 5$. Note that $s_v = 5$ for one grid vertex v. (c) Isocontour with vertices at edge midpoints (before linear interpolation). (d) Isocontour with isovalue 5.

isocontours. The isocontour lookup table, Table, contains sixteen entries, one for each configuration. Each entry, Table$[\kappa]$ is a list of the $E_\kappa^{+/-}$ pairs.

In Figure 2.3 the isocontour edges are drawn connecting the midpoints of each square edge. This is for illustration purposes only. The geometric locations of the isocontour vertices are not defined by the lookup table.

The isocontour lookup table is constructed on the unit square with vertices $(0,0),(1,0),(0,1),(1,1)$. To construct the isocontour in grid square (i,j), we have to map pairs of unit square edges to pairs of square (i,j) edges. Each vertex $v = (v_x, v_y)$ of the unit square maps to $v + (i,j) = (v_x, v_y) + (i,j) = (v_x + i, v_y + j)$. Each edge $\mathbf{e}$ of the unit square with endpoints (v, v') maps to edge $\mathbf{e} + (i,j) = (v + (i,j), v' + (i,j))$. Finally, each edge pair $(\mathbf{e}_1, \mathbf{e}_2)$ maps to $(\mathbf{e}_1 + (i,j), \mathbf{e}_2 + (i,j))$.

The endpoints of the isocontour edges are the isocontour vertices. To map each isocontour edge to a geometric line segment, we use linear interpolation to

```
Input  : F is a 2D array of scalar values.
         Coord is a 2D array of (x, y) coordinates.
         σ is an isovalue.
Result : A set ϒ of isocontour line segments.

MarchingSquares(F, Coord, σ, ϒ)
Read Marching Squares lookup table into Table;
/* Assign "+" or "−" signs to each vertex                          */
foreach grid vertex (i, j) do
    if F[i, j] < σ then Sign[i, j] ← "−";
    else Sign[i, j] ← "+";                          /* F[i, j] ≥ σ */
end
S ← ∅;
/* For each grid square, retrieve isocontour edges                 */
foreach grid square (i, j) do
    /* Grid square vertices are (i, j), (i+1, j), (i, j+1), (i+1, j+1) */
    κ ← (Sign[i, j], Sign[i+1, j], Sign[i, j+1], Sign[i+1, j+1]);
    foreach edge pair (e1, e2) ∈ Table[κ] do
        Insert edge pair (e1 + (i, j), e2 + (i, j)) into S;
    end
end
/* Compute isocontour vertex coordinates using linear interpolation */
foreach bipolar grid edge e with endpoints (i1, j1) and (i2, j2) do
    /* Compute the isosurface vertex we on edge e                  */
    we ← LinearInterpolation
              (Coord[i1, j1], F[i1, j1], Coord[i2, j2], F[i2, j2], σ);
end
/* Convert S to set of line segments                               */
ϒ ← ∅;
foreach pair of edges (e1, e2) ∈ S do
    ϒ ← ϒ ∪ {(we1, we2)};
end
```

Algorithm 2.1. Marching Squares.

position the isocontour vertices as described in Section 1.7.2. Each isocontour vertex v lies on a grid edge $[p, q]$. If s_p and s_q are the scalar values at p and q and σ is the isovalue, then map v to $(1-\alpha)p + \alpha q$ where $\alpha = (\sigma - s_p)/(s_q - s_p)$. Note that since p and q have different signs, scalar s_p does not equal s_q and the denominator $(s_q - s_p)$ is never zero.

The Marching Squares algorithm is presented in Algorithm 2.1. Function `LinearInterpolation`, called by this algorithm, is defined in Algorithm 1.1 in Section 1.7.2.

Figure 2.4 contains an example of a scalar grid, an assignment of positive and negative labels to the grid vertices, the isocontour before linear interpolation, and the final isocontour after linear interpolation.

2.2.2 Running Time

The MARCHING SQUARES algorithm runs in linear time.

Proposition 2.3. *Let N be the total number of vertices of a 2D scalar grid. The running time of the* MARCHING SQUARES *algorithm on the scalar grid is* $\Theta(N)$.

Proof: Reading the MARCHING SQUARE lookup table takes constant time. Each grid square is processed once. At each grid square, at most two isocontour edges are retrieved from the lookup table. Since the number of grid squares is bounded by the number of grid vertices, determining the isocontour edges takes $O(N)$ time.

Computing the isocontour vertex on each grid edge takes time proportional to the number of isocontour vertices. Since each grid edge has at most one isocontour edge, the time to compute isocontour vertices is proportional the number of grid edges. The number of grid edges is less than twice the number of grid vertices, so the number of grid edges is at most $2N$. Thus computing the isocontour vertices takes $O(N)$ time.

The algorithm examines every grid square, so its running time has an $\Omega(N)$ lower bound. Thus, the running time of the MARCHING SQUARES algorithm is $\Theta(N)$. □

2.2.3 Isocontour Properties

To properly discuss the output produced by the MARCHING SQUARES algorithm, we need to differentiate between two cases based on the isovalue. In the first case, the isovalue does not equal the scalar value of any grid vertex. In this case, the MARCHING SQUARES algorithm produces a piecewise linear 1-manifold with boundary. The boundary of the 1-manifold lies on the boundary of the grid. In the second case, the isovalue equals the scalar value of one or more grid vertices. In this case, the MARCHING SQUARES algorithm may not produce a 1-manifold with boundary or the boundary may not lie on the boundary of the grid. For instance, the MARCHING SQUARES algorithm applied to the 3×3 grids in Figures 2.5 and 2.6 produces non-manifold isocontours or isocontours with boundary not on the scalar grid. In Figure 2.5(a), four isocontour line segments intersect at a single point; in Figure 2.5(b), the isocontour is a single point, and in Figure 2.6, the boundary of the isocontour lies inside the grid.

The two cases also differ in the nature of the line segments produced by the algorithm. The isocontour produced by the MARCHING SQUARES algorithm is

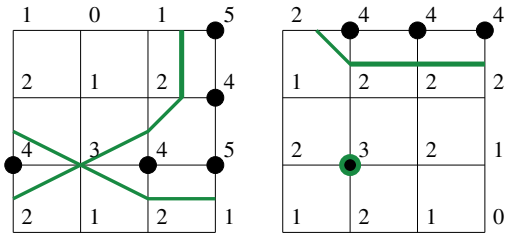


Figure 2.5. Examples of non-manifolds produced by MARCHING SQUARES (isovalue 3). Black vertices are positive. (a) Four curves joining at the grid vertex with isovalue 3. (b) Isosurface includes an isolated point at the grid vertex with isovalue 3.

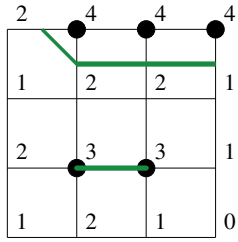


Figure 2.6. Examples of a manifold produced by MARCHING SQUARES whose boundary does not lie on the grid boundary (isovalue 3). Black vertices are positive.

a set of line segments whose vertices lie on the grid edges. If the isovalue does not equal the scalar value of any grid vertex, then these line segments all have positive length. If the isovalue equals the scalar value of one or more grid vertices, then the isocontour may have zero-length edges. For instance, the MARCHING SQUARES algorithm applied to the three grids in Figure 2.7 produces isocontours for isovalue 3 with zero-length edges.

In Figure 2.7(a), the lower-left grid square has configuration 4, producing a single isocontour edge, but both endpoints of that edge map to the vertex in the middle of the grid. In Figure 2.7(b), each grid square produces an isocontour edge, but all four edges have zero length and collapse to a single point. In Figure 2.7(c), leftmost and rightmost grid squares produce zero-length isocontour edges and two middle grid squares produce two duplicate isocontour edges on a grid edge.

MARCHING SQUARES returns a finite set, Υ, of line segments. The isocontour is the union of those line segments. The vertices of the isocontour are the endpoints of the line segments.

The following properties apply to all isocontours produced by the MARCHING SQUARES algorithm.

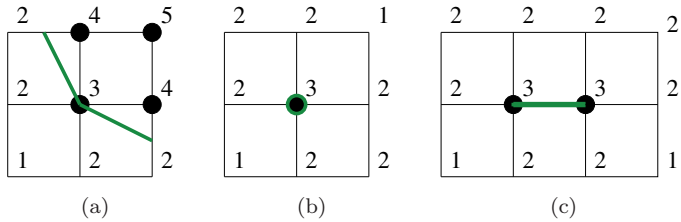


Figure 2.7. Examples of zero-length contour edges produced by MARCHING SQUARES (isovalue 3). Black vertices are positive. (a) Isocontour with one zero-length isocontour edge (from lower-left grid square). (b) Isocontour with four zero-length isocontour edges. (c) Another isocontour with four zero-length isocontour edges. Isocontour also has two duplicate nonzero isocontour edges (from the two middle grid squares).

Property 1. *The isocontour is piecewise linear.*

Property 2. *The vertices of the isocontour lie on grid edges.*

Property 3. *The isocontour intersects every bipolar grid edge at exactly one point.*

Property 4. *The isocontour does not intersect any negative or strictly positive grid edges.*

Property 5. *The isocontour separates positive grid vertices from negative grid vertices and strictly separates strictly positive grid vertices from negative grid vertices.*

Set $\mathbb{Y} \subseteq \mathbb{X}$ separates point $p \in \mathbb{X}$ from point $q \in \mathbb{X}$ if every path in $\mathbb{X}$ connecting p to q intersects $\mathbb{Y}$. Set $\mathbb{Y}$ strictly separates p from q if $\mathbb{Y}$ separates p from q and neither p nor q is on $\mathbb{Y}$. (See Section 1.7.1 and Appendix B.9.)

Properties 3 and 4 imply that the isocontour intersects a minimum number of grid edges. If both endpoints of a grid edge have scalar value equal to the isovalue, then the isocontour may intersect the grid edge zero, one, or two times or may contain the grid edge. (See Figure 2.8.)

A grid vertex may have scalar value equal to the isovalue and yet no isocontour passes through any edge containing that grid vertex. For instance, the MARCHING SQUARES algorithm returns the empty set when run on the scalar grid in Figure 2.9 with isovalue 3. Each vertex, including the center vertex, is positive, so each grid square has configuration 9 (Figure 2.2) and has no isocontour edges.

By Property 3, the isocontour intersects every bipolar grid edge. However, the bipolar grid edge may be intersected by zero-length isocontour edges as in Figure 2.7(b).

The following properties apply to MARCHING SQUARES isocontours whose isovalues do not equal the scalar value of any grid vertex.

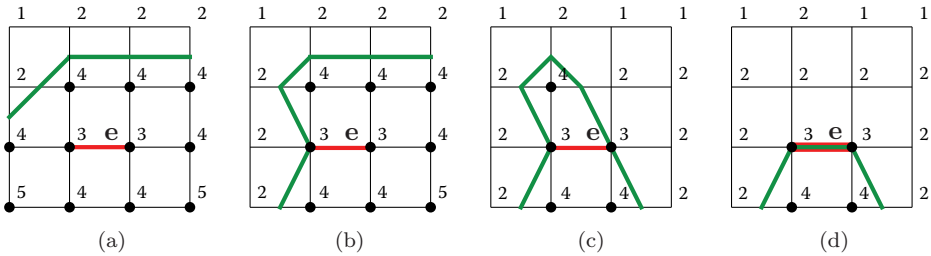


Figure 2.8. Examples of grid edges with both endpoint scalar values equal to the isovalue (3). Black vertices are positive. (a) Red grid edge **e** does not intersect the isocontour. (b) Red grid edge **e** intersects the isocontour at one endpoint. (c) Red grid edge **e** intersects the isocontour at both endpoints. (d) Red grid edge **e** is contained in the isocontour.

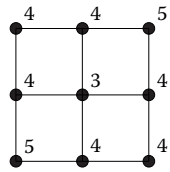


Figure 2.9. Example of a scalar grid whose MARCHING SQUARES isocontour is the empty set, even though the center grid vertex has scalar value equal to the isovalue 3. All vertices are positive.

Property 6. *The isocontour is a piecewise linear 1-manifold with boundary.*

Property 7. *The boundary of the isocontour lies on the boundary of the grid.*

Property 8. *Set Υ does not contain any zero-length line segments or duplicate line segments, and the line segments in Υ form a "triangulation" of the isocontour.*

The triangulation in Property 8 simply means that line segments in Υ intersect at their endpoints. The isocontour is one-dimensional and does not contain any triangles.

2.2.4 Proof of Isocontour Properties

We give a proof of each of the properties listed in the previous section.

Property 1. *The isocontour is piecewise linear.*

Property 2. *The vertices of the isocontour lie on grid edges.*

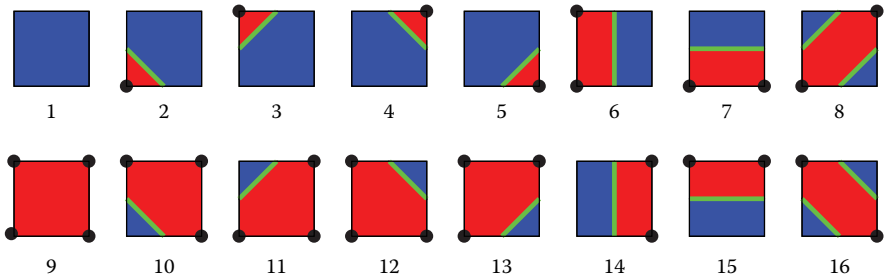


Figure 2.10. Red, positive regions and blue, negative regions for each square configuration. The green isocontour is part of the positive region. Black vertices are positive.

Proof of Properties 1 & 2: The MARCHING SQUARES isocontour consists of a finite set of line segments, so it is piecewise linear. These line segments intersect only at their endpoints and thus form a triangulation of the isocontour. The endpoints of these line segments lie on the grid edges, confirming Property 2. □

Property 3. *The isocontour intersects every bipolar grid edge at exactly one point.*

Property 4. *The isocontour does not intersect any negative or strictly positive grid edges.*

Proof of Properties 3 & 4: Each isocontour edge is contained in a grid square. Since the grid squares are convex, only isocontour edges with endpoints (vertices) on the grid edge intersect the grid edge. If the grid edge has one positive and one negative endpoint, the unique location of the isocontour vertex on the grid edge is determined by linear interpolation. Thus the isocontour intersects a bipolar grid edge at only one point.

If the grid edge is negative or strictly positive, then no isocontour vertex lies on the grid edge. Thus the isocontour does not intersect negative or strictly positive grid edges. □

Within each grid square the isocontour partitions the grid square into two regions. Let the positive region for a grid square $\mathbf{c}$ be the set of points which can be reached by a path ζ from a positive vertex. More precisely, a point p is in the positive region of $\mathbf{c}$ if there is some path $\zeta \subset \mathbf{c}$ connecting p to a positive vertex of $\mathbf{c}$ such that the interior of ζ does not intersect the isocontour. A point p is in the negative region of $\mathbf{c}$ if there is some path $\zeta \subset \mathbf{c}$ connecting p to a negative vertex of $\mathbf{c}$ such that ζ does not intersect the isocontour. Since any path $\zeta \subset \mathbf{c}$ from a positive to a negative vertex must intersect the isocontour, the positive and negative regions form a partition of the square $\mathbf{c}$. Figure 2.10 illustrates the positive and negative regions, colored red and blue, respectively, for each square configuration.

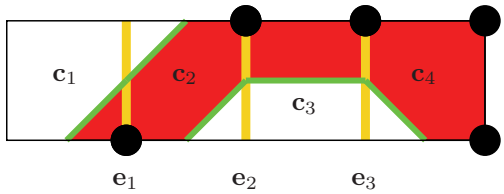


Figure 2.11. Adjacent grid squares, $\mathbf{c}_1$, $\mathbf{c}_2$, $\mathbf{c}_3$, and $\mathbf{c}_4$, and their positive (red) regions, $R^+_{\mathbf{c}_1}$, $R^+_{\mathbf{c}_2}$, $R^+_{\mathbf{c}_3}$ and $R^+_{\mathbf{c}_4}$, respectively. Yellow edges $\mathbf{e}_1$, $\mathbf{e}_2$ and $\mathbf{e}_3$ separate the squares. Positive regions agree on the grid square boundaries, i.e., $R^+_{\mathbf{c}_1} \cap \mathbf{e}_1 = R^+_{\mathbf{c}_2} \cap \mathbf{e}_1$ and $R^+_{\mathbf{c}_2} \cap \mathbf{e}_2 = R^+_{\mathbf{c}_3} \cap \mathbf{e}_2$ and $R^+_{\mathbf{c}_3} \cap \mathbf{e}_3 = R^+_{\mathbf{c}_4} \cap \mathbf{e}_3$.

Note the asymmetry in the definitions of the positive and negative regions. For the positive region the *interior* of ζ does not intersect the isocontour, while for the negative region the entire path ζ must not intersect the isocontour. Thus, the positive region contains the isocontour while the negative region does not. The positive region is also closed. Any point within the positive region that does not lie in the isocontour has a neighborhood contained in the positive region.

Every negative vertex is contained in the negative region since the zero-length path connects the vertex to itself. Similarly, every positive vertex is contained in the positive region.

Let $R^+_{\mathbf{c}}$ be the positive region for a grid square $\mathbf{c}$. We claim that positive and negative regions agree on the grid square boundaries. For instance, in Figure 2.11 $R^+_{\mathbf{c}_1} \cap \mathbf{e}_1$ equals $R^+_{\mathbf{c}_2} \cap \mathbf{e}_1$ where $R^+_{\mathbf{c}_1}$ and $R^+_{\mathbf{c}_2}$ are the positive regions for grid squares $\mathbf{c}_1$ and $\mathbf{c}_2$, respectively, and $\mathbf{e}_1$ is the edge between $\mathbf{c}_1$ and $\mathbf{c}_2$. Similarly, $R^+_{\mathbf{c}_2} \cap \mathbf{e}_2$ equals $R^+_{\mathbf{c}_3} \cap \mathbf{e}_2$ and $R^+_{\mathbf{c}_3} \cap \mathbf{e}_3$ equals $R^+_{\mathbf{c}_4} \cap \mathbf{e}_3$.

Lemma 2.4. *Let $\mathbf{c}_1$ and $\mathbf{c}_2$ be adjacent grid squares where each vertex of $\mathbf{c}_1$ and $\mathbf{c}_2$ has a positive or a negative label. Let p be a point in $\mathbf{c}_1 \cap \mathbf{c}_2$. Point p is in $R^+_{\mathbf{c}_1}$ if and only if p is in $R^+_{\mathbf{c}_2}$.*

Proof: If p is a grid vertex, then p is in $R^+_{\mathbf{c}_1}$ and $R^+_{\mathbf{c}_2}$ if it is positive and not in $R^+_{\mathbf{c}_1}$ or $R^+_{\mathbf{c}_2}$ if it is negative. Otherwise, p must be in the interior of some grid edge $\mathbf{e}$. If edge $\mathbf{e}$ is positive, then p is in $R^+_{\mathbf{c}_1}$ and $R^+_{\mathbf{c}_2}$. If edge $\mathbf{e}$ is negative, then p is not in $R^+_{\mathbf{c}_1}$ or $R^+_{\mathbf{c}_2}$. If one endpoint, v_1, is positive and the other endpoint, v_2, is negative, then the isocontour in both grid squares intersects the grid edge in the same interpolated point q. The closed segment $[v_1, q]$ is in both $R^+_{\mathbf{c}_1}$ and $R^+_{\mathbf{c}_2}$ while the segment $(q, v_2]$ (open at q and closed at v_2) is in neither. Thus if p is in $[v_1, q]$, then p is in both $R^+_{\mathbf{c}_1}$ and $R^+_{\mathbf{c}_2}$ and if p is in $(q, v_2]$, then p is in neither. □

Using Lemma 2.4, we prove that the isocontour separates positive vertices from negative ones.

Property 5. *The isocontour separates positive grid vertices from negative grid vertices and strictly separates strictly positive grid vertices from negative grid vertices.*

Proof: For all the possible configurations, a path from a positive vertex to a negative one in a grid square must intersect the isocontour. We must show that this also holds true for paths through many grid squares.

Let R^+ be the union of the positive regions over all the grid squares. Consider a path ζ in the grid from a positive grid vertex to a negative one. The positive grid vertex lies in R^+ while the negative one does not. Thus ζ must intersect some point p on the boundary of R^+ where it crosses out of R^+. Every neighborhood of p must contain points that are not in R^+.

Since R^+ is closed, point p lies in R^+. Thus point p lies in $R^+_{\mathbf{c}'}$ for some grid square $\mathbf{c}'$. By Lemma 2.4, point p lies in $R^+_{\mathbf{c}}$ for every grid square $\mathbf{c}$ containing p. Assume p is not on the isocontour. Within each grid square containing p, some neighborhood of p is contained in the positive region for that grid square. The union of those neighborhoods is a neighborhood of p within the grid and is contained in R^+. Thus ζ does not cross out of R^+ at p. We conclude that p must lie on the isocontour and that ζ intersects the isocontour. Thus the isocontour separates positive from negative grid vertices.

If the scalar value of a grid vertex does not equal the isovalue, then the grid vertex does not lie on the isocontour. Thus the isocontour strictly separates strictly positive grid vertices from negative ones. (By definition, the scalar value of a negative vertex never equals the isovalue.) □

To prove properties 6 and 7, we prove something slightly more general.

Proposition 2.5. *Let p be any point on the* Marching Squares *isocontour that is not a grid vertex with scalar value equal to the isovalue.*

1. *If p is in the interior of the grid, then the isocontour restricted to some sufficiently small neighborhood of p is a 1-manifold.*
2. *If p is on the boundary of the grid, then the isocontour restricted to some sufficiently small neighborhood of p is a 1-manifold with boundary.*

Proof: Let v be a grid vertex with scalar value s_v. If s_v is not the isovalue, then the isocontour does not contain v, so point p is not v. If s_v equals the isovalue, then, by assumption, point p is not v. Therefore, point p is not a grid vertex.

If p lies in the interior of a grid square, then it lies in the interior of some isocontour edge. The interior of this edge is a 1-manifold containing p.

Assume p lies on the boundary of a grid square but not on the boundary of the grid. Since p is not a grid vertex, point p must lie in the interior of some grid edge $\mathbf{e}$ with one positive and one negative vertex. The two grid squares containing $\mathbf{e}$ each contain a single contour edge with endpoint at p. The interior of these two contour edges and the point p form a 1-manifold containing p.

Finally, assume p lies on the boundary of the grid. Since p is not a grid vertex, point p is contained in a single grid square. This grid square contains a single contour edge with endpoint at p. This contour edge is a manifold with boundary containing p. □

Properties 6 and 7 apply to MARCHING SQUARES isocontours whose isovalues do not equal the scalar value of any grid vertex.

Property 6. *The isocontour is a piecewise linear 1-manifold with boundary.*

Property 7. *The boundary of the isocontour lies on the boundary of the grid.*

Proof of Properties 6 & 7: Consider a point p on the isocontour. Since the isovalue does not equal the scalar value of any grid vertex, point p is not a grid vertex. By Proposition 2.5, the isocontour restricted to some suitably small neighborhood of point p is either a 1-manifold or a 1-manifold with boundary. Thus the isocontour is a 1-manifold with boundary. Since the restricted isocontour is a 1-manifold whenever p is in the interior of the grid, the boundary of the isocontour must lie on the grid boundary. □

The last property is that Υ does not contain any zero-length or duplicate edges and forms a triangulation of the isocontour.

Property 8. *Set Υ does not contain any zero-length line segments or duplicate line segments, and the line segments in Υ form a "triangulation" of the isocontour.*

Proof: Since no grid vertex has scalar value equal to the isovalue, no isocontour vertex lies on a grid vertex. By Property 4, each bipolar grid edge contains only one isocontour vertex. Thus, the linear interpolation on isocontour vertices does not create any zero-length or duplicate isocontour edges. Since isocontour edges intersect only at their endpoints, Υ forms a triangulation of the isocontour. □

2.2.5 2D Ambiguity

Set $E_\kappa^{+/-}$ is the set of bipolar square edges for configuration κ. The combinatorial structure of the isocontour depends upon the matching of the elements of $E_\kappa^{+/-}$. If $E_\kappa^{+/-}$ has two elements, then there is no choice. However, if $E_\kappa^{+/-}$ has four bipolar edges, then there are two possible pairings and two possible isocontours that could be constructed for configuration κ. Configurations 8 and 16 from Figure 2.2 have four bipolar edges. They are called ambiguous configurations. These two ambiguous configurations are shown in Figure 2.12 along with the two combinatorially distinct isocontours for each ambiguous configuration.

Choosing different isocontours for the ambiguous configurations will change the topology of the overall isocontour. For instance, Figure 2.13 shows the same

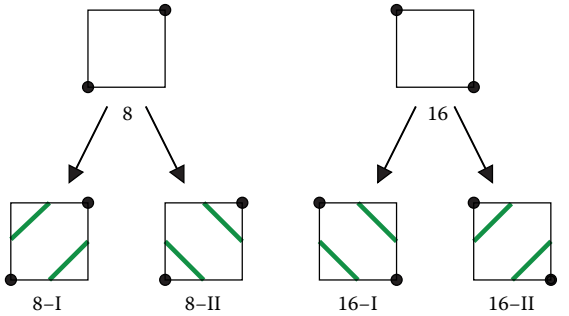


Figure 2.12. Ambiguous square configurations.

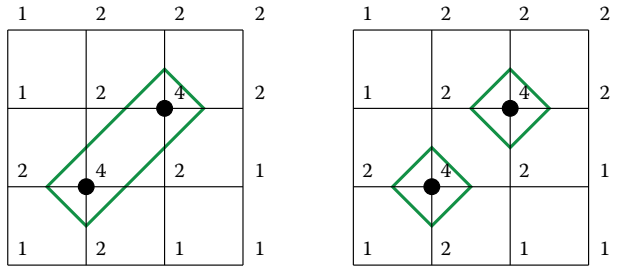


Figure 2.13. Topologically distinct isocontours created by using different isocontours for the ambiguous configuration in the central grid square.

scalar grid with two topologically distinct isocontours created by different resolutions of the ambiguous configurations. The first isocontour has two components while the second has one.

While the choice of isocontours for the ambiguous configurations changes the isocontour topology, any of the choices will produce isocontours that are 1-manifolds and strictly separate strictly positive vertices from negative vertices. As we shall see, this is not true in three dimensions.

2.3 Marching Cubes

2.3.1 Algorithm

The three-dimensional Marching Cubes algorithm follows precisely the steps in the two-dimensional Marching Squares algorithm. Input to the March-

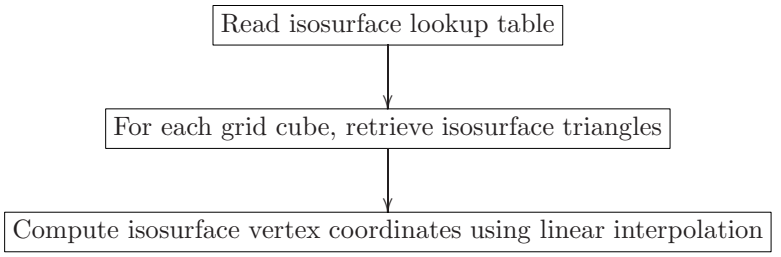


Figure 2.14. Marching Cubes.

ing Cubes algorithm is an isovalue and a set of scalar values at the vertices of a three-dimensional regular grid. The algorithm has three steps. (See Figure 2.14.) Read the isosurface lookup table from a preconstructed data file. For each cube, retrieve from the lookup table a set of isosurface triangles representing the combinatorial structure of the isosurface. The vertices of these triangles form the isosurface vertices. Assign geometric locations to the isosurface vertices based on the scalar values at the cube edge endpoints. We explain the last two steps below.

Grid vertices are labeled positive or negative as described in Section 2.1. Grid edges are labeled positive, negative, or bipolar.

The combinatorial structure of the isosurface within each cube is determined from the configuration of the cube's vertex labels. In order to separate the positive vertices from the negative ones, the isosurface must intersect any cube edge that has one positive and one negative endpoint. An isosurface that intersects a minimal number of grid edges will not intersect any edge whose endpoints are both strictly positive or whose endpoints are both negative.

Since each vertex is either positive or negative and a cube has eight vertices, there are $2^8 = 256$ different configurations of cube vertex labels. Many of these configurations are rotations or reflections of one another. By exploiting this symmetry, the number of distinct configurations can be reduced to twenty-two.[1] These distinct configurations are listed in Figure 2.15. All other configurations are rotations or reflections of these twenty-two.

For each cube configuration κ, let $E_\kappa^{+/-}$ be the set of edges with one positive and one negative endpoint. The isosurface lookup table contains 256 entries, one for each configuration κ. Each entry is a list of triples of edges of $E_\kappa^{+/-}$. Each triple $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$ represents a triangle whose vertices lie on $\mathbf{e}_1$, $\mathbf{e}_2$, and $\mathbf{e}_3$. The list of triples define the combinatorial structure of the isosurface patch for

[1] Lorensen and Cline's original paper on Marching Cubes [Lorensen and Cline, 1987a] listed only fifteen configurations. For reasons discussed in Section 2.3.5, twenty-two configurations are preferable.

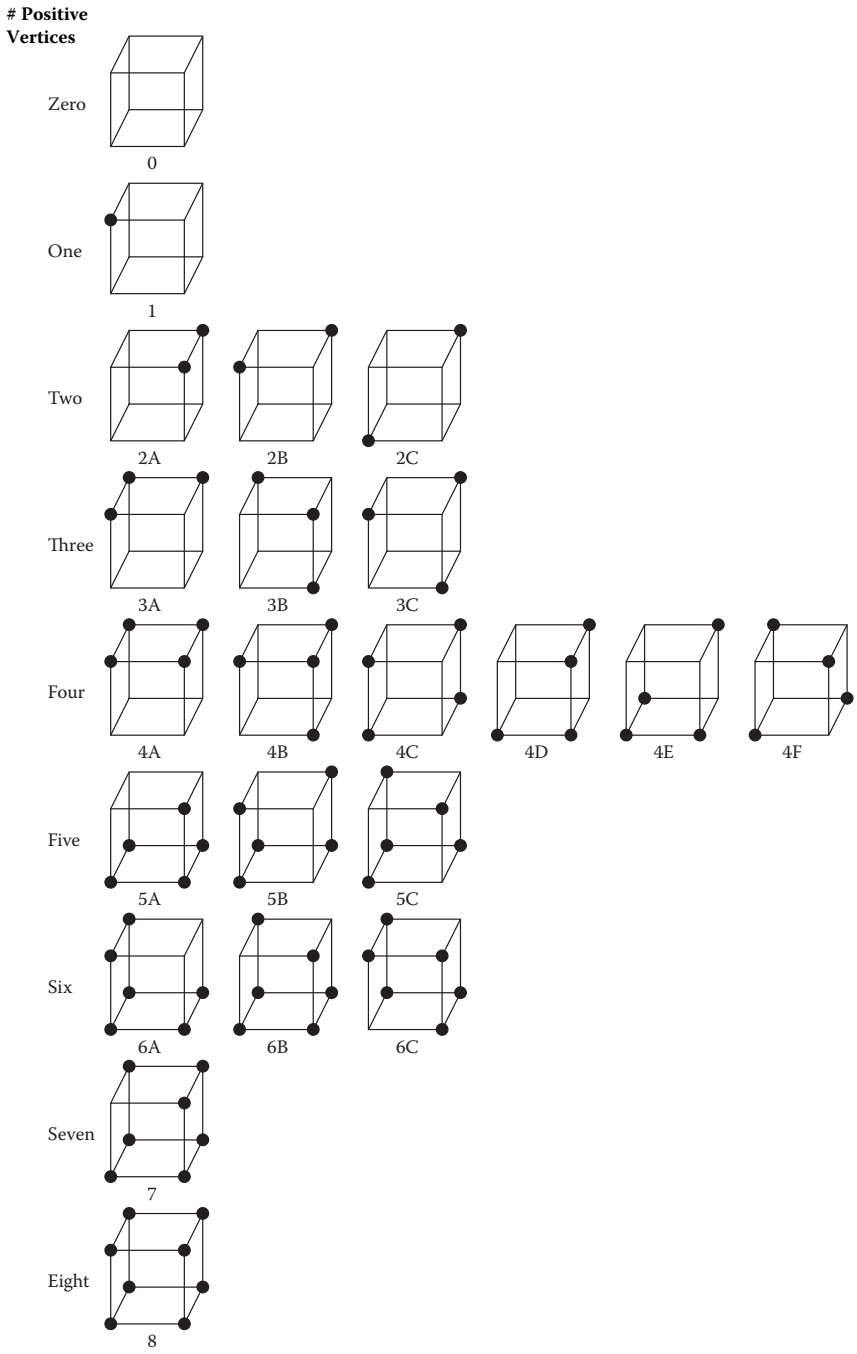


Figure 2.15. Twenty-two distinct cube configurations. Black vertices are positive.

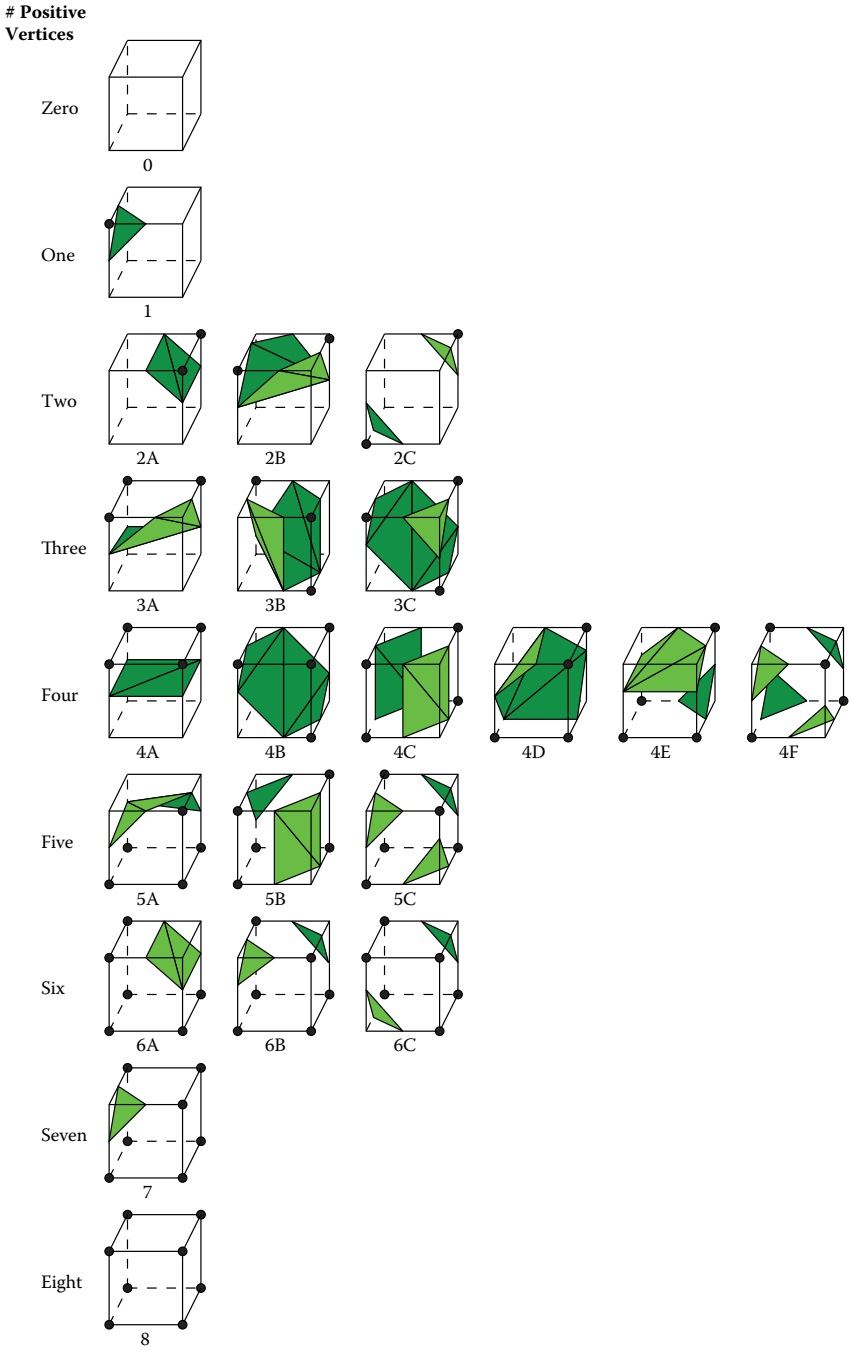


Figure 2.16. Isosurfaces for twenty-two distinct cube configurations.

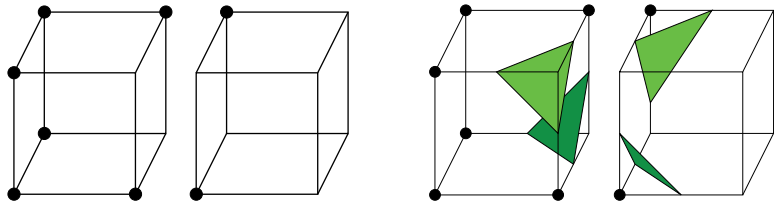


Figure 2.17. (a) Adjacent configurations sharing a common face. (b) Incompatible isosurface patches for the adjacent configurations.

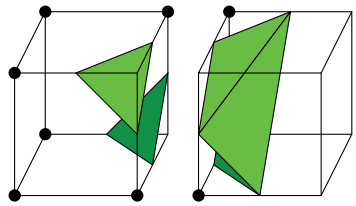


Figure 2.18. Compatible isosurface patches for adjacent configurations in Figure 2.17(a).

configuration κ. The isosurface patch intersects every edge of $E_\kappa^{+/-}$ exactly once and does not intersect any other grid cube edges.

To define the 256 entries in the table, it is only necessary to determine the table entries for the twenty-two distinct configurations. The table entries for the other configurations can be derived using rotation and reflection symmetry. Figure 2.16 contains the twenty-two distinct cube configurations and their isosurfaces.

The isosurface lookup table is constructed on the unit cube with vertices $(0,0,0),(1,0,0),(0,1,0),\ldots,(0,1,1),(1,1,1)$. To construct the isosurface in grid cube (i,j,k), we have to map unit cube edges to edges of cube (i,j,k). Each vertex $v=(v_x,v_y,v_z)$ of the unit cube maps to $v+(i,j,k)=(v_x,v_y,v_z)+(i,j,k)=(v_x+i,v_y+j,v_z+k)$. Each edge $\mathbf{e}$ of the unit square with endpoints (v,v') maps to edge $\mathbf{e}+(i,j,k)=(v+(i,j,k),v'+(i,j,k))$. Finally, each edge triple $(\mathbf{e}_1,\mathbf{e}_2,\mathbf{e}_3)$ maps to $(\mathbf{e}_1+(i,j,k),\mathbf{e}_2+(i,j,k),\mathbf{e}_3+(i,j,k))$.

In Figure 2.16, the isosurface vertices lie on the midpoints of the grid edges. This is for illustration purposes only. The geometric locations of the isosurface vertices are not defined by the lookup table.

The vertices of the isosurface triangles are the isosurface vertices. To map each isosurface triangle to a geometric triangle, we use linear interpolation to position the isosurface vertices as described in Section 1.7.2. Each isosurface vertex v lies on a grid edge $[p,q]$. If s_p and s_q are the scalar values at p and q and σ is the isovalue, then map v to $(1-\alpha)p+q$ where $\alpha=(\sigma-s_p)/(s_q-s_p)$.

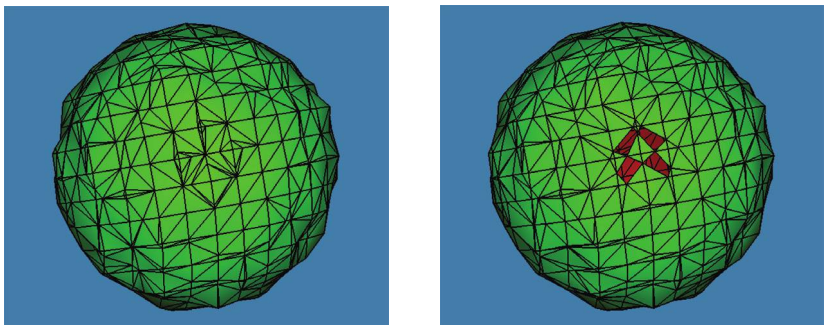
```
Input  : F is a 3D array of scalar values.
         Coord is a 3D array of (x, y, z) coordinates.
         σ is an isovalue.
Result : A set ϒ of isosurface triangles.

MarchingCubes(F, Coord, σ, ϒ)
Read Marching Cubes lookup table into Table;
/* Assign "+" or "−" signs to each vertex                          */
foreach grid vertex (i, j, k) do
    if F[i, j, k] < σ then Sign[i, j, k] ← "−";
    else Sign[i, j, k] ← "+";                          /* F[i, j, k] ≥ σ */
end
T ← ∅;
/* For each grid cube, retrieve isosurface triangles               */
foreach grid cube (i, j, k) do
    /* Cube vertices are (i, j, k), (i+1, j, k), ..., (i+1, j+1, k+1) */
    κ ← (Sign[i, j, k], Sign[i+1, j, k], Sign[i, j+1, k], ..., Sign[i+1, j+1, k+1]);
    foreach edge triple (e1, e2, e3) ∈ Table[κ] do
        Insert edge triple (e1 + (i, j, k), e2 + (i, j, k), e3 + (i, j, k)) into T;
    end
end
/* Compute isosurface vertex coordinates using linear interpolation */
foreach bipolar grid edge e with endpoints (i1, j1, k1) and (i2, j2, k2) do
    /* Compute the isosurface vertex we on edge e                  */
    we ← LinearInterpolation
            (Coord[i1, j1, k1], F[i1, j1, k1], Coord[i2, j2, k2], F[i2, j2, k2], σ);
end
/* Convert T to set of triangles                                   */
ϒ ← ∅;
foreach triple of edges (e1, e2, e3) ∈ T do
    ϒ ← ϒ ∪ {(we1, we2, we3)};
end
```

Algorithm 2.2. Marching Cubes.

Note that since p and q have different sign, scalar s_p does not equal s_q and the denominator $(s_q - s_p)$ is never zero.

The Marching Cubes algorithm is presented in Algorithm 2.2. Function `LinearInterpolation`, called by this algorithm, is defined in Algorithm 1.1 in Section 1.7.2.

Two configurations can be adjacent to one another if the face they share in common has the same set of positive and negative vertices. Figure 2.17 contains



(a) (b)

Figure 2.19. (a) Spherical isosurface of noisy point cloud. (b) Isosurface of noisy point cloud constructed using incorrect isosurface lookup table. The sphere interior is colored red and is visible through holes in the sphere. The holes are caused by incompatible isosurface patches in the isosurface lookup table.

an example of two such configurations. The isosurface patches for each configuration should be constructed so that their boundaries align on the common face. As shown in Figure 2.17, two reasonable isosurface patches for adjacent configurations can have boundaries that do not align on the common face and so are incompatible. Isosurfaces constructed using such incompatible isosurface patches for adjacent configurations may have "holes" and may not be a 2-manifold. (See Figure 2.19.) Compatible isosurface patches for the configuration in Figure 2.17(a) are given in Figure 2.18.

The isosurface patches for the twenty-two distinct configurations in Figure 2.16 were constructed so that they and all the 256 derived configurations are compatible. The isosurface patch boundaries align on the common face of any two adjacent configurations. As will be discussed in Section 2.3.5, Lorensen and Cline's original MARCHING CUBES algorithm [Lorensen and Cline, 1987a] lacked this property.

If the constructed isosurface is a manifold, then the manifold is orientable. After the construction of the isosurface, the isosurface can be assigned an orientation by assigning consistent orientations to all its triangles. However, a better approach is to properly orient the triangles in the lookup table. Each isosurface patch in a lookup table entry separates a positive region from a negative one. As noted in Section 1.7.1, a triangle orientation determines a vector orthogonal to the triangle. Orient each triangle in the isosurface patch so that the induced orthogonal vector points toward the positive region. Retrieve the oriented triangles from the lookup table to determine the orientation of the isosurface triangles. The orientations of the triangles are consistent and form an orientation of the isosurface.

2.3.2 Running Time

The Marching Cubes algorithm runs in linear time.

Proposition 2.6. *Let N be the total number of vertices of a 3D scalar grid. The running time of the* Marching Cubes *algorithm on the scalar grid is $\Theta(N)$.*

The proof is similar to the proof for the 2D Marching Squares algorithm in Section 2.2.2 and is omitted.

2.3.3 Isosurface Properties

The isosurface produced by the Marching Cubes algorithm has the same properties as the isocontour produced by the Marching Squares algorithm. As in the 2D version, we differentiate between the case where the isovalue equals the scalar value of one or more grid vertices and the case where it does not. If the isovalue does not equal the scalar value of any grid vertices, then the isosurface is a piecewise linear, orientable 2-manifold with boundary. If the isovalue equals the scalar value of some grid vertex, then the isosurface may not be a 2-manifold and the isosurface may have zero-length edges and zero-area triangles.

Marching Cubes returns a finite set, Υ, of oriented triangles. The isosurface is the union of these triangles. The vertices of the isosurface are the triangle vertices.

The following properties apply to all isosurfaces produced by the Marching Cubes algorithm.

Property 1. *The isosurface is piecewise linear.*

Property 2. *The vertices of the isosurface lie on grid edges.*

Property 3. *The isosurface intersects every bipolar grid edge at exactly one point.*

Property 4. *The isosurface does not intersect any negative or strictly positive grid edges.*

Property 5. *The isosurface separates positive grid vertices from negative ones and strictly separates strictly positive grid vertices from negative grid vertices.*

Properties 3 and 4 imply that the isosurface intersects a minimum number of grid edges. As in two dimensions, if both endpoints of a grid edge have scalar value equal to the isovalue, then the isosurface may intersect the grid edge zero, one, or two times or may contain the grid edge.

By Property 3, the isosurface intersects every bipolar grid edge. However, the bipolar grid edge may be intersected by zero-area isosurface triangles.

The following properties apply to the MARCHING CUBES isosurfaces whose isovalues do not equal the scalar value of any grid vertex.

Property 6. *The isosurface is a piecewise linear, orientable 2-manifold with boundary.*

Property 7. *The boundary of the isosurface lies on the boundary of the grid.*

Property 8. *Set Υ does not contain any zero-area triangles or duplicate triangles and the triangles in Υ form a triangulation of the isosurface.*

Scanning devices usually produce data sets whose scalar values are 8-bit, 12-bit, or 16-bit integers. For such data sets, a MARCHING CUBES isosurface with noninteger isovalues has no degenerate triangles and is always a manifold. Typically, the isovalue $(x + 0.5)$ is used, where x is some integer.

2.3.4 Proof of Isosurface Properties

We give a proof of each of the properties listed in Section 2.3.3. The proofs are the same as the proofs for MARCHING SQUARES isocontours.

Property 1. *The isosurface is piecewise linear.*

Property 2. *The vertices of the isosurface lie on grid edges.*

Proof of Properties 1 & 2: The MARCHING CUBES isosurface consists of a finite set of triangles, so it is piecewise linear. By construction, the vertices of these triangles lie on the grid edges. □

Property 3. *The isosurface intersects every bipolar grid edge at exactly one point.*

Property 4. *The isosurface does not intersect any negative or strictly positive grid edges.*

Proof of Properties 3 & 4: Each isosurface triangle is contained in a grid cube. Since the grid cubes are convex, only isosurface triangles with vertices on the grid edge intersect the grid edge. If the grid edge has one positive and one negative endpoint, the unique location of the isosurface vertex on the grid edge is determined by linear interpolation. Thus the isosurface intersects a bipolar grid edge at exactly one point.

If the grid edge is negative or strictly positive, then no isosurface vertex lies on the grid edge. Thus the isosurface does not intersect negative or strictly positive grid edges. □

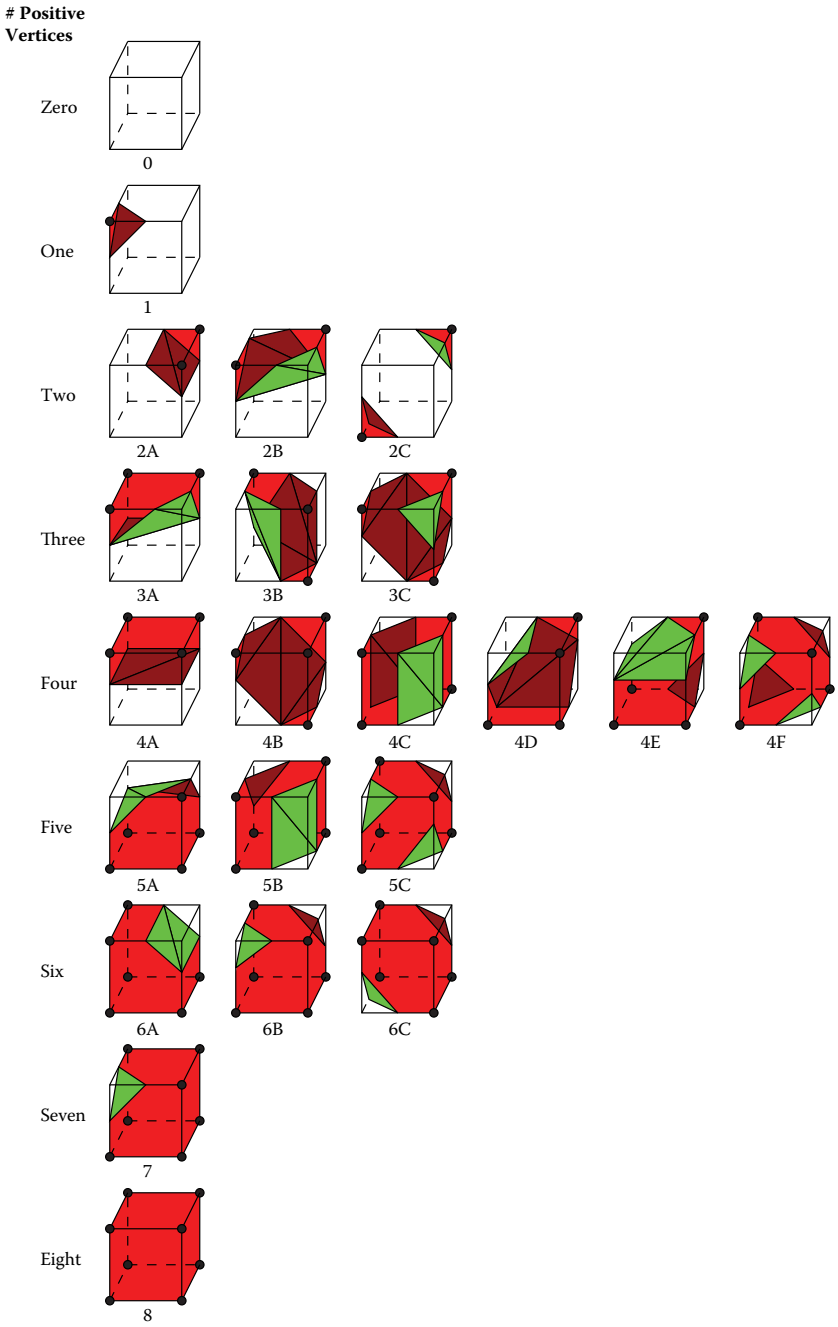


Figure 2.20. Positive regions (red) for the twenty-two distinct cube configurations. Visible portion of each isosurface is green. Portion of the isosurface behind each positive region is colored dark red.

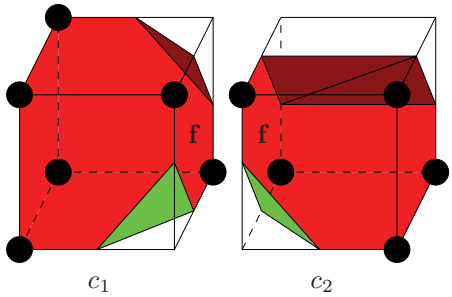


Figure 2.21. Adjacent cubes $\mathbf{c}_1$ and $\mathbf{c}_2$ and (red) positive regions $R^+_{\mathbf{c}_1}$ and $R^+_{\mathbf{c}_2}$. Black cube vertices are positive. Shared facet $\mathbf{f}$ lies between the two cubes. Note that $R^+_{\mathbf{c}_1} \cap \mathbf{f}$ equals $R^+_{\mathbf{c}_2} \cap \mathbf{f}$.

The proof of Property 5, the separation property, is similar to the proof of Property 5 for the MARCHING SQUARES isosurface (Section 2.2.4). We first extend the definition of positive and negative regions from grid squares to grid cubes.

Within each grid cube the isosurface partitions the grid cube into two regions. Define the positive region for a grid cube $\mathbf{c}$ to be points $p \in \mathbf{c}$ where some path $\zeta \subset \mathbf{c}$ connects p to a positive vertex of $\mathbf{c}$ and the interior of ζ does not intersect the isosurface (Figure 2.20). Define the negative region for a grid cube $\mathbf{c}$ to be points $p \in \mathbf{c}$ where some path $\zeta \subset \mathbf{c}$ connects p to a negative vertex of $\mathbf{c}$ and ζ does not intersect the isosurface. Since any path $\zeta \subset \mathbf{c}$ from a positive to a negative vertex must intersect the isosurface, the positive and negative regions form a partition of the cube $\mathbf{c}$.

The positive region is closed and contains the isosurface. Any point within the positive region that does not lie on the isosurface has a neighborhood contained in the positive region.

Every negative vertex is contained in the negative region since the zero-length path connects the vertex to itself. Every positive vertex is contained in the positive region since a path in a cube from any negative cube vertex to a positive one must intersect the isosurface.

We claim that positive and negative regions agree on the grid cube boundaries (Figure 2.21). Let $R^+_{\mathbf{c}}$ be the positive region for a grid cube $\mathbf{c}$.

Lemma 2.7. *Let $\mathbf{c}_1$ and $\mathbf{c}_2$ be adjacent grid cubes where each vertex of $\mathbf{c}_1$ and $\mathbf{c}_2$ has a positive or a negative label. Let p be a point in $\mathbf{c}_1 \cap \mathbf{c}_2$. Point p is in $R^+_{\mathbf{c}_1}$ if and only if p is in $R^+_{\mathbf{c}_2}$.*

The proof of Lemma 2.7 is based on an exhaustive examination of all possible adjacent configurations and is omitted. A more detailed and satisfying analysis is given in Chapter 5, which contains an algorithm for generating the lookup table for the MARCHING CUBES and similar algorithms.

Using Lemma 2.7, we prove that the isosurface separates positive vertices from negative ones.

Property 5. *The isosurface separates positive grid vertices from negative ones and strictly separates strictly positive grid vertices from negative grid vertices.*

Proof: For all the possible configurations, a path from a positive vertex to a negative one in a grid cube must intersect the isosurface. We must show that this also holds true for paths through many grid cubes.

Consider a path ζ in the grid from a positive grid vertex to a negative one. The positive grid vertex lies in R^+ while the negative one does not. Thus ζ must intersect some point p on the boundary of R^+ where it crosses out of R^+. Every neighborhood of p must contain points that are not in R^+.

Since R^+ is closed, point p lies in R^+. Thus point p lies in $R^+_{\mathbf{c}'}$ for some grid cube $\mathbf{c}'$. By Lemma 2.7, point p lies in $R^+_{\mathbf{c}}$ for every grid cube $\mathbf{c}$ containing p. If p does not lie on the isosurface, then some neighborhood of p is contained in the positive region $R^+_{\mathbf{c}}$ of each grid cube $\mathbf{c}$ containing p. The union of those neighborhoods is a neighborhood of p within the grid and is contained in R^+. Thus ζ does not cross out of R^+ at p. We conclude that p must lie on the isosurface and that ζ intersects the isosurface. Thus the isosurface separates positive from negative grid vertices.

If the scalar value of a grid vertex does not equal the isovalue, then the grid vertex does not lie on the isosurface. Thus the isosurface strictly separates strictly positive grid vertices from negative ones. (By definition, the scalar value of a negative vertex never equals the isovalue.) □

As in two dimensions, to prove Properties 6 and 7, we first prove something slightly more general.

Proposition 2.8. *Let p be any point on the* Marching Cubes *isosurface that is not a grid vertex with scalar value equal to the isovalue.*

1. *If p is in the interior of the grid, then the isosurface restricted to some sufficiently small neighborhood of p is a 2-manifold.*

2. *If p is on the boundary of the grid, then the isosurface restricted to some sufficiently small neighborhood of p is a 2-manifold with boundary.*

Proof: Let v be a grid vertex with scalar value s_v. If s_v is not the isovalue, then the isosurface does not contain v, so point p is not v. If s_v equals the isovalue, then, by assumption, point p is not v. Therefore, point p is not a grid vertex.

If p lies in the interior of a grid cube, then it lies in the interior of some isosurface patch. The interior of this patch is a 2-manifold containing p.

Assume p lies on the boundary of a grid cube but not on the boundary of the grid. Since p is not a grid vertex, point p either lies in the interior of a grid edge

or the interior of a square grid facet. If point p lies in the interior of a square grid facet, then it is contained in two isosurface triangles lying in two adjacent grid cubes. The interior of the union of these two isosurface triangles forms a 2-manifold containing p. If point p lies in the interior of a grid edge $\mathbf{e}$, then p is contained in four isosurface patches lying in the four grid cubes containing $\mathbf{e}$. The interior of the union of these four patches forms a 2-manifold containing p.

Finally, assume p lies on the boundary of the grid. Since p is not a grid vertex, point p either lies in one or two grid cubes. If p lies in a single grid cube, then a single isosurface triangle in this grid cube contains p. This triangle is a 2-manifold with boundary containing p. If p lies in two grid cubes, then the union of the isosurface patches in these two grid cubes form a 2-manifold with boundary containing p. □

Properties 6 and 7 apply to MARCHING CUBES isosurfaces whose isovalues do not equal the scalar value of any grid vertex.

Property 6. *The isosurface is a piecewise linear, orientable 2-manifold with boundary.*

Property 7. *The boundary of the isosurface lies on the boundary of the grid.*

Proof of Properties 6 & 7: Consider a point p on the isosurface. Since the isovalue does not equal the scalar value of any grid vertex, point p is not a grid vertex. By Proposition 2.8, the isosurface restricted to some suitably small neighborhood of point p is either a 2-manifold or a 2-manifold with boundary. Thus the isosurface is a 2-manifold with boundary. All of the triangles in the isosurface are oriented so that the induced orthogonal vector points toward the positive vertices; thus, all the triangle orientations are consistent and the manifold is orientable.

Since the restricted isosurface is a 2-manifold whenever p is in the interior of the grid, the boundary of the isosurface must lie on the grid boundary. □

The last property is that Υ does not contain any zero-area or duplicate triangles and forms a triangulation of the isosurface.

Property 8. *Set Υ does not contain any zero-area triangles or duplicate triangles and the triangles in Υ form a triangulation of the isosurface.*

Proof: Since no grid vertex has scalar value equal to the isovalue, no isosurface vertex lies on a grid vertex. By Property 3, each bipolar grid edge contains only one isosurface vertex. Thus, the linear interpolation on isosurface vertices does not create any zero-area or duplicate isosurface triangles. The isosurface triangles within a grid cube are a subset of the triangulation of an isosurface patch and so the nonempty intersection of any two such triangles is either a vertex or an edge of both. The isosurface patches in Figure 2.16 were constructed so that the intersection of any two isosurface triangles in adjacent grid cubes is either a vertex or an edge of both. Since the nonempty intersection of any two triangles in Υ is a face of both triangles, Υ forms a triangulation of the isosurface. □

2.3.5 3D Ambiguity

The 256 configurations of positive and negative vertices are generated from the twenty-two configurations in Figure 2.15 using rotational and reflection symmetry. There is another kind of symmetry that is not being used. Configuration κ_1 is the complement of configuration κ_2 if κ_1 can be derived from κ_2 by switching κ_2's positive and negative vertex labels. The configurations with five, six, seven, or eight positive vertices are all complements of configurations with three, two, one, and zero positive vertices. Complementary symmetry reduces the number of unique configurations from twenty-two to fourteen.

Unfortunately, complementary symmetry creates incompatible isosurface patches in the isosurface table. For instance, the configuration on the left in Figure 2.17 is a rotation of configuration 6B from Figure 2.15 while the configuration on the right is a rotation of configuration 2B. Configuration 2B is the complement of configuration 6B. Both isosurface patches on the left and on the right come from a rotation of the isosurface patch for 6B in Figure 2.16. The boundaries of the isosurface patches clearly do not align on the square between the two cubes.

Lorensen and Cline's paper [Lorensen and Cline, 1987a] used complementary symmetry and so created incompatible isosurface patches in the isosurface table [Dürst, 1988].[2] The problem configurations were the ones that had ambiguous configurations on their square facets. These are configurations 2B, 3B, 3C, 4C, 4E, 4F, 5B, 5C and 6B. Numerous papers were written on how to handle these problematic configurations. The most widely adopted approach, proposed in [Montani et al., 1994] and in [Zhou et al., 1994], is to simply drop complementary symmetry. This is the approach adopted in the previous section.

As discussed in Section 2.2.5, the two-dimensional ambiguous configurations have two isocontour edges and two combinatorially different ways to position those edges. One of these edge positions separates the two negative vertices by isocontour edges (8-I and 16-I in Figure 2.12) while the other separates the positive vertices (8-II and 16-II in Figure 2.12).

The border of a cube's three-dimensional isosurface patch defines an isocontour on each of the cube's square facets. If some configuration's isosurface patch separates the negative vertices on the facet while an adjacent configuration's isosurface patch separates the positive ones, then the isosurface edges on the common facet will not align. The isosurface patches in Figure 2.16 do not separate the positive vertices on any facet.[3] Moreover, the derived isosurface surface patches in any rotation or reflection of the configurations also do not separate positive vertices on any facet. Thus the isosurface patches in any two adjacent

[2] Lorensen and Cline's paper did not use reflection symmetry and so they had fifteen, not fourteen, distinct configurations.

[3] Note that in configuration 2C, the two positive vertices do not share a facet. The isosurface patch separates the two positive vertices, but not on any facet. Since these vertices do not share a facet, the isosurface patch aligns with all possible adjacent configurations.

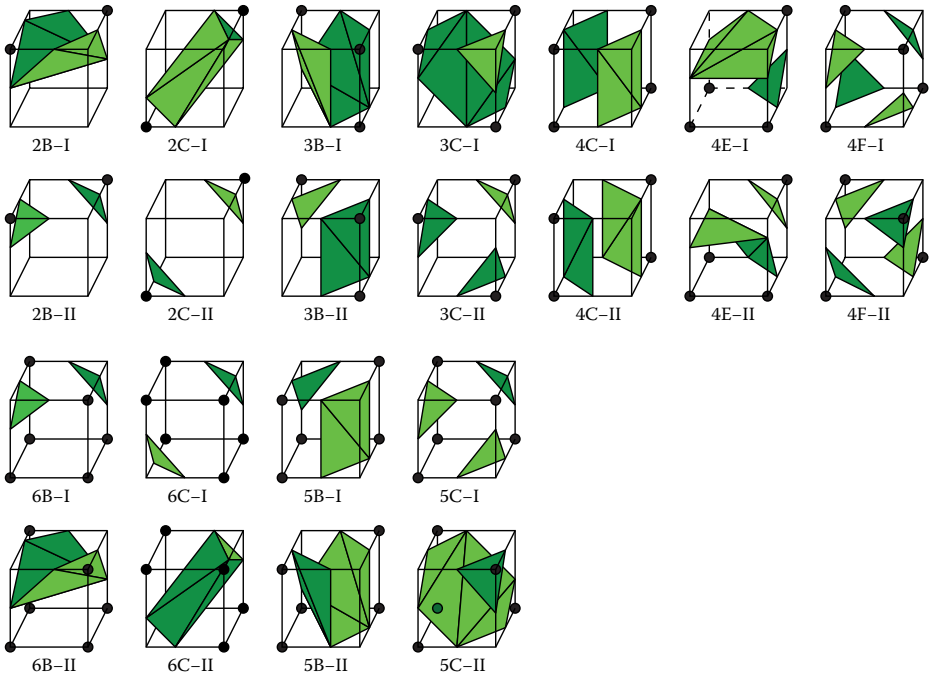


Figure 2.22. Two "natural" isosurface patches for the ambiguous 3D configurations. Isosurface patches in the first and third rows separate negative vertices. Isosurface patches in the second and fourth rows separate positive vertices.

cubes are properly aligned on their boundaries. An equally valid, but combinatorially distinct, isosurface table could be generated by using isosurface patches that do not separate the negative vertices on any square facet.

The ambiguous configurations in $\mathbb{R}^3$ are 2B, 2C, 3B, 3C, 4C, 4E, 4F, 5B, 5C, 6B and 6C. These configurations have topologically distinct piecewise linear isosurfaces with all the properties listed in the previous section. Ambiguous configurations admit two "natural" isosurface patches, one separating positive vertices and one separating negative vertices. (See Figure 2.22.)

As we have already discussed, configurations 2B, 3B, 3C, 4C, 4E, 4F, 5B, 5C and 6B have square facets with ambiguous 2D configurations. Configurations 2C and 6C are the exceptions.[4] None of their square facets have an ambiguous configuration, yet they admit two topologically distinct piecewise linear isosurfaces. (See Figure 2.22.) Usually, isosurface patches 2C-II and 6C-I are used since they each require only two isosurface triangles and do not create the "tunnel" in 2C-I and 6C-II.

[4]The numerous papers on resolving the ambiguous 3D configurations do not discuss these two, but they do admit topologically distinct isosurfaces and they are ambiguous.

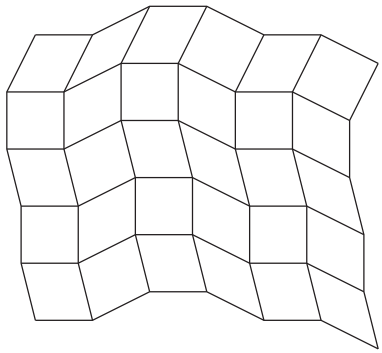


Figure 2.23. Curvilinear grid. Grid cells are quadrilaterals.

2.3.6 Curvilinear Grids

Sections 2.2 and 2.3 describe the Marching Squares and Marching Cubes algorithms for regular grids where each grid element is a geometric square or cube. These algorithms work equally well for curvilinear grids that have the same combinatorial structure as regular grids but whose elements are convex quadrilaterals or hexahedra. (See Figure 2.23.) The grid vertices are assigned positive and negative labels just as in the case of the regular grid. The same lookup table is used to determine the combinatorial structure of each isocontour or isosurface patch within a grid element. The isosurface geometry is determined by using linear interpolation to position the grid vertices along each grid edge.

2.4 Marching Tetrahedra

The Marching Cubes algorithm applies to regular and curvilinear grids. However, many applications decompose space into unstructured tetrahedral meshes composed of tetrahedral mesh elements. Marching Tetrahedra is a modified version of the Marching Cubes algorithm.

2.4.1 Algorithm

Input to the Marching Tetrahedra algorithm is an isovalue, a tetrahedral mesh, and a set of scalar values at the vertices of the tetrahedra. The mesh tetrahedra must form a triangulation of some three-dimensional region, i.e., the intersection of any two tetrahedra is a vertex, edge, or facet of both tetrahedra or is empty.

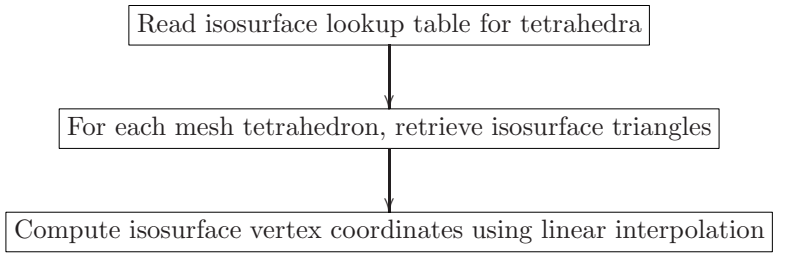


Figure 2.24. Marching Tetrahedra.

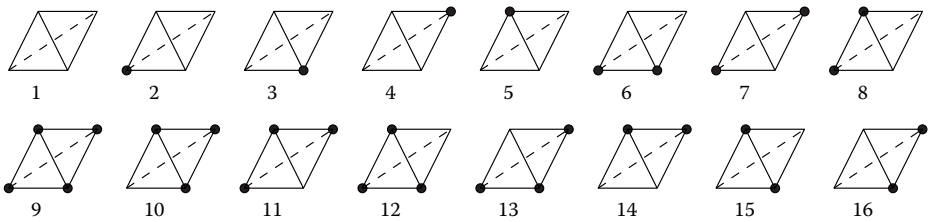


Figure 2.25. Tetrahedral configurations. Black vertices are positive.

The algorithm has three steps. (See Figure 2.24.) Read in the isosurface lookup table from a preconstructed data file. For each tetrahedron, retrieve from the lookup table a set of isosurface triangles representing the combinatorial structure of the isosurface. The endpoints of these edges form the isosurface vertices. Assign geometric locations to the isosurface vertices based on the scalar values at the triangle edge endpoints. We explain the last two steps next.

Mesh vertices are labeled positive or negative, as described in Section 2.1. Mesh edges are labeled positive, negative, or bipolar.

Since each vertex is either positive or negative and a tetrahedron has four vertices, there are $2^4 = 16$ different configurations of tetrahedron vertex labels. These configurations are listed in Figure 2.25. Applying affine transformations, only five of these configurations are distinct (Figure 2.26).

The combinatorial structure of the isosurface within each tetrahedron is determined from the configuration of the tetrahedron's vertex labels. In order to separate the positive vertices from the negative ones, the isosurface must intersect a tetrahedron edge that has one positive and one negative endpoint. An isosurface which intersects a minimal number of mesh edges will not intersect any edge whose endpoints are both strictly positive or both negative.

Each isosurface patch separates the tetrahedron into two regions, a positive one containing the positive vertices and a negative one containing the negative

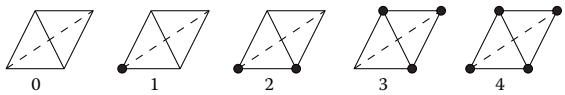


Figure 2.26. Distinct tetrahedral configurations. Configuration k has exactly k positive (black) vertices.

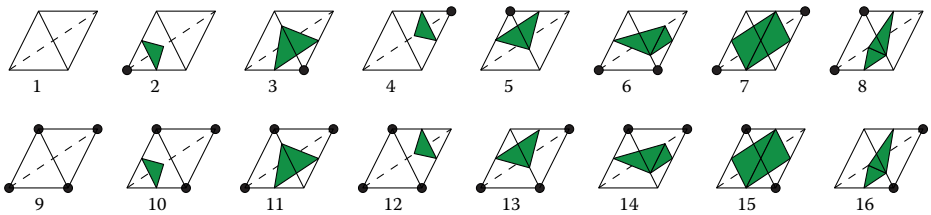


Figure 2.27. Tetrahedral isosurface patches.

vertices. Each tetrahedron on the isosurface patch is oriented so that the induced normal points to the positive region.

In Figure 2.27, the isosurface vertices lie on the midpoints of the grid edges. This is for illustration purposes only. The geometric locations of the isosurface vertices are not defined by the lookup table.

For each tetrahedral configuration κ, let $E_\kappa^{+/-}$ be the set of edges with one positive and one negative endpoint. The isosurface lookup table contains 16 entries, one for each configuration κ. Each entry is a list of triples of edges of $E_\kappa^{+/-}$. Each triple $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$ represents an oriented triangle whose vertices lie on $\mathbf{e}_1$ and $\mathbf{e}_2$ and $\mathbf{e}_3$. The orientation of the triangle is given by the order, $\mathbf{e}_1$, $\mathbf{e}_2$, $\mathbf{e}_3$. The list of triples defines the combinatorial structure of the isosurface patch for configuration κ. The isosurface patch intersects every edge of $E_\kappa^{+/-}$ exactly once and does not intersect any other grid tetrahedron edges.

To determine the isosurface patch for a specific mesh tetrahedron, match the vertices of the mesh tetrahedron to the vertices of the tetrahedron in the lookup table by numbering the vertices of the mesh tetrahedron from 1 to 4. The vertex numbering can be arbitrary and will not affect the isosurface topology, although it may alter slightly the isosurface triangulation and geometry. Determine the lookup table configuration that matches the configuration of positive and negative vertices in the mesh tetrahedron and retrieve the isosurface patch from the lookup table.

To map each isosurface triangle to a geometric triangle, we use linear interpolation to position the isosurface vertices as described in Section 1.7.2. Each isosurface vertex v lies on a grid edge $[p, q]$. If s_p and s_q are the scalar values at p and q and σ is the isovalue, then map v to $(1-\alpha)p+\alpha q$ where $\alpha = (\sigma - s_p)/(s_q - s_p)$.

2.4.2 Isosurface Properties

The isosurface produced by MARCHING TETRAHEDRA has the same properties as the isosurface produced by the MARCHING CUBES algorithm (Section 2.3.3). The proofs are the same as the proofs in Section 2.3.4 and are omitted.

MARCHING TETRAHEDRA returns a finite set, Υ, of oriented triangles. The isosurface is the union of these triangles. The vertices of the isosurface are the triangle vertices.

The following properties apply to all isosurfaces produced by the MARCHING TETRAHEDRA algorithm.

Property 1. *The isosurface is piecewise linear.*

Property 2. *The vertices of the isosurface lie on grid edges.*

Property 3. *The isosurface intersects every bipolar grid edge at exactly one point.*

Property 4. *The isosurface does not intersect any negative or strictly positive grid edges.*

Property 5. *The isosurface separates positive grid vertices from negative ones and strictly separates strictly positive grid vertices from negative grid vertices.*

Properties 3 and 4 imply that the isosurface intersects a minimum number of grid edges. As with MARCHING CUBES, if both endpoints of a grid edge have scalar value equal to the isovalue, then the isosurface may intersect the grid edge zero, one, or two times or may contain the grid edge.

By Property 3, the isosurface intersects every bipolar grid edge. However, the bipolar grid edge may be intersected by zero-area isosurface triangles.

Under appropriate conditions, the isosurface produced by MARCHING TETRAHEDRA is a 2-manifold with boundary. One condition is that the isovalue does not equal the scalar value of any grid vertex. MARCHING TETRAHEDRA can be applied to any tetrahedral mesh, as long as the tetrahedra form a triangulation. As shown in Figure 2.28, the isosurface produced by MARCHING TETRAHEDRA may not be a manifold, even if the isovalue does not equal the scalar value. The problem is that the underlying space is not a manifold. A similar problem could face an isosurface constructed by MARCHING CUBES from an arbitrary mesh of cubes. In Section 2.3, we applied MARCHING CUBES to a regular grid whose underlying space is a manifold.

Consider a tetrahedral mesh that has the following two conditions:

- The isovalue does not equal the scalar value of any mesh vertex.
- The tetrahedral mesh is a partition of a 3-manifold with boundary.

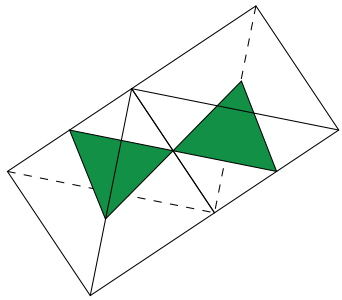


Figure 2.28. Non-manifold isosurface produced by MARCHING TETRAHEDRA. The tetrahedral mesh consists of two tetradedra sharing an edge. The region covered by the tetrahedral mesh is not a manifold with boundary.

Under these conditions, the isosurface produced by MARCHING TETRAHEDRA has the following two properties:

Property 6. *The isosurface is a piecewise linear, orientable 2-manifold with boundary.*

Property 7. *The boundary of the isosurface lies on the boundary of the grid.*

Property 8. *Set Υ does not contain any zero-area triangles or duplicate triangles and the triangles in Υ form a triangulation of the isosurface.*

One significant difference between MARCHING CUBES and MARCHING TETRAHEDRA is that MARCHING TETRAHEDRA has no ambiguous configurations. The four vertices of each tetrahedron are connected by tetrahedron edges and each such edge is intersected at most once by the isosurface, so there is no possible ambiguity in the isosurface construction.

2.4.3 Cube Tetrahedralization

MARCHING TETRAHEDRA can be used for isosurface construction in regular grids by tetrahedralizing the grid cubes. There are two distinct ways to tetrahedralize a grid cube, one that breaks the cube into five tetrahedra and one that breaks it into six. (See Figure 2.29.) The decomposition into five tetrahedra consists of a single large tetrahedra in the cube center and four smaller tetrahedra cutting off four corners of the cube. The decomposition into six tetrahedra consists of six congruent tetrahedra that all share a diagonal edge between two opposing corners of the cube.

The decomposition of a grid cube into tetrahedra creates diagonal edges on the square facets of the cube. These diagonals must match the diagonals created on adjacent cubes for the decomposition to be a triangulation. If all cubes

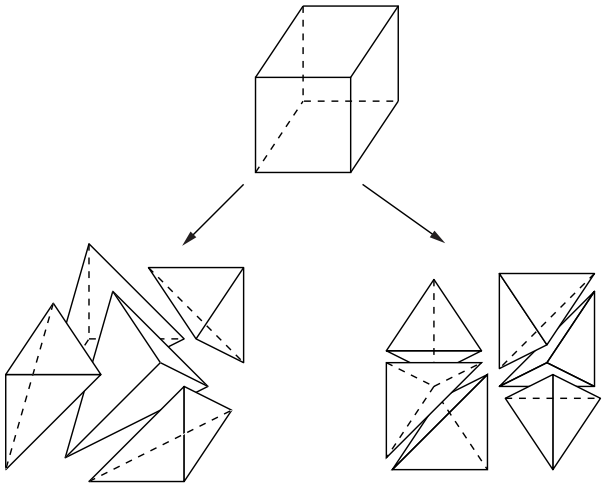


Figure 2.29. Subdivision of a cube into five or six tetrahedra.

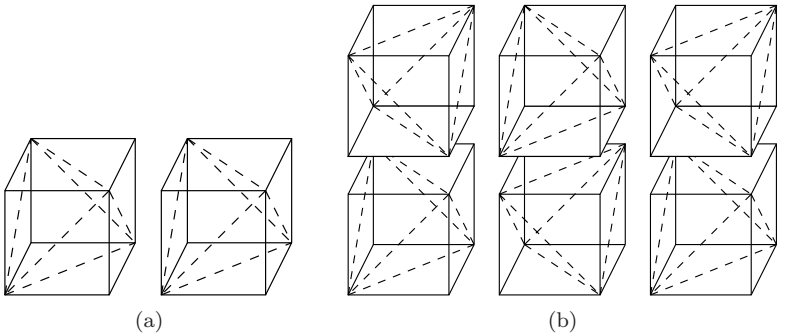


Figure 2.30. (a) Duplicate tetrahedralizations of two cubes into five tetrahedra with mismatching diagonals on the common face between the two cubes. (b) Alternating tetrahedralization of cube into five tetrahedra with matching diagonals on common faces between cubes.

are decomposed in the same manner into six tetrahedra, then these diagonals do match. However, if the cubes are decomposed in the same manner into five tetrahedra, then the diagonals of adjacent cubes will fail to match (Figure 2.30(a)). Two decompositions into five tetrahedra must be used in an alternating manner for the diagonals to match (Figure 2.30(b)).

The regular grid cubes have ambiguous configurations while the tetrahedral decomposition does not. What happened to the ambiguous configurations? These configurations are resolved by the choice of triangulation. For instance, in Figure 2.31, the first triangulation gives an isosurface patch with two compo-

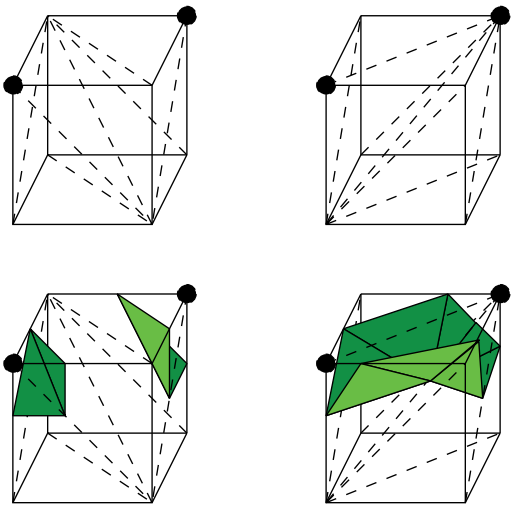


Figure 2.31. Different tetrahedralizations of a cube into six tetrahedra.

nents corresponding to 2B-II in Figure 2.22 while the second gives an isosurface patch with one component corresponding to 2B-I.

Using MARCHING TETRAHEDRA on a triangulated regular grid in place of MARCHING CUBES increases the number of triangles in resulting isosurfaces. The size of that increase depends upon the different configurations and the manner in which the tetrahedral decomposition intersects those configurations.

2.4.4 Isocontouring a Triangular Mesh

A two-dimensional simplified version of MARCHING TETRAHEDRA can be used to isocontour a triangular mesh.[5] The simplified version does not require a table lookup since each mesh triangle contains at most one isocontour edge.

Input to the isocontouring algorithm is an isovalue, a triangular mesh, and a set of scalar values at the vertices of the mesh. The mesh triangles must form a triangulation of some two-dimensional region, i.e., the intersection of any two triangles must be a vertex or edge of both triangles or must be empty.

For each bipolar mesh edge $[p, q]$, use linear interpolation to position the isosurface vertex on $[p, q]$ as described in Section 1.7.2. If s_p and s_q are the scalar values at p and q and σ is the isovalue, then map the isosurface vertex to $\alpha p + (1 - \alpha)q$ where $\alpha = (s_q - \sigma)/(s_q - s_p)$.

[5] While the name "Marching Triangles" might be appropriate for the two-dimensional version of MARCHING TETRAHEDRA, the name is already used by Hilton et al. [Hilton et al., 1996] for a completely different algorithm that constructs a triangular mesh on an implicit surface.

A triangle contains zero, one, two, or three positive vertices. If a triangle contains one or two positive vertices, then it contains two mesh edges with one positive and one negative endpoint. Connect the isocontour vertices lying on each of these mesh edges by an isocontour edge. If the isovalue equals the isovalue of exactly one of the triangle vertices v and is less than the isovalue of the other two vertices, then the isocontour vertices for each of these mesh edges will both be located at v. In that case, either connect them by a zero-length isocontour edge or merge them into a single vertex.

The resulting isocontour has all the same properties listed in Section 2.2.3 as the isocontour produced by MARCHING SQUARES. As with tetrahedral meshes, triangular meshes have no ambiguous configurations of positive and negative vertices.

2.5 Notes and Comments

Lorensen and Cline published the MARCHING CUBES algorithm in 1987 [Lorensen and Cline, 1987a]. A year earlier, Wyvill, McPheeters, and Wyvill [Wyvill et al., 1986] published a somewhat similar isosurface extraction algorithm, but without the use of isosurface patch lookup tables. A US patent [Lorensen and Cline, 1987b] for the MARCHING CUBES algorithm was granted in 1987 and the patent expired in 2005.

Dürst [Dürst, 1988] noted the problem discussed in Section 2.3.5 of incompatible surface patches in Lorensen and Cline's isosurface lookup table. Montani, Scateni, and Scopigno [Montani et al., 1994] and Zhou, Shu, and Kankanhalli [Zhou et al., 1994] proposed the corrected isosurface lookup tables described in this chapter. Wilhelms and Gelder [Gelder and Wilhelms, 1990] explored different techniques for consistently resolving ambiguous facets. Nielson and Hamann [Nielson and Hamann, 1991] used bilinear interpolation to resolve ambiguous facets. Their algorithm, the ASYMPTOTIC DECIDER, is described in Chapter 4.

The MARCHING TETRAHEDRA algorithm was proposed by a number of people [Bloomenthal, 1988, Bloomenthal, 1994, Carneiro et al., 1996, Payne and Toga, 1990, Shirley and Tuchman, 1990] although not necessarily under that name. Chan and Purisima [Chan and Purisima, 1998] apply MARCHING TETRAHEDRA to a tetrahedralization based on the body-centered cubic lattice. Theußl, Möller, and Gröller [Theußl et al., 2001] show that the body-centered cube lattice provides more accurate sampling than the regular grid. Carr, Möller, and Snoeyink [Carr et al., 2001, Carr et al., 2006b] discuss the effects of different tetrahedral subdivisions on isosurfaces generation.

The cube configurations in Figure 2.15 are distinct under the transformations of rotation and reflection. In [Banks et al., 2004], Banks, Linton, and Stockmeyer discuss counting the number of distinct configurations of cubes, hypercubes, and other convex polyhedra using computational group theory software.

The MARCHING CUBES algorithm does not produce good representations of sharp edges or corners. Kobbelt et al. [Kobbelt et al., 2001] modified the MARCHING CUBES algorithm to more accurately reconstruct sharp edges and corners when input is the distance to a given surface. Other algorithms for reproducing sharp edges and corners are presented in [Ho et al., 2005] and [Schaefer and Warren, 2004]. Dual contouring algorithms for reconstructing sharp edges and corners are referenced in the notes and comments section of Chapter 3.

The algorithms in [Ho et al., 2005, Schaefer and Warren, 2004] and the dual contouring algorithms can also create multiresolution isosurfaces (multiresolution isosurfaces are defined and discussed in Chapters 9 and 10). All the algorithms for reconstructing sharp edges and corners require accurate measurements of the gradients or surface normals at isosurface vertices.

The MARCHING CUBES and MARCHING TETRAHEDRA algorithms produce triangulated isosurface meshes with many small, thin triangles. There are numerous geometric algorithms for improving the quality and reducing the size of a triangle surface mesh. (See the surveys [Cignoni et al., 1998, Luebke, 2001, van Kaick and Pedrini, 2006, Alliez et al., 2007].) We will only mention the work specific to isosurface generation.

The MARCHING CUBES and MARCHING TETRAHEDRA algorithms produce poor quality triangles when the isosurface passes close to a grid or mesh vertex. Tzeng [Tzeng, 2004] improves triangle quality by warping the underlying regular grid so that isosurface vertices are near grid edge midpoints and far from the grid vertices. Improvements on MARCHING TETRAHEDRA triangle quality based on "snapping" tetrahedral mesh vertices to the isosurface are given in [Hall and Warren, 1990, Treece et al., 1999, Labelle and Shewchuk, 2007]. Labelle and Schewchuk [Labelle and Shewchuk, 2007] provide guaranteed bounds on isosurface triangle quality using a "snapping"-based algorithm. Raman and Wenger [Raman and Wenger, 2008] provide guaranteed bounds on MARCHING CUBES isosurface triangle quality using the "snapping" of regular grid vertices to the isosurface.

Various triangulations can be used for isosurface patches in the MARCHING CUBES lookup table. Dietrich et al. [Dietrich et al., 2008, Dietrich et al., 2009a, Dietrich et al., 2009b] improve the quality of the isosurface triangles by modifying and selecting the "best" triangulation. Nielson [Nielson, 2003a] gives triangulations for all cases in the lookup table such that the projection to some coordinate plane is one-to-one. This allows local representation of the isosurface as a function of the two axes of the coordinate plane.

Alternative approaches have been developed for generating isosurface meshes with quality elements. Schreiner et al. [Schreiner et al., 2006] use an advancing front technique to generate high-quality isosurfaces. Dey and Levine [Dey and Levine, 2007] present a quality isosurface mesh generation algorithm using the Voronoi diagram of a set of sample points on the isosurface. Both methods are adaptive, with more triangles inserted in areas of high curvature. Based on a related paper [Cheng et al., 2004], Dey and Levine's algorithm provides

guarantees of triangle quality but at the expense of potentially adding large numbers of triangles in high curvature areas. Both the advancing front technique and the Voronoi-based technique take time that is an order of magnitude longer than the running time of the MARCHING CUBES algorithm.

Isosurface patches in the isosurface lookup table can be constructed of smooth elements such as bicubic or Bezier surfaces instead of piecewise linear triangles. Smooth isosurface patches are described in [Gallagher and Nagtegaal, 1989, Hall and Warren, 1990, Hamann et al., 1997, Theisel, 2002].

Nielson et al. [Nielson et al., 2003] present an algorithm to improve smoothness of the MARCHING CUBES isosurface by moving isosurface vertices along grid edges.

The SPIDERWEB algorithm by Karron [Karron, 1992] constructs an isosurface by placing an isosurface vertex on each bipolar grid edge and connecting those vertices to another isosurface vertex within the grid cube. It produces substantially more isosurface triangles than MARCHING CUBES or dual contouring and creates non-manifold "bubbles" whenever a grid facet has four bipolar edges (i.e., the grid facet is ambiguous). Properties and proofs of SPIDERWEB isosurfaces are discussed in [Karron and Cox, 1992] and [Karron et al., 1993].

Dual contouring algorithms, isosurface construction in four dimensions, and interval volumes are in the notes and comments sections of Chapters 3, 6, and 7.

Approximating isosurface normals from volumetric data is discussed in [Cline et al., 1988, Möller et al., 1997a, Möller et al., 1997b, Nielson et al., 2002, Lee et al., 2008, Hossain et al., 2011]. Approximating Gaussian and mean curvature is discussed in [Nielson, 2003a, Kindlmann et al., 2003].

Patera and Skala [Patera and Skala, 2004] compare the accuracy of the MARCHING CUBES isosurface with the accuracy of isosurfaces extracted from tetrahedralizations of the regular grid and with isosurfaces extracted from the body-centered cubic lattice.

Etiene et al. [Etiene et al., 2009] describe techniques for assessing the correctness of isosurface extraction programs. Pöthkow and Hege [Pöthkow and Hege, 2011] give a method for computing the positional uncertainy of isosurface vertices. They represent this uncertainty by coloring the isosurface.

Nooruddin and Turk [Nooruddin and Turk, 2003] use isosurfaces to repair polygonal models. They convert the polygonal model to a volumetric data set. Grid vertices with scalar value 1 lie inside the model while grid vertices with scalar value 0 lie outside. They extract a MARCHING CUBES isosurface with scalar value between 0 and 1. As noted in Section 2.3.3, this isosurface is a manifold. Ju [Ju, 2004] gives a variation of this algorithm using dual contouring. Ju's approach detects and patches hole boundaries. Bischoff, Pavic, and Kobbelt [Bischoff et al., 2005] repair polygonal models by using morphological operators to close the holes. They also use dual contouring to reconstruct the polygonal surface. Ju in [Ju, 2009] surveys techniques for repairing polynomial models.

Newman and Yi in [Newman and Yi, 2006] provide an excellent survey of the MARCHING CUBES algorithm and its variants.