A marching algorithm for isosurface extraction
from face-centered cubic lattices

Yusuf SAHİLLİOĞLU

Computer Engineering Department, METU, Ankara, Turkey
ys@ceng.metu.edu.tr

Abstract: This work provides a novel method that extracts isosurfaces from Face-Centered Cubic (FCC) lattices. It has been theoretically shown that sampling volumetric data on an FCC lattice tiled by rhombic dodecahedra is more efficient than sampling it on a Cartesian lattice tiled by cubes, in that FCC lattice is capable of representing the same dataset as a Cartesian lattice with the same accuracy, but with about 23% fewer samples. This fact, coupled with the nice properties of rhombic dodecahedra, encouraged us to develop this related isosurface extraction technique. Thanks to the sparser sampling required by the FCC lattices, the de-facto standard isosurface extraction algorithm, namely the Marching Cubes, is accelerated significantly, as demonstrated. This reduced sampling rate also leads to a decrement in the number of triangles of the extracted models when compared with the Marching Cubes result. Finally, topological consistency problem of the original Marching Cubes algorithm is also resolved. We show the potential of our algorithm with an indirect volume rendering application.

Keywords: Indirect volume rendering, Isosurface extraction, non-Cartesian lattice, Face-Centered Cubic lattice, Marching.

1. Introduction

Implicit representation of data, either in the form of mathematical functions, e.g., a sphere, or thresholded real-world datasets, e.g., a medical image, is converted to a polygon-based
representation through the isosurface extraction operation. Isosurfaces that represent points of equal values (e.g., distance, intensity) within a volume of space can be utilized in many applications such as shape deformation, constructive solid geometry, and indirect volume rendering, the last one being our particular interest in this paper.

The visualization of volumetric datasets is typically based on the representation of the underlying point lattice. While Marching Cubes [1] has provided a simple yet successful solution to this rendering problem by using a Cartesian lattice, it has been shown that results can be enhanced drastically, both in terms of quality, efficiency, and robustness by using non-Cartesian lattices such as the Body-Centered Cubic (BCC) and Face-Centered Cubic (FCC) lattices. Next-generation volumetric data sampling schemes [2] have brought the attention to the usage of such lattices, which in a sense left the Cartesian cubic lattice used by the Marching Cubes algorithm obsolete.

The optimality of the BCC lattice has been proven by [2] by reducing the problem of sampling to a well-known mathematical problem of 3D sphere packing. BCC has almost one third (29.3%) fewer sampling density compared to the Cartesian lattice while maintaining the quality of the extracted isosurface. FCC lattice is the second best choice for this purpose, as it achieves the same accuracy of a Cartesian lattice with 23% fewer samples [3].

In this paper, we will be introducing an isosurface extraction technique to render volumetric data sampled on the FCC lattices. The main reason for preferring FCC over BCC is that, FCC achieves a similar efficiency as BCC while tiling the space with rhombic dodecahedra, which are simpler to process than the truncated octahedra used in BCC tiling. Mentioning the previous work in Section 3, we address the problem in Section 4. We present our results in Section 5, followed by the conclusion at Section 6.
2. **Contributions**

Our contributions can be summarized as follows:

- Tiling the space with rhombic dodecahedra.
- Finding the indexing of the decomposition of parallelepipeds into six tetrahedra that leads to manifold extractions.
- Developing a comparative framework with the well-known Cartesian lattices and Marching Cubes algorithm to clearly (visually and quantitatively) demonstrate the advantage of our fast non-Cartesian method.

3. **Related Work**

A preliminary isosurface extraction technique, called contour tracing, connects adjacent representative contours of the object with triangles [4]. This idea, which dates back to 1975, is now mostly ruled out due to the following problems: existence of many contours in each slice, and difficulty in tracing slices that exhibit high variances in between.

A better extraction idea is [5], which represents the volume by subdividing it into small cubes, called cuberilles. These cuberilles are centered on a single sample, and rendered only if the iso-value in their center is above the given threshold, which in turn renders only the boundary cubes that separate the inside cubes from the outside ones. This idea of cube rendering causes undesired jaggy structure in the output, as demonstrated in Figure 1-a.

An even better approximation is the Marching Cubes algorithm [1], which overcomes the staircase problem of cuberilles using linear interpolation along the edges of the cubes to be marched, as shown in Figure 1-b. This edge interpolation is possible due to the placement of samples to the cube vertices, which effectively forms the Cartesian cubic lattice. This algorithm, in fact, is so promising that it is regarded as the de-facto standard
geometry-based isosurface extraction technique, although it has ambiguous pathological cases that cause topological inconsistencies.

An analogous algorithm to [1] is Marching Tetrahedra [6], which produces, for each tetrahedron (instead of a cube), a facet approximation to the isosurface. While this choice clears the ambiguities of [1], it suffers from the excessive number of triangles produced.

In this paper, we will make use of Marching Tetrahedra as an auxiliary procedure to our global isosurface extraction algorithm, hence not suffering from the topological inconsistencies inherent to the Marching Cubes algorithm.

There are also algorithms that utilize lattices other than the Cartesian cubic lattice. They are both motivated by the analytical advantages of such non-Cartesian lattices [2]. There also exists sufficient evidence from the nature that promote the usage of these alternative non-Cartesian lattices. Honeycomb is arguably the most obvious example of a non-Cartesian lattice in nature [7]. Honey bees instinctly construct hexagonal cells that form the non-Cartesian hexagonal lattice representation of their honeycomb. An intuitive explanation of this choice is based on the fact that the hexagon covers the largest surface area for its perimeter length while tiling the 2D plane [8]. Hence, bees use the minimal amount of wax to cover the space in their honeycomb construction. In other words, they effectively optimize the volume of the honey cells. This intuition extends well to 3D and explains geometrically why non-Cartesian lattices need ~30% fewer samples (wax) than the Cartesian cubic lattice in order to represent (cover) the same volume.

For the non-Cartesian works, [9] and [10] developed marching algorithms for BCC lattices. Since the mesh that arises from BCC lattices involves a large number of cells, their work rather focuses on reducing the number of cells by clustering tetrahedra into either octahedra or hexahedra. [11] and [12], on the other hand, use FCC lattices to
leverage the sampling advantage of a non-Cartesian lattice. They, however, possess the
problem of topological ambiguity stemming from triangulating octahedral cells. [13]
alleviates such topological problems by dividing each cell into a number of tetrahedra, an
approach we also employ in our algorithm.

Applications dealing with the isosurfaces extracted from the volumetric data arise mostly
in the medicine and geometry processing domains. For the former, one may visualize and
further assess a medical condition using the intensity values specific to the organs [14-16]. For the latter, we see shape deformation [17-18] and reconstruction [19-22]
applications running on signed distance fields, as well as sampling of functions [8][23]
and boolean operations in constructive solid geometry [24-25]. We also see various other
fields, such as chemistry [26], crystallography [27], condensed matter physics [28], and
solid state physics [29], utilizing isosurfaces defined on different grid structures in order
to study the structure of atoms and molecules.

4. Algorithm

The block diagram of our overall isosurface extraction algorithm is given in Figure 2. The
continuous scalar field $f$ that implicitly represents the volumetric data of interest is
discretized in the tiling rhombic dodecahedra cells of the Face-Centered Cubic (FCC)
lattice. Our problem is to extract the isosurface, which is defined to be the zero set of

$$f: \mathbb{R}^3 \rightarrow \mathbb{R}, \text{i.e., } \{x \in \mathbb{R}^3 \mid f(x) = 0\}.$$
that one can also infer more information from the FCC lattice than he can from the
Cartesian lattice when both lattices are equipped with the same number of points.
Once the FCC lattice is constructed, we apply a marching algorithm, as described in the
sequel. Marching, which is the idea of processing each tiling polyhedron one-by-one
(hence the act of marching through the cells), combines simplicity with the high speed
stemming from the usage of lookup tables.
4.1 FCC Tiling and Marching
The tiling polyhedra of the FCC lattice is rhombic dodecahedron [30], which we obtain
by placing six cubes on the faces of a seventh central cube, and then joining the centers
of the outer cubes with the vertices of the central cube, as illustrated in Figure 4. Values
of \( f \) within rhombic dodecahedra are derived by trilinear interpolation. Since the trilinear
approximation of \( f \) is actually linear along rhombic dodecahedra edges, isosurface
corresponding to the iso-value of zero can be interpolated based on linear interpolation
when fed into a marching algorithm.
For easier visualization, we provide in Figure 5-left the 2D FCC tiling obtained by a
mapping from rhombic dodecahedron (3D) to hexagon (2D). We visualize in Figure 5-
right that each rhombic dodecahedron cell in gap-free FCC tiling is decomposed into 4
parallelepipeds. Marching through those parallelepipeds would yield the extraction of
isosurface from the 3D FCC lattice, which is our ultimate aim.
Marching directly through parallelepipeds, however, can result in undesired t-junctions
and holes due to the topological ambiguity problems inherent to the Marching Cubes
algorithm. Recall that a parallelepiped is simply a hexahedron with six parallelogram
faces, which in turn is homeomorphic to a cube that Marching Cubes algorithm can
process. To achieve robustness against such topology problems, we march through
tetrahedra obtained by further decomposing each parallelepiped as depicted in Figure 6, which creates 6 tetrahedra per parallelepiped.

4.3 Tetrahedra Alignment

Tetrahedra within a rhombic dodecahedron, shown in Figure 7-left, as well as across any given two rhombic dodecahedra, shown in Figure 7-right, must be aligned in such a consistent way that the corresponding edges of adjacent tetrahedra fall completely onto each other. A misalignment would otherwise create t-junctions and holes in the resulting isosurface. Decomposition given in Figure 6-right achieves the desired alignment. Please see also the accompanying video for this alignment.

5. Results and Discussion

We have conducted several tests in which isosurfaces extracted by our method and by the original Marching Cubes algorithm are compared, both in terms of accuracy and efficiency. An isosurface in this section is the zero set of the following function $f$:

$$f(x) = \min\{G[\text{Proj}^n(x)] - 0.5\},$$

where $\text{Proj}$ is the projection of the point $x$ to the $n^{th}$ binary silhouette image $I^n$ (0 for outside, 1 for inside), and the interpolator function $G$ is directly copied from [22]. Namely, the function $G$, taking values in $[0,1]$, is the bilinear interpolation of the sub-pixelic projection of $(x', y')$ of the point $x$. Thus, the isolevel function $f(x)$ takes value in $[-0.5, 0.5]$, and the zero crossing of this function reveals the isosurface. The value $f(x)$ is provided by the image of the silhouette that is farthest away from the point, or in other words, where $G$ assumes its minimum value. We have used silhouette images of two real-
world objects from 72 different views, each obtained by rotating the subject 5 degrees around the vertical axis, as shown in Figure 8.

We demonstrate the visual quality of the output meshes comparatively for the Cartesian cubic lattice and the Face-Centered Cubic (FCC) lattice throughout Figures 9-12. Additionally, we also quantify the mesh qualities in Tables 1-2 by measuring the compatibilities of the corresponding normals between the ground-truth surfaces, shown in Figure 8-bottom, and the resulting extracted surfaces via the following function:

\[ Q(n, n') = \left( \frac{\sum q(n_i \cdot n'_j)}{n} \right) \]

where the unit normal set \( n = \{ n_1, \ldots, n_n \} \) defines the final orientation, and vertex \( j \) of the extracted surface with the unit normal \( n'_j \) is in correspondence with the closest vertex \( i \) of the ground-truth surface whose unit normal is \( n_i \), and

\[ q(x) = 1-x \text{ if } x > 0, \ 1 \text{ if } x \leq 0 \]

is used to penalize incompatible normals. Other metrics that quantify the performance of the isosurface extractions are the resolutions of the lattices and output meshes as well as the execution times reported in Tables 1-3.

In Figure 9, we show the visual quality of the isosurface extracted from the Marching Cubes algorithm for data points sampled on the Cartesian cubic lattice in comparison with our corresponding quality from the marching algorithm for data points sampled on the FCC lattice. First thing to notice is that while the Cartesian lattice uses 9270 points (in the form of a 36x15x18 grid structure), FCC lattice employs only 6912 points (in 24x12x24 layout) to achieve a visual quality on a par with the Cartesian framework. We can even favor the output from the FCC lattice considering that it captures the gap between the pinky and ring fingers, whereas the Cartesian output does not. We also note that, thanks to the lower-resolution lattice being employed, extraction with our marching
algorithm is faster than that of Marching Cubes algorithm. Another advantage of this sparse lattice is the less number of triangles produced in the final representation. We finally pull down the resolution of the Cartesian lattice to be as close as to the FCC lattice by using 32x13x16=6656 points, and observe similar execution times and final number of triangles at the expense of degraded mesh quality; please see Table 1. Figure 10 provides a similar comparison with increased lattice resolutions to see the behavior of the execution times and the final number of triangles. We upsample the Cartesian lattice to 19467 points and the corresponding FCC lattice has 12450 points. A linear increase in execution time reveals scalability of our approach, as shown in Table 1. Note that our competitor, namely the Marching Cubes method on the Cartesian cubic lattice, still fails to capture the gap between the pinky and ring fingers despite the increased resolution. The final number of triangles by the Marching Cubes method on the Cartesian lattice is about 1.5 times larger than our result (5872 vs. 3847), which is yet another advantage of our method. As far as the execution times are concerned, we observe in Table 1 that extractions from the FCC lattice via our marching algorithm are faster than the Marching Cubes method running on the Cartesian cubic lattice. Figure 11 and the accompanying Table 2 demonstrate the extraction of a different isosurface with the same comparison settings as in the case of the hand isosurface. It is again observed that in significantly less time, FCC lattice produces an isosurface that is on a par with the Cartesian result in terms of visual quality. For the isosurfaces in Figure 11-top, FCC lattice is composed of 16x14x24 = 5376 sampling points, whereas the Cartesian lattice has a higher resolution of 24x18x17 = 7344 points. For the isosurfaces in Figure 11-bottom, FCC and Cartesian lattices have 10210 and 15422 points,
respectively. Thanks to the reduced sampling rate of our FCC lattices, the average number
of triangles needed to represent the elephant isosurface is about 1.5 times less with our
method when compared with the Marching Cubes results.

We also perform further quantitative evaluation of our extraction in comparison with the
Marching Cubes extraction based on the popular ad-hoc tool Metro [31]. To this end, we
compare the geometric Euclidean distance from our resulting surface and from Marching
cubes surface to the ground-truth surface based on the closest matching points as .29 and
.33, respectively, where the maximum distance in average is normalized to 1. The same
ordered pair of numbers is (.28, .32) for the elephant model, which again favors our
resulting extraction in both two metrics.

We finally provide comparisons for the extractions based on synthetic data, namely
analytical functions. The first function is the Marschner-Lobb (ML) isolevel function
introduced in [32]:

\[ f(x) = [(1-\sin(\pi z/2)) + \mu(1+\Omega(\sqrt{x^2 + y^2}))] / 2(1+\mu) \]

where

\[ \Omega(r) = \cos(2\pi \alpha \cos(\pi r/2)), \]

where \( x, y, z \) are coordinates of the point \( x \), \( \mu = 0.25 \), and \( \alpha = 6 \), as suggested by [32].

When executed on the 41x41x41 Cartesian and 32x32x64 FCC lattices, i.e., roughly the
same resolutions with about 67K points, Marching Cubes and our marching method
produce the images in Figure 12. Isosurface extracted from the non-Cartesian FCC lattice
by our marching method is better than that of the former method, i.e., Marching Cubes
on the Cartesian lattice. This verifies our another positive claim about FCC lattices
mentioned in the beginning of Section 4: we infer more information from the FCC lattice
than we do from the Cartesian lattice when both lattices are equipped with the same
number of points. We also note that the execution times of both methods are similar as
the lattice resolutions are similar (Table 3). We support this claim further by noting that
when the cubic lattices in Figures 9 and 11 are downsampled to 6.9K and 5.3K points,
respectively, to match the number of samples used by the corresponding FCC lattices, we
see losses in visual quality.

We should also note in Table 3 that the number of triangles in the resulting ML surfaces
favors Marching Cubes method as we are using lattices of almost equal density. In such
a case, it is natural to expect more triangles from our method as it boils down to Marching
tetrahedra which is known to produce more triangles than the Marching Cubes. Please
finally note that it is recently shown that the Marshner Lobb function is not convenient
for comparison of cubic, BCC, and FCC lattices, as it does not have a spherical frequency
spectrum [3]. We nevertheless want to provide extractions based on this function as it
demonstrates advantages, e.g., smoothness, and disadvantages, e.g., excessive triangles,
of our method.

The other synthetic analytical functions we employ are the spherical with linear falloff:
\[ f(x) = \sqrt{x^2 + y^2 + z^2}, \]
and the following simple max function:
\[ f(x) = \max(x, y, z) \]
Using spherical falloff, we have the chance to compare our resulting isosurface with the
ground-truth spherical isosurface that we obtain by starting with a crude tetrahedron
approximation and repeatedly bisecting the facets at the same time moving them to the
surface of the sphere. We observe in Figure 13 that our isosurface mesh is as smooth as
the ground-truth isosurface obtained at a subdivision level with almost the same number
of points as our mesh. We also show in Figure 13 the smoothness of the three planes resulting from the processing of the simple max function.

All of the execution times are obtained on a laptop with 2GB RAM and 2GHz processor. The system can easily be further sped up once the marching procedure that processes each tiling cell independently is spread on multiple cores. We finally note that the source code, executable, and video for the method that we present in this paper are publicly available.

6. Conclusion

We have introduced a novel isosurface extraction scheme based on the efficient Face-Centered Cubic (FCC) lattice and marching techniques. Having tiled up the volumetric space with the rhombic dodecahedra cells, we decompose each tiling cell into tetrahedra to be able to apply the Marching Tetrahedra algorithm. Tests on the implicit data based on different isovalue functions revealed the quantitative and qualitative performance as well as the time efficiency of our FCC-based results in comparison with the conventional Cartesian-based results. We point out the importance of the tetrahedra alignment, whose absence might otherwise create topological inconsistencies in the resulting isosurface manifold.

After careful decomposition of the rhombic dodecahedron cells of the FCC lattice into tetrahedra, our method boils down to the Marching Tetrahedra algorithm, which may produce excessive number of triangles while handling the topological ambiguity problems. As a future work, we consider to develop an edge collapse procedure based on a convenient error metric to reduce the triangle count. Another solution can be the incorporation of the Marching Cubes method into our framework as it produces less triangles than Marching Tetrahedra does. We should, however, be careful to design this
hybrid framework as Marching Cubes method may lead to topological problems on some configurations, especially when coupled with the Marching Tetrahedra algorithm.

References


**List of figures**

**Figure 1** Isosurfaces extracted by Cuberille (a) and Marching Cubes (b) methods.
Figure 2 Overall isosurface extraction scheme.

Figure 3 The nesting structure of the non-Cartesian FCC lattice (left), and the non-Cartesian BCC lattice (middle) for the original data given in the Cartesian lattice (right). Notice the reduced sampling densities of the non-Cartesian lattices. For the same volume, FCC and BCC lattices have half and quarter sampling density, respectively, compared to the Cartesian lattice.

Figure 4 Creation of a rhombic dodecahedron from six outer and one central cubes [30].

Figure 5 2D FCC tiling with hexagonal cells (left) are lifted up to 3D FCC tiling by using rhombic dodecahedron cells (right). Each rhombic dodecahedron in 3D tiling decomposes into 4 parallelepipeds (right-bottom).
**Figure 6** Decomposition of a parallelepiped into six tetrahedra used by our algorithm.

**Figure 7** Alignment of tetrahedra within a rhombic dodecahedron (left) and across two dodecahedra (right). Corresponding edges of adjacent tetrahedra align well (some correspondences highlighted through cyan and orange lines).

**Figure 8** (Left pair) Original images of the elephant and hand objects and some of the silhouette images from the hand (middle). (Right pair) Corresponding ground-truth digital surfaces obtained by [22] for evaluation purposes.
Figure 9 Isosurface extraction based on our method on the FCC lattice (left) and Marching Cubes method on the Cartesian cubic lattice (right). Two different views are given at each column.

Figure 10 Isosurface extraction based on our method on the high-resolution FCC lattice (left) and based on Marching Cubes method on the high-resolution Cartesian cubic lattice (right). Two different views provided at each column.
Figure 11 Isosurface extraction based on our method (a) and Marching Cubes method (c) on the low- (a,c) and high-resolution (b,d) FCC lattices and Cartesian cubic lattices.

Figure 12 ML isosurface extracted from the FCC lattice by our marching method (left) and from the Cartesian lattice by the Marching Cubes method (right). Two different views provided for each isosurface output.

Figure 13 Ground-truth sphere surface obtained by subdividing an initial tetrahedron (left pair) has the same smoothness as our resulting sphere isosurface extracted from FCC lattice (middle pair). Yet another smooth surface from our method matching the ground-truth planes (rightmost).

List of tables

<table>
<thead>
<tr>
<th>Lattice</th>
<th>Method</th>
<th># Lattice Pts</th>
<th>Time (secs)</th>
<th># Triangles</th>
<th>$Q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCC</td>
<td>Our</td>
<td>6912</td>
<td>0.09</td>
<td>2133</td>
<td>.31</td>
</tr>
<tr>
<td>Cartesian</td>
<td>MC</td>
<td>9270</td>
<td>0.21</td>
<td>3249</td>
<td>.33</td>
</tr>
<tr>
<td>Cartesian</td>
<td>MC</td>
<td>6656</td>
<td>0.08</td>
<td>2074</td>
<td>.43</td>
</tr>
<tr>
<td>FCC</td>
<td>Our</td>
<td>12450</td>
<td>0.17</td>
<td>3847</td>
<td>.26</td>
</tr>
<tr>
<td>Cartesian</td>
<td>MC</td>
<td>19467</td>
<td>0.44</td>
<td>5872</td>
<td>.27</td>
</tr>
</tbody>
</table>
Table 1 Isosurface extraction information for the hand isosurface. MC stands for Marching Cubes. This table presents how the execution times (4th column) change as we switch from the well-known Cartesian lattice (rows 2-3 and 5-6) to our FCC lattice (rows 1 and 4). Accuracy change is also reported in the 6th column. First 3 rows are for the coarse lattice and the other 3 are for the denser lattice. Same layout applies to Table 2.

<table>
<thead>
<tr>
<th>Lattice</th>
<th>Method</th>
<th># Lattice Pts</th>
<th>Time (secs)</th>
<th># Triangles</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCC</td>
<td>Our</td>
<td>5376</td>
<td>0.08</td>
<td>2596</td>
<td>.32</td>
</tr>
<tr>
<td>Cartesian</td>
<td>MC</td>
<td>7344</td>
<td>0.19</td>
<td>3899</td>
<td>.35</td>
</tr>
<tr>
<td>Cartesian</td>
<td>MC</td>
<td>5265</td>
<td>0.07</td>
<td>2570</td>
<td>.39</td>
</tr>
<tr>
<td>FCC</td>
<td>Our</td>
<td>10210</td>
<td>0.16</td>
<td>4540</td>
<td>.28</td>
</tr>
<tr>
<td>Cartesian</td>
<td>MC</td>
<td>15422</td>
<td>0.43</td>
<td>6867</td>
<td>.29</td>
</tr>
<tr>
<td>Cartesian</td>
<td>MC</td>
<td>10107</td>
<td>0.15</td>
<td>4436</td>
<td>.35</td>
</tr>
</tbody>
</table>

Table 2 Isosurface extraction info for hand isosurface. MC stands for Marching Cubes.

<table>
<thead>
<tr>
<th>Lattice</th>
<th>Method</th>
<th># Lattice Pts</th>
<th>Time (secs)</th>
<th># Triangles</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCC</td>
<td>Our</td>
<td>65536</td>
<td>0.81</td>
<td>21829</td>
</tr>
<tr>
<td>Cartesian</td>
<td>MC</td>
<td>68921</td>
<td>0.89</td>
<td>19993</td>
</tr>
</tbody>
</table>

Table 3 Isosurface extraction info for the ML isosurface. MC stands for Marching Cubes.