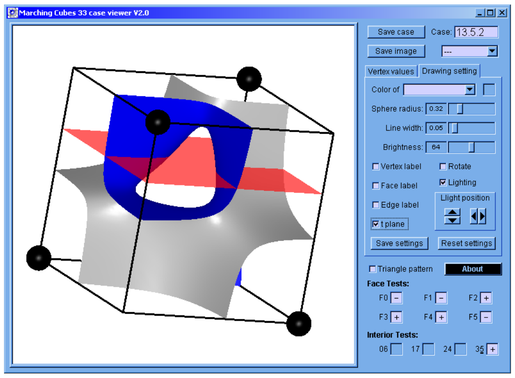
A Fast and Memory-Saving Marching Cubes 33 Implementation with the Correct Interior Test

David Vega M.
Universidad de Carabobo

Javier Abache R.
Universidad de Carabobo

David Coll G.
Instituto Venezolano de Investigaciones Científicas



Case: 13.5.2
Color of
Sphere radius: 0.32
Line width: 0.05
Brightness: 64
☐ Vertex label ☐ Rotate
☐ Face label ☑ Lighting
☐ Edge label
☑ t plane
☐ Triangle pattern
Face Tests:
F0 − F1 − F2 +
F3 + F4 + F5 −
Interior Tests:
06 17 24 35 +

Figure 1. Screenshot of "Marching Cubes 33 case viewer" application.

Abstract

The marching cubes (MC) algorithm, which generate isosurfaces from a regular 3D grid, is widely used in visualization of volume data from a CT scan, MRI, X-ray diffraction, quantum mechanical calculation, or mathematical function. In this paper, we introduce a C-language code for the marching cubes 33 (MC33) algorithm. The routines of this implementation were optimized for fast execution and low memory consumption. The array of triangles, point coordinates, and normal vectors of the generated surfaces do not require an extensive continuous memory, which improves the memory allocation. The interior test was properly performed considering that one of the four main diagonals of the grid cube must first be selected.

1. Introduction

Marching cubes (MC) [Lorensen and Cline 1987] is the best-known method to build isosurfaces, starting from a scalar function ($F(x, y, z) = \alpha$, α is the isovalue) evaluated at the array of uniformly distributed points in a regular three-dimensional grid. The isosurface consists of a set of triangles in which the vertices are located on the edges of each grid cell (cube), and their coordinates are obtained by linear interpolation along each cube edge.

In this method, one cube is processed at a time. Each cube vertex is classified as positive or negative when its values is greater, respectively, smaller than the given isovalue. If the eight cube vertices have the same sign, the isosurface does not intersect the cube. Otherwise, the intersection points between the edges and the isosurface must be determined. These intersection points are the vertices of the triangles that form the isosurface. The way to connect the vertices is obtained from a triangle pattern stored in a look-up table (LUT).

The LUT contains all of the possible options, and each entry has a triangle pattern according to the way in which the isosurface and the grid cube intersect. There are $2^8 = 256$ possible options (the eight cube vertices can be positive or negative). However, the number of options can be reduced to 128, since the options are equivalent when the signs of all cube vertices are inverted. In the original MC [Lorensen and Cline 1987], only those options with a maximum of four positive vertices were considered. By grouping the options that are equivalent when applying a rotation operation, they were classified in 15 cases, shown in Figure 2.

A main weakness of the MC algorithm is that it can produce surfaces with holes as a result of the topological inconsistency between common faces in adjacent cubes. For each configuration, the LUT contains only one variant of the isosurface topology, while the trilinear function, Equation (1), supports multiple variants in some of these configurations. These variants appear when the two kinds of ambiguities are solved,

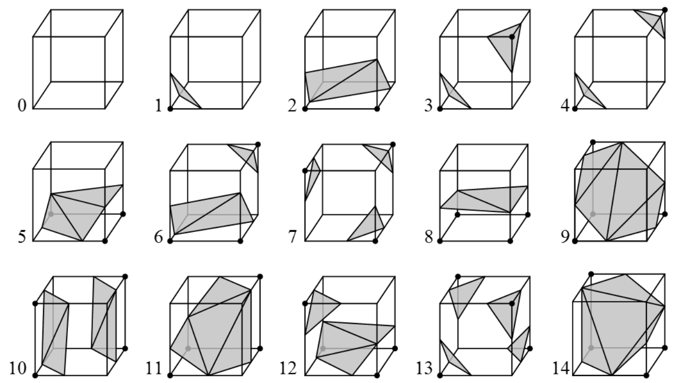


Figure 2. MC triangle patterns. The function value is greater than the isovalue in the marked vertices.

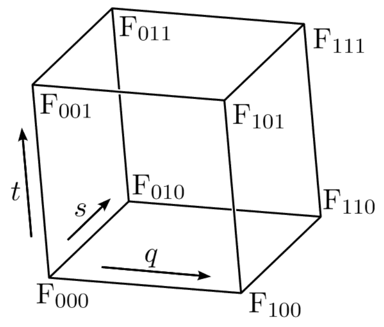


Figure 3. Parameters for the trilinear interpolation formula, Equation (1).

one on the cube faces and the other inside the cube:

$$\begin{aligned} F(q,s,t) = {} & (1-q)(1-s)(1-t)\mathrm{F}_{000} + q(1-s)(1-t)\mathrm{F}_{100} \\ & + (1-q)s(1-t)\mathrm{F}_{010} + qs(1-t)\mathrm{F}_{110} + (1-q)(1-s)t\mathrm{F}_{001} \\ & + q(1-s)t\mathrm{F}_{101} + (1-q)st\mathrm{F}_{011} + qst\mathrm{F}_{111} \end{aligned} \tag{1}$$

where F_{ijk} is the function evaluated at cube vertices and q, s, and t range from 0 to 1 (see Figure 3).

The ambiguity problem for the faces was solved using bilinear interpolation [Nielson and Hamann 1991]; as a result the surfaces are topologically consistent and 20 cases were added to the 15 initial ones. Some of the added patterns never occur when a trilinear approximation is used for describing the scalar function (F), as shown by Chernyaev [1995]. Additionally, in the Chernyaev paper, the internal ambiguity was solved and the number of cases was increased to 33 (MC33).

There is another method similar to MC33 to resolve the ambiguities, which is also based on a trilinear approximation, increasing the number of cases to 34 [Nielson 2003]; however, the described tests are more complex.

In the present work, the MC33 algorithm was programmed with the purpose of generating isosurfaces of molecular and crystalline electronic charge density. The first version of this implementation of MC33 was finalized in 2008. In 2012, an article was published showing the errors in all the MC implementations available on the web [Etiene et al. 2012]. At that time, we also decided to make our implementation available on the web. This version can still be obtained at http://alfa.facyt.uc.edu.ve/quimicomp/Descargar/marching_cubes_33.7z. Currently, we have optimized the algorithm to reduce the processing time. The latest version can be downloaded from: https://sourceforge.net/projects/facyt-quimicomp/files/marching_cubes_33_c_library.7z.

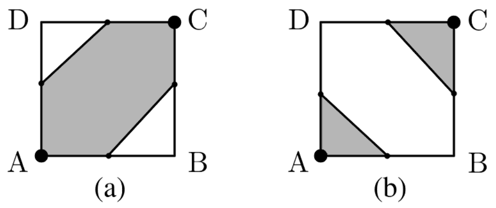


Figure 4. Two ways to connect the intersection points of the face edges with the isosurface (small black dots). If the inequality Equation (2) is valid, it must be connected like (a), one positive area (gray filled), or else like (b), separated positive areas.

2. Background

For simplicity, we set the isovalue to zero. This is equivalent to subtracting the isovalue from the scalar function evaluated at each grid point.

The ambiguity on a face arises when the values of two opposite vertices are positive (greater than the isovalue) and the other two are negative (smaller than the isovalue), which implies that the isosurface intersects the four edges. There are two ways to connect the points of intersection of the isosurface with the face edges. This ambiguity is resolved by verifying the inequality (2) [Chernyaev 1995].

$$\mathrm{AC} - \mathrm{BD} > 0, \tag{2}$$

where A, B, C, and D are the function values at the face vertices. If the inequality (2) is valid, then the A and C vertices are joined, otherwise the vertices are separated by the isosurface (see Figure 4). An internal ambiguity arises when two opposite vertices (which have the same sign) of a cube main diagonal are separated on the faces, but they can be joined inside the cube. This ambiguity can also be resolved, by applying the interior test [Chernyaev 1995].

The function F varies bilinearly over a plane parallel to a cube face. If we fix any variable (q, s, or t) in Equation (1), for example t, then F takes the form

$$F(q,s) = (1-q)(1-s)\mathrm{A}_t + (1-q)s\mathrm{B}_t + qs\mathrm{C}_t + q(1-s)\mathrm{D}_t \tag{3}$$

where

$$\begin{aligned} \mathrm{A}_t &= (1-t)\mathrm{F}_{000} + t\mathrm{F}_{001}, \\ \mathrm{B}_t &= (1-t)\mathrm{F}_{010} + t\mathrm{F}_{011}, \\ \mathrm{C}_t &= (1-t)\mathrm{F}_{110} + t\mathrm{F}_{111}, \\ \mathrm{D}_t &= (1-t)\mathrm{F}_{100} + t\mathrm{F}_{101}. \end{aligned} \tag{4}$$

Let A_0, B_0, C_0, D_0 be the function values at the cube vertices on the face with $t = 0$, and A_1, B_1, C_1, D_1 the values at the vertices on the face with $t = 1$. Let A_0 and C_1 be the vertices that can be joined inside the cube (see Figure 5).

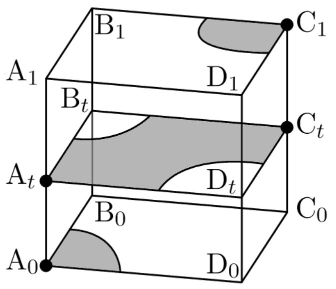


Figure 5. If the A_0 and C_1 vertices are joined inside the cube, then there is an intermediate plane (at $t = t_{\max}$) where A_t and C_t are joined (gray area).

If the A_0 and C_1 vertices are joined inside the cube, then there is some t value, between 0 and 1, such that

$$A_tC_t - B_tD_t > 0. \tag{5}$$

Applying linear interpolation (since function F varies linearly along the edges), we have

$$\begin{aligned} A_t &= A_0 + (A_1 - A_0)t, \\ B_t &= B_0 + (B_1 - B_0)t, \\ C_t &= C_0 + (C_1 - C_0)t, \\ D_t &= D_0 + (D_1 - D_0)t. \end{aligned} \tag{6}$$

Substituting Equation (6) in Equation (5), a second-degree equation is obtained:

$$at^2 + bt + c > 0, \tag{7}$$

where

$$\begin{aligned} a &= (A_1 - A_0)(C_1 - C_0) - (B_1 - B_0)(D_1 - D_0), \\ b &= C_0(A_1 - A_0) + A_0(C_1 - C_0) - D_0(B_1 - B_0) - B_0(D_1 - D_0), \\ c &= A_0C_0 - B_0D_0. \end{aligned} \tag{8}$$

Calculating a, b, and c values, Equation (8), it is possible to know whether or not the diagonally opposite vertices are joined inside the cube. The following conditions must be verified:

- whether the parabola described by Equation (7) opens downward ($a < 0$),
- whether $t = t_{\max} = -b/2a$ belongs to the interval (0, 1),
- whether A_t and C_t have the same sign and the inequality (5) is valid,
- whether the sign of A_0 or C_1 are the same as A_t and C_t.

If any of these conditions is not verified, the vertices A_0 and C_1 are separated.

The interior test can also be carried out using a plane perpendicular to the s- or q-axis (Figure 3). However, the test gives the same result independently of the chosen axis, except for the case 10.1 of MC33, which will be discussed later.

It is important to remark that the Chernyaev's interior test [1995] describes a method to check if the A_0 and C_1 vertices are connected through a tunnel. If it is intended to test another pair of opposite vertices, the sign of some parameters and the sign of comparisons could change (this will be described in Section 4.1). The interior test has been misinterpreted [Lewiner et al. 2003], and it has even been erroneously reported that this test fails [Custodio et al. 2013].

3. Triangle Patterns of Other Implementations

Some MC algorithm implementations can be downloaded in the C [Bourke 1994] and Java [Lingrand et al. 2002] languages, with the proper topological inconsistencies of this algorithm. A previous implementation of the MC33 algorithm was published in C++, which includes the source code [Lewiner et al. 2003]. However, the interior test was not programmed correctly, and a later published correction [Custodio et al. 2013] (C-MC33) is also defective.

We found two errors in the triangle patterns obtained from C-MC33. The first one is for the case 13.5.2 of MC33[1]. This only occurs when the vertex [1, 1, 0], the upper-right vertex not marked in Figure 6 (a), joins the center of the cube and its value is negative. The C-MC33 implementation correctly recognizes the case, but the triangle pattern is incorrect (see Figure 6). When the signs of all vertices are inverted, C-MC33 displays the correct pattern.

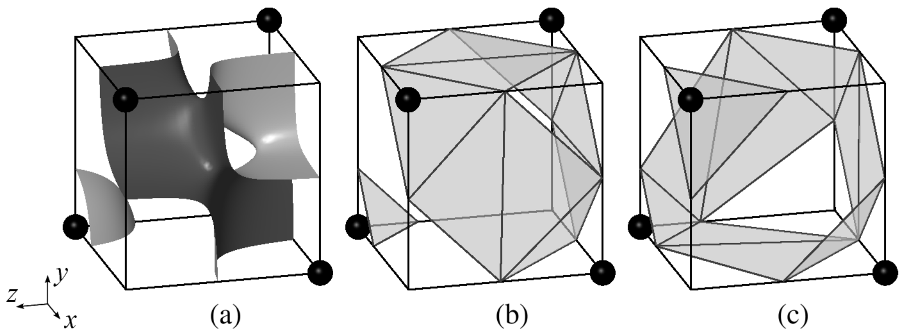


Figure 6. The MC33 case 13.5.2. (a) Trilinear function isosurface with isovalue equal to zero; the vertices with black spheres have positive values; (b) Correct MC33 triangle pattern for the isosurface (a); (c) triangle pattern obtained by C-MC33.

[1]One to three numbers are used to label the MC33 cases [Chernyaev 1995]. The first number corresponds to the original MC case (see Figure 2), the second is the number of one of the variants obtained when the vertices of ambiguous faces are joined in different ways. When the internal ambiguity exists, the third number is 2 if the diagonal vertices are joined inside the cube and 1 if they are not.

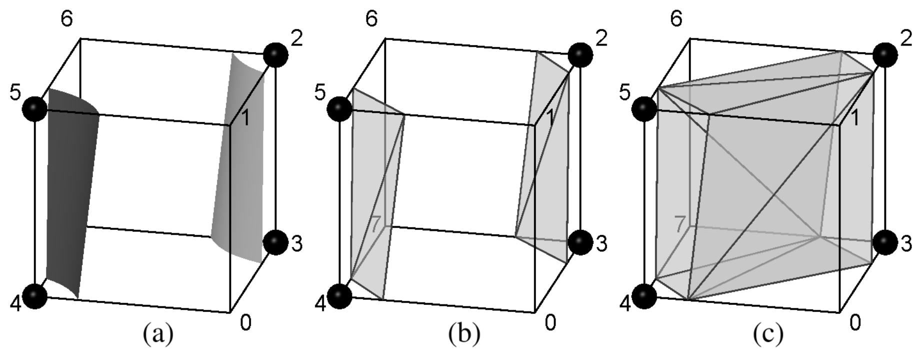


Figure 7. The MC33 case 10.1.1. (a) Trilinear function isosurface with isovalue equal to zero where the values of the vertices are {–7,–6, 2, 3, 2, 3,–7,–6}; (b) correct MC33 triangle pattern for the isosurface (a); (c) triangle pattern obtained by C-MC33.

The other error is related to the interior test. For some vertex values, C-MC33 identifies incorrectly the case 10.1.1 (negative interior test, see Section 4.2) as 10.1.2 (positive interior test). For example, this happen for the following vertex values: {–7,–6, 2, 3, 2, 3,–7,–6} and {–12,–10, 2, 3, 4, 6,–6,–5} (see Figure 7 for the vertex labels).

Another tested MC implementation is TMC [Grosso 2016]. This algorithm is based on the asymptotic decider and the interior ambiguity is solved by calculating internal points. This improves the adjustment of the triangles to the isosurface of the trilinear function, but increases the number of vertices and triangles and, as a consequence, the calculation time. This article has a good summary of the various methods proposed to deal with the ambiguities. The code for TMC is available online [Grosso 2017].

In most cases, TMC does not produce any triangle pattern if the face test can not predict whether the positive or negative vertices are joined. When the expression on the left side of inequality (2) is equal to zero, the positive and negative areas on the face are separated by two perpendicular lines where the function value is zero. The algorithm could connect any of the both pairs of opposite vertices (see Figure 8), but TMC indicates an error and aborts the determination of the triangle pattern. This condition is uncommon and is more frequent in grids from MRI and CT scans. For the Stag beetle dataset [Gröller et al. 2005] with an isovalue equal to 500, TMC shows the error 48 times and does not determine the triangle pattern in 35 cubes.

Although there is no doubt about the triangle pattern for case 10.2 of MC33, when the values at the vertices make the surface quite symmetrical (see Figure 9), TMC indicates an error and does not determine the triangle pattern.

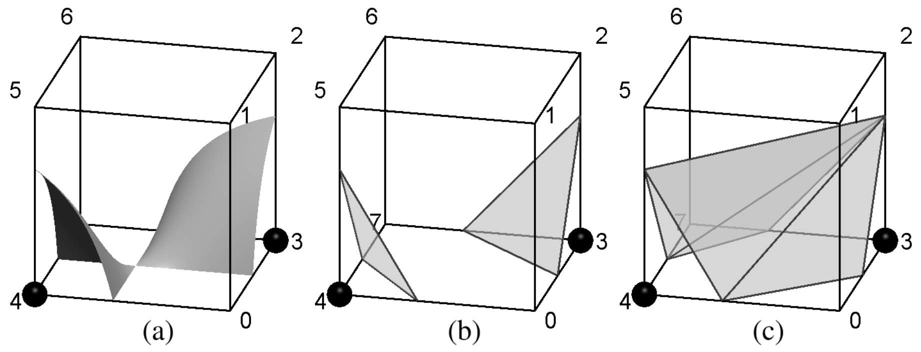


Figure 8. MC33 case 3.1 or 3.2. For the vertex values {–6,–3,–3, 6, 4,–2,–2,–4}, (a) is the isosurface of trilinear function with isovalue zero and the product of opposite vertex values on the bottom face are equal. In the MC33 triangle pattern, the connected vertices can be 0 and 7, as in (b) (case 3.2), or 3 and 4, as in (c) (case 3.1). Either case is acceptable. For this configuration TMC displays an empty cube.

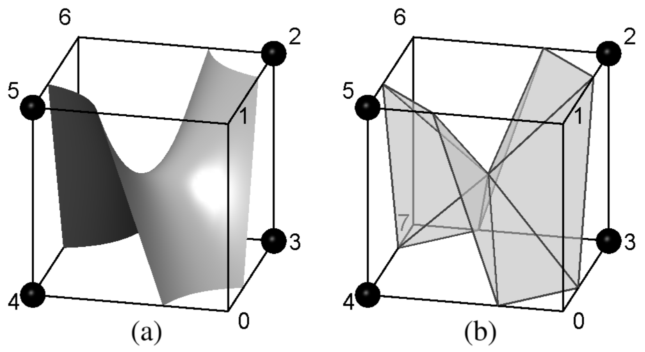


Figure 9. For the vertex values {–4,–9, 5, 8, 8, 5,–9,–4}, (a) is the isosurface of trilinear function with isovalue zero and (b) is the MC33 triangle pattern (case 10.2). When the values of opposite vertices on each face where the face test must be applied (top and bottom faces in this example) are equal, TMC displays an empty cube.

4. Algorithm

4.1. Interior Test

Although the interior test of MC33 was described in 1995 [Chernyaev 1995], no other versions of this algorithm are available with this test correctly implemented.

For the interior test, it is assumed that the diagonally opposite vertices have the same sign (except for the case 13.5). There are four main diagonals in the cube; in other words, there are four pairs of opposite vertices and, at least, four different ways to apply the interior test. In the implemented algorithm, the diagonal must be specified to perform the interior test. The notation of Figure 5 will be used for the vertices of the cubes in the four cases described below, and only one plane (of the three possible orientations) will be used.

The algorithm of the interior test contains some common steps and other steps that depend on the chosen diagonal:

The a and b values are calculated by Equation (8). If a is zero, the vertices will not be joined, and if it is not zero then $t = t_{\max} = -b/2a$. If the t value is outside of interval (0, 1), the vertices will not be joined. If t is inside, then A_t, B_t, C_t, and D_t are calculated by Equation (6), and continues with one of the following cases (depending on the selected diagonal):

Case 0: Test for A_0 and C_1 vertices. The vertices are joined if all the following conditions are verified: $a < 0$, $A_t\,C_t > B_t\,D_t$, and A_t, C_t, and A_0 have the same sign.

Case 1: Test for B_0 and D_1 vertices. The vertices are joined if all the following conditions are verified: $a > 0$, $A_t\,C_t < B_t\,D_t$, and B_t, D_t and B_0 have the same sign.

Case 2: Test for C_0 and A_1 vertices. The vertices are joined if all the following conditions are verified: $a < 0$, $A_t\,C_t > B_t\,D_t$, and A_t, C_t, and C_0 have the same sign.

Case 3: Test for D_0 and B_1 vertices. The vertices are joined if all the following conditions are verified: $a > 0$, $A_t\,C_t < B_t\,D_t$, and B_t, D_t, and D_0 have the same sign.

In our implementation, most of the triangle patterns of the LUT were ordered based on the diagonal index that must be passed to the interior test function. In the table of case 12, an auxiliary array of diagonal indices was used.

4.2. Case 10.1.1 and 10.1.2

In cases 10.1.1 and 10.1.2, there are two ambiguous faces which are opposite faces of the cube. The cube vertices are joined in the same way on those faces, and the interior test must be used to discriminate between the two cases. In contrast to the other cases, the interior test can be applied to two diagonals. An additional complication arises when the plane containing both diagonals is not perpendicular to the plane t (A_t, B_t, C_t, D_t), in which case although both diagonals are joined inside of the cube (case 10.1.2), one test will be negative and the other positive. In fact, when projecting the involved vertices on the plane t, the points A_t and C_t for one of the diagonals, and the points B_t and D_t for the other one are obtained. As shown in Figure 10, if A_t and C_t are joined, then B_t and D_t are not, and vice versa. For one of the diagonals, the test fails. Nevertheless, if the plane containing both diagonals is perpendicular to the

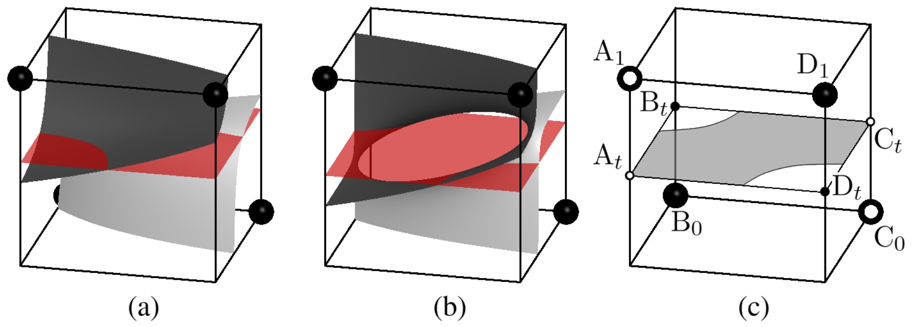


Figure 10. Examples of trilinear function isosurfaces for cases 10.1.1 and 10.1.2 are shown in (a), respectively, (b). The plane at $t = t_{\max}$ is shown in red in both cubes. In (c), the plane of (b) was drawn, filling the area where the function is positive in gray. The A$_1$ and C$_0$ vertices of one main diagonal of the cube are projected onto the plane at A$_t$, respectively, C$_t$, and similarly, the B$_0$ and D$_1$ vertices at B$_t$, respectively, D$_t$. For (b), the interior test will be positive if it is applied to the diagonal (A$_1$, C$_0$) and negative for the (B$_0$, D$_1$).

plane t, then the involved vertices will be projected on only a pair of points, A$_t$, C$_t$, or B$_t$, D$_t$, and the result of the test will be the same for both diagonals. See Figure 11.

In summary, if the plane containing the two diagonals to be tested is not perpendicular to the plane t (Figure 10), the interior test is considered positive if the result is positive for any of the diagonals. If the plane of the diagonals is perpendicular to the plane t (Figure 11), it is sufficient to test only one diagonal.

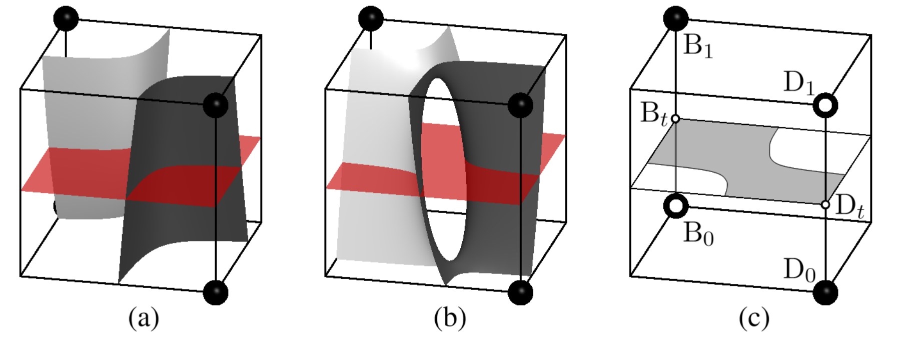


Figure 11. Examples of trilinear function isosurfaces for cases 10.1.1 and 10.1.2 are shown in (a), respectively, (b). The plane at $t = t_{\max}$ is shown in red in both cubes. The same plane of (b) is shown in (c). When projecting the vertices of the diagonals (B$_0$, D$_1$) and (B$_1$, D$_0$) onto the plane, only the B$_t$ and D$_t$ points are obtained. For (b), the interior test will be positive for both these diagonals.

4.3. Case 13.5.1 and 13.5.2

In the LUT, case 13 has the most number of variants. It is possible to choose some variants with only the face tests. The subcase 13.5 (positive face tests for three faces with a common vertex and negative for the other three faces) has two variants that depend on the result of the interior test: negative for 13.5.1 and positive for 13.5.2 [Chernyaev 1995]. Despite the fact that the interior test was deduced for a case where a couple of diagonally opposite vertices have the same sign, and that in the case 13, all the opposite vertices have different signs, the interior test can be applied. Although the interior test determines which variant (13.5.1 or 13.5.2) must be selected, still an ambiguity persists: There are two versions for the 13.5.2 variant, Figures 12(b) and 12(c).

The interior test function was modified by adding a flag, equal to 1 when the test is applied to case 13, and 0 to other cases. The function returns the value 0 if some of the described conditions in Section 4.1 are not valid, except when comparing the sign between one of the vertices (A_0, B_0, C_0 or D_0) of a diagonal and the corresponding point on the plane t (A_t, B_t, C_t, D_t). Then, if the flag is 1, the function returns 1 when the signs are different, and returns 2 if they are equal. A 0 value indicates that both vertices of the diagonal are not joined to the cube center region (case 13.5.1,

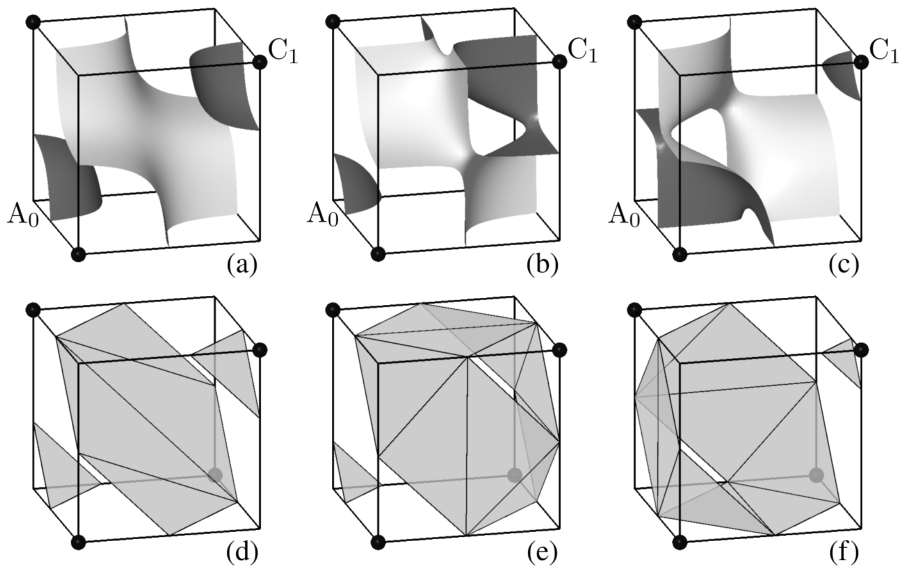


Figure 12. The 13.5.1 variant, (a) and (d), and the 13.5.2 variants (b), (c), (e) and (f). The figures (a), (b) and (c) are isosurfaces from trilinear functions, and (d), (e) and (f) are the respective triangle patterns. The marked vertices correspond to positive function values. The results of the face tests are the same for (a), (b) and (c). The interior test result is 0 for (a), 1 for (b), and 2 for (c).

Figure 12(a)); a value of 2 indicates that the A_0, B_0, C_0, or D_0 vertex is joined to the center (case 13.5.2, Figure 12(c)), and a value of 1 means that the opposite vertex (C_1, D_1, A_1, or B_1) is joined to the center (Figure 12(b)).

5. Implementation Details

Some important details of our marching cubes implementation are described here. The triangle patterns stored in the LUT (MC33_LookUpTable.h) were built using Maple 7, starting from a pattern for each variant and automatically generating the other ones. For each of the 24 rotation operations of the cubic system, three matrices were built: one for the edges (to obtain the triangle pattern), another for the vertices (cube index), and the third for the faces (face test results). The repeated triangle patterns were discarded. For proper display of surfaces, all the triangle vertices were placed in the LUT in a determined order.

In addition to the triangle patterns, the algorithm computes a unit vector normal to the isosurface at each triangle vertex, using trilinear interpolation. This vector is useful to shade the surface. The fast inverse squared root algorithm [Lomont 2003] was used to normalize the vector.

In a similar way to other marching cubes implementations [Lewiner et al. 2003; Watt and Watt 1992], auxiliary structures are used to prevent calculating the vertices more than once. Unlike Lewiner's implementation [2003], the grid is walked only once, and the auxiliary structures require only a small fraction of memory compared to that occupied by the grid. The indices of the vertices lying on the edges parallel to the x- and y- axes are stored in four matrices of integers (The size of each matrix is approximately equal to the size of a z-slice), and those corresponding to the z-axis in two vectors. Meanwhile, the cubes are analyzed, the indices are simultaneously stored and read from these structures. Finally the allocated memory for these structures is released. For the bunny dataset [Yoo 2000], the memory occupied by the grid and the auxiliary structures is 374 MiB when using our code and about 1440 MiB using the C-MC33 [Custodio et al. 2013] or TMC [Grosso 2017] code. For the stag beetle dataset [Gröller et al. 2005], the memory occupied in our implementation is 1.3 GiB vs. 5.1 GiB for the other implementations.

When a vertex value is zero, the intersection point with the surface is the same vertex. It is possible that this point is stored more than once, as belonging to more than one edge of the cube. In this algorithm, when one of these intersection points is stored, its index is stored in the temporary structures at the positions corresponding to the other edges (which have that vertex in common), preventing it from being recalculated when analyzing another of the edges. And before storing the triangles, the vertex indices are verified, discarding the triangles with repeated vertices (zero area).

For most datasets, the number of cubes intersected by the surface is quite low. For example, for the bunny dataset with an isovalue of 1500 or 1500.5, the isosurface only goes through 1,703,733 cubes out of a total of 94,003,560 (1.81%). For the stag beetle dataset with a isovalue of 500, 3,760,897 cubes from 340,446,573 (1.10%) are intersected by the isosurface. In this implementation some tricks have been used to reduce the time spent in moving through the cubes: The way of labeling the vertices was changed, the vertices from 0 to 3 are in a plane parallel to the yz-plane, and also the vertices from 4 to 7, but with a different x-value. In addition two pointers were created, each one pointing to an array of four real values (float), where the differences of each group of vertices with the isovalue are stored. For each increment in the innermost counter (x-axis), the pointers are exchanged, so the vertices from 4 to 7 of the previous cube are converted into the vertices from 0 to 3 of the new cube, and the difference is not recalculated. The cube index value (eight bits corresponding to the cube vertices, each bit is 1 if the vertex value is greater than the isovalue, else 0) was calculated with the `signbit` function. Additionally, the bit shift operator was used to move the bits from the position corresponding to the vertices 4 to 7 to the position corresponding to the vertices 0 to 3, when the next cube index value is calculated.

To inspect the MC33 case identification, the corresponding triangle patterns, and the test results, the Marching Cubes 33 case viewer application (Figure 1) was used; it is available at https://facyt-quimicomp.neocities.org/Vega_en.html#case_viewer. By using this software, the results of the interior test were extensively verified, and with a modified version (to include the triangle patterns of other implementations), Figures 6 to 12 were generated. The Marching Cubes 33 case viewer was programmed in C++ using the FLTK (http://www.fltk.org/) and OpenGL libraries.

To avoid using large continuous memory space for the surface data, the algorithm uses an array with an extra index. The point and normal coordinates, the color, and the triangle indices are stored in groups of 4096 elements (default value, always a power of two), and the extra index changes when the block of 4096 elements is filled. This algorithm is able to handle and display isosurfaces with more than a million vertices on a personal computer with only 512 MiB RAM.

The data type of the grid data stored in memory can be changed, before compiling the code, by defining `GRD_data_type`. The default value is float.

Descriptions for the grid data (`_GRD`) and surface (`surface`) structures, useful functions, and usage of our Marching cubes 33 version are included in the source code in the supplementary materials (`http://jcgt.org/published/0008/03/01/marching_cubes_33.zip`). The library has been successfully compiled in Windows, macOS and Linux. The makefile is included.

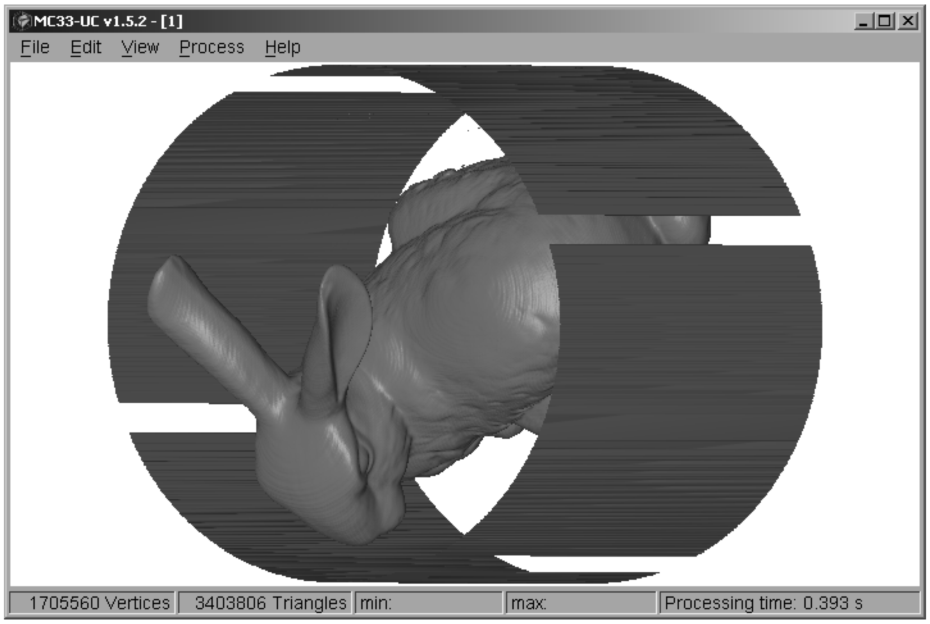


Figure 13. Surface with isovalue of 1500 extracted from the bunny dataset.

6. Testing the Implementation

The implementation was tested on two desktop computers, one with an Intel® Core™ i7-4790 CPU @ 3.60 GHz (16 GiB RAM), Windows 10 64-bit operating system, and the other with an Intel® Core™ i7-2600 CPU @ 3.40 GHz (8 GiB RAM), Debian 8.10 64-bit operating system. The algorithm was not parallelized.

To compare with other implementations, the bunny and the stag beetle dataset were used. The bunny dataset has $512 \times 512 \times 361$ points, and it was obtained from the Stanford volume data archive [Yoo 2000]. The stag beetle dataset has $832 \times 832 \times 494$ points, and it was obtained from the Institut für Computergraphik und Algorithmen [Gröller et al. 2005]. Routines for reading these datasets are included in the source code. An application (MC33-UC) that uses our library was also programmed. Figures 13 and 14 are screenshots of this application. Figure 13 shows the isosurface of the bunny dataset with isovalue of 1500, and Figure 14 shows the isosurface of the stag beetle dataset with isovalue of 500. Information about MC33-UC and the download link are available at https://facyt-quimicomp.neocities.org/Vega_en.html#MC33_UC.

Tables 1 and 2 compare the processing times of our MC33 against the C-MC33 [Custodio et al. 2013] and TMC [Grosso 2017], executed on the desktop computers described above. The code was compiled with g++ (GCC 7.2 release for Debian PC and MinGW-w64 7.2.0 for Windows PC), using the -Ofast compiler option. It can be seen that there are fewer vertices and triangles in our implementation. When the data set is composed of integers, unlike the other implementations, the elimination of re-

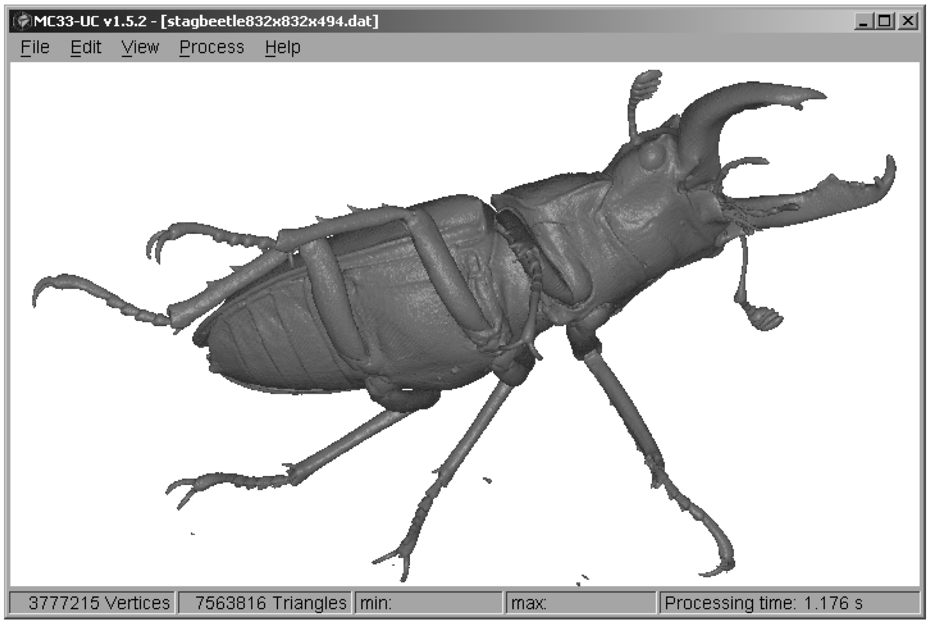


Figure 14. Surface with isovalue of 500 extracted from the stag beetle dataset.

peated vertices and zero-area triangles decreases the number of vertices and triangles if the isovalue is also an integer (1500 vs. 1500.5 for bunny and 500 vs. 500.5 for stag beetle). With an isovalue greater than all values in the dataset (100000) it is possible to evaluate the time used to go through all the cubes only calculating their indices. The processing speed of our implementation is at least 5 times faster than that of the other two implementations.

Code	Isovalue	Vertices	Triangles	Time A* (s)	Time B* (s)
C-MC33	1500	1707100	3406866	1.984	3.919
	1500.5	1706994	3406670	1.984	3.920
	100000	0	0	1.687	3.556
TMC	1500	1707191	3407048	3.025	5.630
	1500.5	1707088	3406858	3.019	5.656
	100000	0	0	2.254	4.630
Our MC33	1500	1705560	3403806	0.391	0.564
	1500.5	1706994	3406670	0.390	0.563
	100000	0	0	0.227	0.387

*Average of 30 values.

Table 1. Comparison of isosurface processing from the bunny dataset for MC implementations on (A) a computer with an Intel® Core™ i7-4790 CPU @ 3.60 GHz, Windows 10 OS and (B) another computer with an Intel® Core™ i7-2600 CPU @ 3.40 GHz, Debian 8.10 OS.

Code	Isovalue	Vertices	Triangles	Time A* (s)	Time B* (s)
C-MC33	500	3793839	7596862	6.632	13.49
	500.5	3792409	7594102	6.624	13.50
	100000	0	0	6.129	12.92
TMC	500	3809556	7628004	10.04	19.72
	500.5	3808140	7625411	10.04	19.72
	100000	0	0	8.160	16.77
Our MC33	500	3777215	7563816	1.171	1.802
	500.5	3792298	7593876	1.168	1.799
	100000	0	0	0.812	1.402

*Average of 30 values.

Table 2. Comparison of isosurface processing from the stag beetle dataset for MC implementations on (A) a computer with an Intel® Core™ i7-4790 CPU @ 3.60 GHz, Windows 10 OS and (B) another computer with an Intel® Core™ i7-2600 CPU @ 3.40 GHz, Debian 8.10 OS.

7. Conclusions

In this paper, we offer an implementation of the Marching Cubes 33 algorithm as a C library, with the following characteristics.

- The interior test was correctly performed.
- The generated surfaces do not have repeated vertices or zero area triangles.
- The implementation does not require extensive memory, its processing times are fast, and it can be adapted to several grid data types.

The application used to test our implementation (Marching Cubes 33 case viewer) and an application that uses the library (MC33-UC) are available on the website: https://facyt-quimicomp.neocities.org/Vega_en.html

The library source code can be obtained from the website: `http://jcgt.org/published/0008/03/01/marching_cubes_33.zip)`

References

Bourke, P., 1994. Polygonising a scalar field. URL: `http://paulbourke.net/geometry/polygonise/`. 6

Chernyaev, E. 1995. Marching cubes 33: Construction of topologically correct isosurfaces. Tech. rep., Institute for High Energy Physics, 142284, Protvino, Moscow Region, Russia. URL: `https://pdfs.semanticscholar.org/7d4b/30bbdeb1d55614e89588b7d9641c9b5a7a78.pdf`. 3, 4, 6, 8, 11

Custodio, L., Etiene, T., Pesco, S., and Silva, C. 2013. Practical considerations on marching cubes 33 topological correctness. *Computers & Graphics 37*, 7, 840–850. URL: `https://github.com/rekka/C_MC33`, doi:10.1016/j.cag.2013.04.004. 6, 12, 14

ETIENE, T., NONATO, L. G., SCHEIDEGGER, C., TIENRY, J., PETERS, T. J., PASCUCCI, V., KIRBY, R. M., AND SILVA, C. T. 2012. Topology verification for isosurface extraction. *IEEE Transactions on Visualization and Computer Graphics 18*, 6, 952–965. URL: https://www.cs.utah.edu/~kirby/Publications/Kirby-58.pdf, doi:10.1109/TVCG.2011.109. 3

GRÖLLER, M. E., GLAESER, G., AND KASTNER, J., 2005. Stag beetle dataset. URL: http://www.cg.tuwien.ac.at/research/publications/2005/dataset-stagbeetle/. 7, 12, 14

GROSSO, R. 2016. Construction of topologically correct and manifold isosurfaces. *Computer Graphics Forum 35*, 5, 187–196. URL: https://doi.org/10.1111/cgf.12975, doi:10.1111/cgf.12975. 7

GROSSO, R. 2017. An asymptotic decider for robust and topologically correct triangulation of isosurfaces: topologically correct isosurfaces. In *Proceedings of the Computer Graphics International Conference*. ACM, New York, NY, USA, 39:1–39:5. URL: https://github.com/rogrosso/tmc, doi:10.1145/3095140.3095179. 7, 12, 14

LEWINER, T., LOPES, H., VIEIRA, A. W., AND TAVARES, G. 2003. Efficient implementation of marching cubes' cases with topological guarantees. *Journal of graphics tools 8*, 2, 1–15. URL: https://www.tandfonline.com/doi/abs/10.1080/10867651.2003.10487582?journalCode=ujgt19, doi:10.1080/10867651.2003.10487582. 6, 12

LINGRAND, D., CHARNOZ, A., GERVAISE, R., AND RICHARD, K., 2002. The marching cubes tutotial: Polygonising a scalar field. URL: http://users.polytech.unice.fr/~lingrand/MarchingCubes/applet.html. 6

LOMONT, C. 2003. Fast inverse square root. *Technical Report*, 32. URL: http://www.lomont.org/Math/Papers/2003/InvSqrt.pdf. 12

LORENSEN, W. E., AND CLINE, H. E. 1987. Marching cubes: A high resolution 3D surface construction algorithm. *ACM SIGGRAPH Computer Graphics 21*, 4, 163–169. doi:10.1145/37401.37422. 1, 2

NIELSON, G. M., AND HAMANN, B. 1991. The asymptotic decider: resolving the ambiguity in marching cubes. In *Proceedings of the 2nd conference on Visualization'91*. Los Alamitos, CA, 83–91. 3

NIELSON, G. M. 2003. On marching cubes. *IEEE Transactions on visualization and computer graphics 9*, 3, 283–297. doi:10.1109/TVCG.2003.1207437. 3

WATT, A. H., AND WATT, M. 1992. *Advanced animation and rendering techniques*. ACM Press, New York, NY, USA, 313–321. 12

YOO, T., 2000. Bunny dataset. URL: http://graphics.stanford.edu/data/voldata/voldata.html#bunny. 12, 14

Author Contact Information

David Vega M.
Laboratorio de Química Computacional
FACYT – Universidad de Carabobo
Edo. Carabobo – Venezuela
dvega@uc.edu.ve, dvega68@yahoo.com
https://facyt-quimicomp.neocities.org/Vega_en.html

Javier Abache R.
Laboratorio de Química Computacional
FACYT – Universidad de Carabobo
Edo. Carabobo – Venezuela
jabache81@yahoo.es

David Coll G.
Laboratorio de Fisicoquímica Teórica
de Materiales
Centro de Química
Instituto Venezolano de Investigaciones
Científicas
Caracas – Venezuela
dsantiagocoll@gmail.com

Vega, D., Abache, J., Coll, D., A Fast and Memory-Saving Marching Cubes 33 Implementation with the correct interior test, *Journal of Computer Graphics Techniques (JCGT)*, vol. 8, no. 3, 1–18, 2019
http://jcgt.org/published/0008/03/01/

Received:	2018-07-02		
Recommended:	2019-05-16	Corresponding Editor:	Joe Geigel
Published:	2019-08-08	Editor-in-Chief:	Marc Olano