Marching cube algorithm: review and trilinear interpolation adaptation for image-based dosimetric models

D.A. Rajon, W.E. Bolch*
Department of Nuclear and Radiological Engineering, University of Florida, Gainesville, FL 32611-8300, USA

Received 13 May 2002; revised 12 November 2002; accepted 20 February 2003

Abstract

Current internal organ dose assessment methodologies utilize three-dimensional (3D) medical images of the body to model organ shapes and tissue interfaces. These models are coupled to computer programs that measure radionuclide energy deposition or chord-length distributions directly within these images. Previous studies have shown that the rectangular shape of image voxels generates voxel effects that alter the outcome of these calculations. To minimize voxel effects, the present study proposes to use the Marching Cube (MC) algorithm to generate isosurfaces delineating tissue interfaces from the gray-level images. First, a review of the different techniques surrounding the MC algorithm is presented. Next, an adaptation of the algorithm is proposed in which a trilinear interpolation of the gray levels is used to generate a hyperboloid surface within the MCs. This new technique is shown to solve the classic ambiguity problem of the MC algorithm and also to reduce the data size inherent to the triangulated surface. It also provides a simple algorithm to accurately measure distances within the image. The technique is then tested with a mathematical model of trabecular bone. The trilinear interpolation method is shown to remove voxel effects and to produce reliable chord-length distributions across image regions. The technique is thus recommended for use with digital medical images needed for internal radiation transport simulations. The current study is performed for a single isosurface that separates two media within the same image, but it is proposed that the technique can be extended to multiple isosurfaces that delineate several organs or organ regions within 3D tomographic voxels of human anatomy.

Keywords: Marching cubes; Isosurface generation; Trilinear interpolation; Bone marrow; Internal dosimetry

1. Introduction

1.1. Bone marrow dosimetric models

Internal dosimetry plays an important role in the evaluation of risks associated with (1) occupational exposures to radionuclides and (2) normal organ toxicity during radionuclide therapies for cancer treatment. Current internal dosimetry techniques can be traced to methods originally published in 1968 by the Society of Nuclear Medicine’s Medical Internal Radiation Dose (MIRD) Committee [1,2]. The method requires separate determinations of (1) $\bar{N}$, the integral number of nuclear decays that occur within source organs/tissues of the body, and (2) the $S$ value, defined as the absorbed dose to the target organ/tissue per decay in the source [3]. $S$ values, in turn, require assessments of the absorbed fraction, $\phi_{S-T}$, defined as the average fraction of emitted radiation energy in the source tissue that is deposited in the target tissue. Current dosimetric techniques to determine absorbed fractions make use of Monte Carlo radiation transport codes coupled with mathematical models of human organ/tissue anatomy.

Stylized mathematical models of the human body have traditionally been developed using 3D surface equations to delineate organ or tissue boundaries during radiation transport simulations [4–8]. As these stylized models have become more and more complex in their geometric description, their coupling to radiation transport codes has become increasingly challenging. Furthermore, a stylized representation of patient anatomy can lead to patient-specific uncertainties in organ dose. Stylized models have also been created to model microscopic tissue anatomy. In bone marrow dosimetry, the difficulty is to simulate the individual trabeculae of bone that define the marrow cavities. Early studies demonstrated that measurements of
chord-length distributions across both the marrow cavities and bone trabeculae could solve this problem [9]. By comparing electron or beta-particle ranges in marrow or bone with chords randomly chosen from the corresponding chord-length distribution, one can estimate the absorbed fraction at various source energies [10–12]. Other techniques permit 3D transport within a trabecular microstructure that is constructed using these chord-length distributions [13,14]. Dosimetry models that are based upon chord-length distributions are referred to as chord-based models.

Recently, computed tomography (CT) or magnetic resonance (MR) imaging have been used to provide images of human anatomy needed for radiation transport studies. The image voxel is assigned a gray-level that can be used for visualization or for tissue segmentation. These segmented images are then used as input geometries for energy deposition calculations. In these voxel-based models, the computer simulation is largely simplified since the calculation is reduced to the intersection of particle trajectories with rectangular voxel faces. This simplicity improves the calculation speed and the reliability of the dosimetry codes. The technique has already been applied to whole-body models with excellent results [15,16]. Current imaging techniques additionally allow the acquisition of 3D visualization of trabecular bone within ex vivo samples via either micro-computed tomography (microCT) [17,18] or nuclear magnetic resonance (NMR) microscopy [19–22] at resolutions below 100 μm. These high-resolution images are segmented to provide a bone–marrow interface that follows the faces of the image voxels within the radiation transport code [19,20]. This new application of voxel-based models is seen as an improvement over the chord-based models, as it better represents the real geometry of the trabecular-bone microstructure. Another application of 3D microimaging of trabecular bone is the measurement of chord-length distributions directly within the skeletal samples for use in chord-based skeletal dosimetry models [23,24].

Two different techniques thus exist to perform absorbed fraction calculations using digital images. The first is to transport the particle directly within the image voxels, and the second is to use the image to measure chord-length distributions needed for chord-based dosimetry models. Both the simplicity of the voxel shape and the accuracy of the imaging methods to reproduce the anatomic geometry make these techniques appropriate for dosimetry calculations. However, the rectangular voxel faces have geometric consequences known as voxel effects. First, when chord-length distributions are measured through a digital image, repetitive spikes appear along the distributions, and an important overestimation of the short-chord frequency is observed [24,25]. Attempts have been proposed to reduce these effects, but their study using geometrical models (e.g. spheres and distributions of spheres) proved this to be a difficult task [24,25]. Second, the voxelization process does not preserve the surface area of the boundary between two segmented tissues (even though it preserves the volume). Improving the image resolution does not reduce this effect which was analytically shown to approach a 50% even at very small voxel sizes [26,27]. This overestimate of the surface area has important consequences when the voxel-based model is used to calculate radiation energy deposition near the tissue interface [26,27]. The conclusion of these previous studies was that a new technique was needed to transform voxel-based models into surface-based models of trabecular bone [25,27].

1.2. 3D image-rendering techniques

Many algorithms have already been proposed for rendering of 3D surfaces. Cuberilles [28–31] and ray casting [32–34] were the first reported techniques published in the early 1980s. With these volume-rendering techniques, the final 2D picture results from the projection of the 3D image along the viewing direction, with each voxel being carefully assigned an opacity level that determines its relative importance during the projection process [34]. In contrast to volume rendering, surface rendering computes a representation of the interface that separates the object of interest from the background. The surface is usually a set of polygons that are projected along the viewing direction and Gouraud-shaded for better rendering. The simplest solution is to segment the image into two categories of voxels: those inside and those outside the interface surface. The voxel faces that connect two voxels belonging to a different category become the polygons (rectangles in this case) of the interface. An important theory has been developed for surface tracking within n-dimensional images [35,36]. This theory proposes algorithms that are faster than the traditional segmentation techniques, but they require knowledge of at least one surface voxel per object embedded in the image to start the tracking. While they produce surfaces that follow the orthogonal faces of the voxels, these techniques would not resolve the dosimetry-related voxel effects discussed previously.

Shell rendering is another technique developed to improve the processing time and the data storage of a digital image [37–39]. Shell rendering makes use of the surface-tracking algorithm mentioned previously where only those voxels that are close enough to the boundary to contribute highly to the rendering of the object are considered. The gray level (image intensity) of the voxels provides information about the opacity of each voxel during its projection on the pixel within the final 2D rendered image. As a result, shell rendering can be seen as volume rendering using only the vicinity of the boundary. Because the shell contains only that part of the original image that surrounds the boundary, it provides a more compact solution to manipulate the representation of the object. It also preserves fuzziness and digital aspects of the voxels, both shown to be the most important features of a digital image.
Shell rendering can also be seen as a surface representation of the object. The surface is defined by the whole set of voxels contained in the shell instead of an exact surface that separates the voxels into two categories. The voxel gray levels are used to determine the most likely surface location within the shell. An important consequence is that the data structure that stores the shell can be used to measure distances between two points of the rendered surface or between a point in space and a plan defined by three points on the surface [38].

In 1987, Lorenson and Cline proposed the Marching Cube (MC) algorithm [41] that has become a standard for isosurface generation. The principle is that the voxel gray level is used to interpolate the surface location along the segments that join the voxel centers. Next, a set of triangles that represents the object boundary is constructed from these interpolated points. The triangles are projected along the viewing direction and Gouraud-shaded for rendering. This technique is not as efficient as the volume rendering techniques for image visualization, but the triangulated surface no longer follows the orthogonal faces of the voxels and the MC algorithm is seen as a possible solution to dosimetry-related voxel effects. Several problems were discovered after the MC algorithm first appeared, and many adaptations have been proposed to solve these imperfections. Van Gelder has given a detailed description of the mathematical aspects of the algorithm, as well as many of its adaptations [42].

T-shell rendering [43] is a new extension of shell rendering that takes advantage of both the shell structure (compactness and fastness) and the MC algorithm (better surface localization). The technique allows the measurement of surface areas and volumes [44] and facilitates the measurement of distances within objects. It is noted that Nyström et al. [44] did not take into account the interpolation of the gray levels to determine the location of the triangle vertices along the edges of the MCs. Instead, their triangle vertices are located at the middle of the edges. This choice was dictated by computation optimization, but it removes image fuzziness that was seen as an important feature of a digital image [40]. The result is a surface that is not as smooth as it would be under an interpolation calculation; moreover, surface area measurements are overestimated by 8.8% [44]. This overestimation is an improvement over a measurement following the voxel faces (50% overestimate for a sphere [27]); nevertheless, taking into account the interpolation will provide a perfect convergence of the surface area as will be shown in Section 4.1.

1.3. Applications to skeletal dosimetry

As seen in Section 1.1, our aim is to model the bone–marrow interface in a manner that eliminates, or at least reduces to an acceptable level, the voxel effects when measuring distances within the image. For that purpose, we need a model that is capable of giving a precise localization of the boundary so that we can measure distances when traveling through the image. We also want the boundary to be a smooth surface that preserves the real surface area of an object. Among the solutions listed in Section 1.2, the MC algorithm seems to be of greatest interest. As we are also interested in the compactness of the image data, the shell technique may provide a solution to treat larger images. Note that the computation speed considered for image rendering is not a critical issue since we do not intend to treat the entire image as a whole. Instead, we transport particles (or straight trajectories in the case of chord length distributions) through the image and only need to analyze a local group of voxels each time a particle approaches the object boundary. Therefore, the speed of our programs will not be improved by the compactness of the image data but only by how fast we will be able to extract and process useful local information from the image. A consequence is that, for our particular problem, the only improvement of the shell-rendering techniques over the standard MC algorithm may be the compactness of the image data that would allow use of higher image resolutions.

As explained by Udupa and Odhner [37,38], the shell thickness depends on the degree of fuzziness of the shell. The best compactness is obtained with only one layer of voxels stored in the shell. Still, the compactness of the method also depends on the object size compared to the voxel size. Typically, a compression factor of 7 can be achieved with common medical images such as the representation of a skull [37]. This compression factor includes all the information needed to retrieve the location of the voxels within the image as well as the neighbor opacity code (or MC configuration in the case of the T-shell). For a skull (or any large object compared to the voxel size), only a small fraction of the initial voxels would be in the shell, since most of the voxels are either inside or outside the surface. As an example, the image used by Udupa and Odhner [37] has 1.5% of its voxels in the shell (260,000 voxels in the shell for a 256³ image). A typical trabecular bone image represents a homogeneous field of bone spicules whose thickness is on the order of five voxel sizes with a distance between two consecutive spicules of ~15 voxel sizes (using the current capabilities of NMR imaging). With such an object density it was calculated that about 25% of the voxels would be in the shell. With such a large fraction, the compactness would be ~17 times (25/1.5) less than for a skull and the shell would be larger than the image itself. As a consequence, we will not implement the T-shell algorithm, but instead we will take advantage of the simplicity of the MC algorithm.

The purpose of the present study is to propose two implementations of the MC algorithm for image-based dosimetry models. One will use the traditional MC algorithm with triangular surface generation. To overcome the large data requirement of the algorithm, a second implementation will calculate an isosurface based on
the trilinear interpolation of the image gray-level. This later approach proposes a single surface equation per cube that simplifies the exploration of the image by a computer program and does not require additional data than the initial image itself. To introduce these techniques, we will first give a simplistic, but detailed, review of the MC algorithm. Next, our approaches will be described and tested using a mathematical model of a trabecular bone sample. Finally, we will discuss the adaptation of the techniques to anthropomorphic models that contain several isosurfaces.

2. The Marching Cube algorithm

The MC algorithm is commonly divided into two steps: (1) the construction of the isosurface corresponding to an isovalue of gray-levels and (2) the calculation of the gray-level gradient vectors along the isosurface. The first step produces a set of triangles that forms the isosurface. The gradient vectors of the second step are used to produce Gouraud-shaded images that better display a 2D projections of the 3D object. In order to use the isosurface for radiation transport simulations, only the first step is necessary, and thus we will limit the present study to the construction of the isosurface. Of course, image rendering will be used for visualization purposes in the remainder of this text, but this task will be relegated to IDL 5.5 [45].

2.1. Original MC algorithm

The study of a simple 2D case will serve to introduce the algorithm. Fig. 1a is a small image in which each pixel is assigned a gray-level from 0 to 3. For this gray-level range, the isocontour will be defined by the isovalue 1.5. In Fig. 1b, a grid, that delineates the marching squares, has been superimposed on the image. Each square has its vertices located at the center of four adjacent pixels of the initial image with each vertex assigned a black dot if the pixel gray-level value exceeds the isovalue. An edge joining two adjacent square vertices will be crossed by the isocontour if one of its two vertices, and only one, has a dot. In Fig. 1c, all such square edges are tagged with a small circle located at the center of the edge. Next, the circles are joined together by a set of segments that forms the isocontour. Finally,
the location of each small circle is interpolated along the edge, using the gray-levels of the two edge vertices. The final isocontour is represented in Fig. 1d.

The algorithm proposed by Lorensen treats one square at a time. The four vertices of the square are used to decide whether or not the isocontour crosses the square, and to calculate the intersection points. Once a square is treated, the algorithm marches to the adjacent square until the entire image is covered. With four vertices that can be above or below the isovalue, 16 different square configurations are possible. The segmentation of these configurations is shown in Fig. 2a. The configuration numbers (from 0 to 15) represent the binary state of the four vertices. One can notice that configurations 5 and 10 are crossed twice by the isocontour. A computer program can easily calculate the configuration numbers for each marching square and a look-up table can be used to determine which edges will be joined to form the isocontour within the marching square. Next, the interpolation is performed along the edges to more precisely locate the intersection points. Of course, with the same edge being involved in two marching squares, the program can be optimized.

In 3D, the MCs are made of eight vertices located at the centers of eight adjacent voxels. Therefore, there are 256 possible cube configurations. Each configuration can be intersected up to four times by the isosurface. Each intersection is a polygon, whose vertices are interpolated along the 12 edges of the cube. The edges of a polygon are segments that lie within the faces of the cube in the same way that the segments of the 2D isocontour lie within the marching squares. Therefore, Fig. 2a also gives the 16 different possibilities a MC face will be intersected by the isosurface. Since each intersection point is interpolated independently, all polygons joining more than three intersection points are mostly non-planar. As a consequence, the polygons need to be divided into triangles to be handled in equation form by the computer program utilizing the image (e.g. for radiation transport simulations or chord-length measurements).

As mentioned by Lorensen and Cline [41], the triangulation of the 256 possible configurations is a tedious and error-prone task. Fortunately, a large number of symmetries can be found within a cube. In the 2D situation of Fig. 2a, configurations 1, 2, 4, and 8 can be derived from one another by basic rotations and then grouped into a single pattern numbered 1 as shown in Fig. 2b. Furthermore, configurations having three or four dots are the binary complements of configurations with one or zero dots, respectively, and thus there is no need to create new patterns for these configurations. Two complementary configurations will have the same segmentation. Therefore, the study is reduced to the four patterns of Fig. 2b. Note that patterns 2 and 3 do not have complement patterns since two complementary configurations are symmetrical and belong to the same pattern.

In 3D, Lorenzen studied rotations and complementation in the cube, and proposed 15 patterns from which one can derive all 256 configurations [41]. These patterns are presented in the four first rows of Fig. 3. Note that pattern 11 and 14 are symmetric by reflection, but Lorenzen only considered rotations [41,42] and decided to treat them independently. The two last rows in Fig. 3 represent the complement patterns of patterns 0–7. They are not considered in Lorenzen’s algorithm since they are treated as patterns 0–7. Note that patterns 8–14 all have four dots and include configurations that are complement by pairs; consequently, they do not have a complement pattern. The entire study of the triangulation of the isosurface is hence reduced to the study of the 15 patterns.

Overall, the MC algorithm is easy to implement. Marching over the entire image, each cube is treated one at a time using only eight voxels from which a configuration number is calculated. This number is used as an index to a look-up table that contains the number of triangles and the edges of the MC on which each triangle is connected. Finally, the gray-levels of the eight voxels are used to
interpolate each intersection point along its edge. The construction of the 256-configuration look-up table is facilitated by the rotations and the complementations found within the 15 patterns. Another solution would be to create a look-up table for the 15 patterns only, and let the computer program do the rotations and complementations within the cube. This technique, however, does not simplify the algorithm since rotations and complementations have to be treated anyway, and the program still needs a configuration look-up table to determine the pattern of each configuration. In addition, the computer code that performs these operations would be more complex, error-prone, and slower than codes that rely only on look-up table values.

As one can see in Fig. 3, the same intersection point belongs to several triangles of the same MC. In addition, the same cube edge is shared between four adjacent cubes. As a result, the interpolation process can be optimized and the MC triangulation is often divided into two steps: (1) creation of a vertex table by calculating the three coordinates of each intersection point after interpolation, and (2) creation of a triangle table that contains three pointers to three vertices of the first table.

### 2.2. Ambiguity problem

In 1988, Düurst [46] first reported that the MC algorithm might produce holes in the isosurface. These holes have their origin in what is referred to as ‘the ambiguity problem’. Let us consider pattern 3 of the 2D situation (Fig. 2b). Both configurations 5 and 10 belong to this pattern and divide the square into dotted regions that isolate a single non-dotted region. For both configurations, the surface-area segmentation within the square is 25% above the iso-value and 75% below, whereas the symmetry of pattern 3 would instead suggest a 50–50% repartition. Furthermore, why should pattern 3 always propose a segmentation that separates two dotted regions by one non-dotted region? This choice tends to unfairly reduce the size of the dotted regions. It should be possible to also propose a reverse geometry as shown in Fig. 4a.

The ambiguity for the choice between the direct and the reverse segmentation of a square can lead to a totally different result for images with small features. This problem was addressed by Zhou [47] and referred as the type A small feature problem. Fig. 4b, illustrates the worst-case scenario: the image is alternately made of dotted and non-dotted regions.

![Fig. 3. The MC patterns proposed by Lorensen for 3D images. Patterns 0e–7c are the complement of patterns 0–7 and are designed alike. Since patterns 8–14 have four vertices above the iso-value and four vertices below, they include their own complements.](image)

![Fig. 4. (a) The two possible designs for the ambiguous square. (b) Depending on the design chosen, the same gray-level image would have a completely different segmentation. Direct: all ambiguous squares have a direct design. Reverse: all ambiguous squares have a reverse design. Alternate: the squares are alternatively designed with direct and reverse segmentation. Random: the decision is made randomly for each square.](image)
voxels. The direct segmentation isolates small dotted regions within an infinite non-dotted field, whereas the reverse segmentation isolates small non-dotted regions within an infinite dotted field. The symmetry of the initial image suggests a symmetric segmented image with identical shapes for the dotted and non-dotted regions. In 2D, either alternatively or randomly choosing between the direct or the reverse segmentation for each square can solve the problem. The results are shown in Fig. 4b. The alternate segmentation gives symmetrical stripes, preserves the surface-area, and proposes symmetric shapes. However, the result shows an anisotropy not suggested by the initial image. Consequently, the random segmentation would be preferred.

The example in Fig. 4b is an extreme case with perfect symmetry between the dotted and the non-dotted regions. In most images, this symmetry does not exist in the real structure, and both the alternate and the random segmentations are not suitable. The decision between direct and reverse segmentation should be done independently for each square using the four vertices of the square. Such a solution has been proposed by Nielson [48] and will be discussed later.

2.3. Hole problem

In 3D, the ambiguity problem appears each time a face of a cube is ambiguous. In Fig. 3, one can see that six of the regular patterns and three of the complement patterns have ambiguous faces. In 2D, both the direct and the reverse segmentation can coexist within the same image because the choice between direct and reverse only changes the segments between the intersection points, but not their location along the edges. In 3D, an ambiguous face is shared by one cube that may use the direct segmentation and one cube that may use the reverse one. This situation would lead to intersecting segments that do not match properly. Therefore, in 3D an alternate or a random segmentation such as the one proposed in Fig. 4b cannot produce a continuous isosurface.

Lorensen proposed to treat the complement configurations with the same pattern design [41]. For example, in Fig. 3, patterns 3 and 3c are designed with the same triangles. However, the bottom face of pattern 3 is of the direct type whereas the bottom face of pattern 3c is of the reverse type. When a pattern 3c is flipped over and put underneath a pattern 3, the result is the discontinuity shown on the left panel of Fig. 5a. The problem appears each time a complement pattern is connected to a regular pattern by an ambiguous face. Fig. 5a also shows a pattern 10 on top of a pattern 6c (center) and a pattern 6 on top of a pattern 3c (right). The result of this unsolved ambiguity problem is a hole in the isosurface such as shown in Fig. 6a. Fig. 6a is a 22 × 22 NMR microscopy image of a small segment of the trabecular bone microstructure. The left image is the direct display of the triangles produced by the MC algorithm using the patterns of Fig. 3. On the right picture, the rendered image is shown as generated by IDL 5.5 using the triangle list.

2.4. Image size and processing time

Another significant concern that the early users of the MC algorithm had to face was the amount of data to be generated [49,50]. Each intersection point needs three coordinates. With 4 bytes to store a floating point number on most computer systems, this represents 12 bytes per point. For each triangle, three pointers are required to access the intersection points. Again, 12 bytes are required to store the three pointers. The image shown in Fig. 6a has 2110 intersection points (or triangle vertices) and 4214 triangles. That is a total of 75,888 (12 × 2110 + 12 × 4214) bytes required to store the data (both the triangles and the intersection points). The initial image with one byte per voxel occupies only 10,648 (22 × 22 × 22) bytes. The MC algorithm has in effect multiplied the size of the data by a factor of seven. Of course, this multiplication factor would
be reduced with a smaller voxel size since a smaller fraction of the cubes would be on the isosurface. However, the resolution used for Fig. 6 (between 4 and 10 times smaller than the smallest features of the object size) is a typical ratio to obtain a good definition of the object. Furthermore, the multiplication by seven has been observed on most trabecular bone images tested. Therefore, an ordinary 256 $\times$ 256 $\times$ 256 image would create about 3.3 million summits and 6.6 million triangles, which represents more than 110 MB.

The construction of the triangulated isosurface shown in Fig. 6 takes about 0.1 s on a 750 MHz Unix machine. The look-up table indexed by the configuration numbers provides a fast access to the triangle designs and allows generating about 40,000 triangles per second. The isosurface of a 256 $\times$ 256 $\times$ 256 image can be treated in < 3 min. Once created, the isosurface needs to be processed and this is a greater concern. Being able to do fast mathematical transformations of an image containing millions of triangles is a difficult challenge. Furthermore, the MC algorithm initially proposed does not optimize the number of triangles generated. One can see in Fig. 6 that many triangles are generated for the flat surface that represents the right edges of the sample (about 250 triangles). This surface could easily be modeled with fewer triangles. Treating the image one cube at a time has reduced the complexity of the algorithm but leads to the impossibility of adapting the size of the triangles to the local geometry.

### 2.5. Marching tetrahedrons

The simplest solution to solve the hole problem is to eliminate ambiguous faces. A tetrahedron has four vertices and only 16 possible configurations. To segment the triangular faces, a single connection joining two edges is required. This extreme simplicity leads to only three patterns and two complement patterns. They are represented in Fig. 7a. There is no ambiguous face and only
Many propositions have been made to break up a MC into a set of tetrahedrons \([51–59]\). Fig. 7b proposes two solutions to fit five tetrahedrons into a cube. One can see in Fig. 7b that two opposite faces of a cube have a different break up. Therefore, both breaking-up patterns proposed in Fig. 7b must be used alternatively to avoid any discontinuity within the isosurface. Furthermore, a cube vertex can be owned by either one or four tetrahedrons. Therefore, alternating the two segmentation methods within the same image will force a vertex to be owned either by 8 or 32 tetrahedrons. The consequence is that an isolated dot would be surrounded by one of the two surfaces shown in Fig. 7c and the isosurface would not respect the homogeneity of the initial image.

In Fig. 7d, the cube is broken up into six tetrahedrons. Since two opposite faces of the cube have the same design, the same break-up pattern can be applied to the entire image. No discontinuity would be found within the isosurface, but an isolated dot would be surrounded by an isosurface like the one shown in Fig. 7e and the isosurface would not respect the isotropy of the initial image.

Other solutions have been proposed to break up a MC into tetrahedrons so that the resulting isosurface respects the homogeneity and the isotropy of the initial image. They involve 24 or even 48 tetrahedrons per cube. These solutions tend to produce smoother isosurfaces than the initial MC algorithm and they solve the ambiguity problem. Nevertheless, they generate a large number of triangles. Therefore, one may prefer the extended MC algorithm described in Section 2.6.

2.6. Extended MC algorithm

The hole problem is a consequence of the reverse design implicitly proposed to the ambiguous faces of the complement patterns 3c, 6c, or 7c, when they are treated as patterns 3, 6 or 7. To solve the problem, one should consider these complement patterns as independent patterns having there own segmentation that would give the ambiguous faces a direct design. This has been proposed by different research groups \([29,30,47,60–62]\) and is shown in Fig. 8. All ambiguous faces of the 23 patterns have a direct segmentation and no hole will be created. Fig. 5b shows how these new patterns connect for the three examples proposed in Fig. 5a. Fig. 6b also shows how the hole of Fig. 6a is resolved.

The hole problem is solved, but the technique will systematically overestimate the volume of the objects below the isovalue, as explained in Section 2.2. For that reason, one may think of a series of patterns that design all ambiguous faces with the reverse segmentation. This new series of patterns is proposed in Fig. 9. All faces of the 23 patterns have a reverse segmentation and the hole problem is also solved. Figs. 5c and 6c show how the holes are resolved with the reverse segmentation.

Lately, a new disambiguation method has been proposed by Delibasis \([63]\). Instead of a look-up table, the algorithm evaluates the isosurface design for each cube. Visiting the cube edges that are crossed by the isosurface (they have one dotted and one non-dotted vertices) allows following the contour of the polygons. The technique reproduces all 23 patterns of the direct segmentation of Fig. 8. The authors did not address the reverse segmentation, but a slight modification of their algorithm would generate the reverse patterns of Fig. 9. Delibasis’ method eliminates the need for a look-up table, but necessitates a complex algorithm to follow the polygon edges within the cube. In addition, the resulting program will be slower than a direct access to a configuration look-up table. Furthermore, the method produces non-planar and non-triangulated polygons and only software that can subdivide the polygons into triangles will be able to manipulate the image.
2.7. Facial deciders

Solving the hole problem by the Extended MC algorithm does not solve the ambiguity problem. Why should we use the direct patterns of Fig. 8 instead of the reverse patterns of Fig. 9? In Fig. 6, one can see that the direct solution creates a slight hollow instead of the hole, whereas the reverse solution creates a slight protrusion. If the same choice is made for the entire image, the result can be deleterious in the case of small objects. Furthermore, the two series of patterns proposed in Figs. 8 and 9 cannot be merged within the same image. For better image representation, the decision between direct and reverse segmentation should be made for each face and not at the MC level.

The average of the four gray-levels of an ambiguous face can be used to decide whether the entire face is more above or below the isovalue \[64\]. If the average is below the isovalue, the direct segmentation should be used for the face. If it is above, the reverse segmentation should be used. Nielson proposed another facial decider that is based on a bilinear interpolation of the gray-level over the ambiguous face \[48\]. The interpolation gives a function \( B(x, y) \) that represents the gray-level field over the face. The gray level is equal to the isovalue along a curve that is a hyperbola with two branches that isolate either the two dotted vertices or the non-dotted vertices. Nielson showed that comparing the function \( B(x, y) \) at the intersection of the hyperbola asymptotes with the isovalue will tell which edges of the face are connected by the hyperbola, thus allowing one to decide between direct or reverse segmentation.

The facial deciders may propose a different segmentation for two ambiguous faces within the same MC, but none of the patterns shown in Figs. 3, 8 or 9 proposes such a situation and new designs need to be created. A configuration with two ambiguous faces would have four different designs. Pattern 13, with six ambiguous faces, has 64 \( (2^6) \) possible designs. Nielson and Hamann showed that some designs could not be resolved by using only the twelve edges of the MC \[48\]. Instead, they proposed to add a vertex that is interior to the MC and whose location depends on the eight vertices of the cube. Fig. 10 shows the four triangulations proposed by Nielson and Hamann using this extra vertex for pattern 10.

The ambiguity problem is solved with both facial deciders, but they require extra configurations, and for most of them more triangles than with the extended MC algorithm. Once more, the problem of the MC algorithm is to stay at the cube level and the size of the triangles only depends on the voxel size whereas they should depend on the size of the local geometry.

Fig. 9. The 23 MC patterns proposed to solve the hole problem using the reverse design on every ambiguous face. Patterns 3, 6, and 7 have the same design as patterns 3c, 6c, and 7c of Fig. 8. Patterns 7c, 6c, and 3c have the same design as patterns 7, 6, and 3 of Fig. 8. Patterns 10, 12, and 13 also have a different design since they also have ambiguous faces.

Fig. 10. The four triangulations proposed by Nielson and Hamann for a pattern 10 to solve the ambiguity problem \[48\]. An ambiguous face can have a direct design whereas the other one has a reverse design.
2.8. Optimizing the image size

The MC algorithm produces triangles that can be very small. In Fig. 6, some triangles are barely visible and could be treated as a single pixel of the resulting image. This idea was developed by the Dividing-Cube technique [65,66]. As a triangle is far more costly to project on a computer screen than a single point, each cube of the original MC algorithm can be divided into sub-cubes until the projection of the cubes are represented by single pixels of the image.

Another consequence of the original MC algorithm is that it does not adapt the triangle size to the object size. To overcome this constraint, Montani et al. proposed the Discrete MC algorithm [49,67,68]. The idea is to merge adjacent cubes that have coplanar polygons (before interpolation) into a unique cell to create larger polygons. Then the polygon vertices are interpolated along the edges of the new cell. A large flat surface as the one seen on the sides of the bone sample of Fig. 6 would be merged into fewer triangles. This technique was proved to reduce the number of triangles by a factor 3 on common images [68].

The marching-triangle algorithm [50,69] is another method that generates a Delaunay triangulation by marching over the surface. It creates a mesh whose density does not depend on the voxel size. The resulting image also reduces the number of triangles by a factor 3 for common images.

These techniques that reduce the amount of triangles are important for rendering purposes because each surface element is treated once and its location is well known at the time it is used. For a computer program that seeks to travel through the image (e.g. as in tracing the path of a radiation particle) the problem is different. The main objective is to calculate the distance from the current location to the next isosurface intersection when traveling in a known direction. For this, the program needs to find what is the first surface element met in this direction. It would be time consuming to explore the entire triangle list of the image to check which triangle comes first. Instead, the initial MC that contains the current location provides the isolation of a few triangles. By traveling one cube at a time, the exploration is limited to the triangles that belong to the cubes that are crossed by the trajectory. As the local cube can be easily calculated from the current location, the traveling process is simplified and the exploration stops when a triangle is intersected. The distance can be computed and the process is repeated from the other side of the surface to the next intersection, eventually with a different direction depending on the purpose of the program. In the case of isosurface generation techniques that provide a list of triangles that are no longer related to the initial cube grid, this localization information is lost and the program would become more complex and slower for radiation transport or chord-length determinations. Therefore, our adaptation will focus on techniques that preserve the initial MC grid.

3. Materials and methods

As mentioned previously, our goal is to minimize the effects due to the rectangular shape of the voxels within 3D images of trabecular bone samples as applied to skeletal dosimetry studies. The first objective is to preserve the surface area of the boundary between the bone trabeculae and the marrow cavities. The second is to eliminate the voxel effects when measuring chord-length distributions through these images. We want to apply a MC-like technique to create an improved interface between bone and marrow. To test our methods, a mathematical model of trabecular bone sample was created and will be discussed first.

3.1. Mathematical bone sample

The mathematical bone sample used in the present study is based on one initially created for dosimetry calculation studies [26]. This original model had two major drawbacks. First, the real area of the interface between bone and marrow could only by estimated through rough approximation. As we want to be able to assess how the MC techniques preserve the surface area, we need to know the exact value of this area through the model. Second, some of the simulated bone trabeculae were so thin that even at high image resolution, connections were present between the spherical marrow cavities. Real samples of trabecular bone do not have extremely thin trabeculae (in comparison), and this artifact would make the surface area of the model more difficult to converge to its exact value.

To avoid these problems, we created another model made of a 1.2 $\times$ 1.2 $\times$ 1.2 cm$^3$ cube of osseous tissue filled with 3400 spheres containing bone marrow (i.e. simulating the marrow cavities). The spheres have a normalized radius distribution given by

$$P(r) = P_m e^{-Pr}.$$  

In Eq. (1), $r$ is the radius of the spheres and $P_m = 44$ cm$^{-1}$ is a parameter chosen so that the chord-length distribution within the entire sample is close to a typical marrow cavity chord-length distribution measured for the cervical vertebrae [70]. The locations of the spheres are randomly chosen within the cube but they must satisfy the three following conditions: (1) no sphere can extend beyond the limits of the cube, (2) no sphere can overlap another sphere of the model, and (3) a sphere must be at a minimum distance of 5% its own radius from any other sphere and from the cube edges. Conditions (1) and (2) allow the exact calculation of the surface area of the interface and condition (3) preserves a minimum thickness between the spheres to avoid any artificial connections of marrow cavities. A 2D cross section of the model is presented in Fig. 11. The exact surface area of the spheres has been calculated and is

$$S_{spheres} = 44.0916655 \text{ cm}^2.$$  

The normalized chord-length distribution within the sample was derived in a previous study [25]. With $l$ representing a chord length across one of the spheres, this distribution is

$$f(l) = \frac{1}{P_m^2} e^{-(P_m l^2)}.$$  \hspace{1cm} (3)

The mean of this distribution has also been calculated and is

$$\bar{l} = \frac{4}{P_m} = 909 \mu m.$$  \hspace{1cm} (4)

Eqs. (2)–(4) will be used as a benchmark for the present study.

An imaging technique was then simulated to create artificial 3D images of the mathematical bone sample. For different resolutions, ranging from 1000 to 24 $\mu m$, a grid structure is placed over the sample to delimit the image voxels. A gray-level (from 0 to 255), that represents the volume fraction of the voxel that is inside the spheres, is then assigned to each voxel. For voxels partially overlapping one or several spheres, a Monte Carlo technique employing 20,000 points per voxel is used to assess the volume fraction. The characteristics of the images are presented in Table 1. Column 1 is the voxel size and column 2 is the image size. With a gray-level based on the volume fraction and ranging from 0 to 255, a natural isovalue to segment the image into bone and marrow regions is shown to be 127.5.

In order to quantify the improvement of the MC-based models over the voxel-based models (using the initial gray-level image), the surface area of the interface was measured within the segmented images. The measurement is simple: for each voxel, each time one of the six adjacent voxels is not on the same side of the isovalue, the area of the common face is added to the isosurface area. The final result is thus divided by two, as each voxel face is counted twice. The cavity chord-length distributions were also measured within the voxel-based images for comparison with the MC-based techniques. The Monte-Carlo technique used is the same as described in a previous study [25,71]. The bin-width to build all histograms of the present study is 6 $\mu m$ and 2 million rays were fired around the sample to perform the measurements.

### 3.2. Direct and reverse extended MC algorithm

For rendering purposes, the hole problem of the initial MC algorithm can be accepted to a certain extent. With a high-resolution image, the holes would be very small and would not alter the general perception of the object for a human viewer. For a computer program trying to find its way through the image, the crossing of a hole would make it unable to remember on what side of the isosurface it then travels. Therefore, holes are not acceptable for applications to radiation transport studies within the image. On the other hand, the reduction of the number of triangles is not a priority. We prefer to keep the cubical grid generated by the MC algorithm so that the program can explore the cubes one at a time and limit its search of the intersection with the isosurface to the few triangles that belong to the local MC. As a consequence, the extended MC algorithm described in Section 2.6 is what is

### Table 1

<table>
<thead>
<tr>
<th>Voxel size ($\mu m$)</th>
<th>Voxel per dimension</th>
<th>Surface-area voxel (cm$^2$)</th>
<th>Surface-area DIR-MC (cm$^2$)</th>
<th>Surface-area REV-MC (cm$^2$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000.0</td>
<td>12</td>
<td>19.58</td>
<td>11.47</td>
<td>8.02</td>
</tr>
<tr>
<td>750.0</td>
<td>16</td>
<td>29.18</td>
<td>20.83</td>
<td>16.56</td>
</tr>
<tr>
<td>600.0</td>
<td>20</td>
<td>34.67</td>
<td>25.70</td>
<td>23.49</td>
</tr>
<tr>
<td>480.0</td>
<td>25</td>
<td>40.53</td>
<td>30.26</td>
<td>29.87</td>
</tr>
<tr>
<td>375.0</td>
<td>32</td>
<td>45.99</td>
<td>34.20</td>
<td>34.98</td>
</tr>
<tr>
<td>300.0</td>
<td>40</td>
<td>50.66</td>
<td>36.74</td>
<td>38.44</td>
</tr>
<tr>
<td>240.0</td>
<td>50</td>
<td>54.86</td>
<td>38.49</td>
<td>40.47</td>
</tr>
<tr>
<td>187.5</td>
<td>64</td>
<td>58.63</td>
<td>39.94</td>
<td>41.75</td>
</tr>
<tr>
<td>150.0</td>
<td>80</td>
<td>61.15</td>
<td>41.02</td>
<td>42.49</td>
</tr>
<tr>
<td>120.0</td>
<td>100</td>
<td>62.97</td>
<td>41.93</td>
<td>43.08</td>
</tr>
<tr>
<td>96.0</td>
<td>125</td>
<td>64.20</td>
<td>42.61</td>
<td>43.41</td>
</tr>
<tr>
<td>75.0</td>
<td>160</td>
<td>65.00</td>
<td>43.14</td>
<td>43.63</td>
</tr>
<tr>
<td>60.0</td>
<td>200</td>
<td>65.44</td>
<td>43.47</td>
<td>43.74</td>
</tr>
<tr>
<td>48.0</td>
<td>250</td>
<td>65.73</td>
<td>43.69</td>
<td>43.81</td>
</tr>
<tr>
<td>37.5</td>
<td>320</td>
<td>65.90</td>
<td>43.85</td>
<td>43.89</td>
</tr>
<tr>
<td>30.0</td>
<td>400</td>
<td>66.00</td>
<td>43.94</td>
<td>43.95</td>
</tr>
<tr>
<td>24.0</td>
<td>500</td>
<td>66.04</td>
<td>44.00</td>
<td>44.01</td>
</tr>
</tbody>
</table>

Note: The isosurface area is calculated within the voxel-based image series (column 3) and within the two MC-based image series (columns 4 and 5, respectively). The real interface area is $S_{spheres} = 44.09$ cm$^2$.
needed for our applications. However, we want to try both the direct and the reverse segmentations and see if they produce a significant difference. The facial deciders would create more triangles within each cube, would slow down the program, and may not provide a significant improvement in the final results.

For image rendering, the MC algorithm generates two lists: a vertex (or intersection-point) list with three coordinates per vertex and a triangle list with three pointers per triangle. The pointers allow a direct access to the vertex list. A synopsis of these two tables is presented in Fig. 12 along with two other tables that are described now. A program that wants to explore the image one MC at a time also needs to remember which triangles belong to which MC. Therefore, a list of the isocubes (cubes on the isosurface) is also required. For each isocube, a pointer (‘1st Tr’ in Fig. 12) to the first triangle of the cube as well as the number of triangles within the cube (‘Tr’ in Fig. 12) are needed. Each cube also needs an identification number (‘No’ in Fig. 12) that is calculated from its location (I, J, K) within the image. As the isocube list is sorted by identification number, a binary search will provide a fast access to the list. For the non-isocubes, one only needs to remember if they are inside or outside the isosurface. Since an image with a high resolution would have long series of inside or outside MCs, a look-up table with one entry per series is sufficient. Each entry gives access to the identification number of the first cube of the series (‘No’ in Fig. 12), the number of cubes in the series (‘Nb’ in Fig. 12), and the status of the series: inside or outside the isosurface (‘St’ in Fig. 12). A binary search also provides a fast access to the table. A program traveling one cube at a time within the image would need to calculate the cube identification number and access the isocube list to check if the cube is on the isosurface. If it is not, it can access the non-isocube list. If it is, the isocube provides access to the triangle list and then to the vertex list.

The direct and the reverse techniques are referred to as DIR-MC-based models and REV-MC-based model, respectively, in the remainder of this text. They were applied to the series of images of the mathematical bone sample. The surface area of the isosurface was calculated by summing the areas of each triangle of the isosurface. The cavity chord-length distributions were also measured for comparison with Eqs. (3) and (4).

### 3.3. Trilinear interpolation isosurface

The MC algorithm proposed above creates four tables (Fig. 12) that describe the isosurface within the image. These tables required a data set ~10 times larger than the image itself for a typical trabecular bone image ( ~7 times for the triangles only (Section 2.4) and ~3 times more if one includes the two cube tables). Furthermore, a Monte Carlo program that travels cube by cube within the image needs to calculate the intersection of a trajectory with the triangles of the next intersected MC. That is not a complex geometrical problem, but it can be time-consuming since there is up to five triangles per cube and the program needs to calculate the intersection with the plane defined by each triangle, as well as to check if the intersection is within the limit of the triangle.

A better solution would be to use a unique equation of the isosurface derived from the initial image and to calculate the intersection of the trajectory with this

![Fig. 12. Synopsis of the database created by the MC algorithm. The non-isocube table allows an access to the status (above or below the isovalue) of the cubes that are not on the isosurface (configurations 0 and 255). The Isocube table contains the cubes that are on the isosurface. First Tr is a pointer to the first triangle of the cube in the triangle list. The triangle table contains the pointers to the vertices of the triangle. The vertex table gives the location of the vertices.](image-url)
surface. Previous research has suggested the idea of using a trilinear interpolation of the eight gray-levels of the cube vertices to create a scalar field of gray-level throughout the cube \[42,72,73\]. The bilinear interpolation used by Nielson and Hamann in their asymptotic decider \[48\] is a restriction of the trilinear interpolation to each ambiguous face of the cube. When equaling the resulting 2D gray-level field to the isovalue, Nielson and Hamann found the equation of a hyperboloid. By extending the technique to the entire MC, the result is the equation of a hyperboloid. This method only requires memory for the initial gray-level image, eliminates the look-up tables used to generate the MC triangles, avoids the time-consuming manipulation of the triangles, and proposes a smoother representation of the isosurface than that given by collections of flat triangles.

For the MC of unit size represented in Fig. 13 and with the coordinate system origin located at vertex \(B_0\), the trilinear interpolation of the gray-level is

\[
B(x, y, z) = B_0(1-x)(1-y)(1-z) + B_1x(1-y)(1-z) + B_2(1-x)y(1-z) + B_3(1-x)(1-y)z + B_4x(1-y)z + B_5xy + B_6(1-x)yz.
\]

(5)

In Eq. (5), \(B(x, y, z)\) is the gray-level at any point within the cube and the \(B_n\) values are the gray-levels at the eight vertices. If ‘Iso’ is the isovalue that will segment the image, the isosurface is represented by

\[
B(x, y, z) = \text{Iso}.
\]

(6)

This hyperboloid has asymptotic surfaces that separate the infinite space into eight regions. Four of these regions contain the four lobes of the hyperboloid. Depending on the location of the MC relative to the surface, 1, 2, 3, or 4 of these lobes can intersect the cube. Some examples are represented in Fig. 14 for different cube patterns. In Fig. 14, the numbers in parentheses are the configuration numbers calculated from the vertex numbering proposed in Fig. 13. Fig. 14a represents a pattern 1. Note that the linear interpolation of the connecting points along the edges of the cube is a restriction of the trilinear interpolation. Therefore, the connecting points of the triangle proposed by Lorensen are the same as the connecting points of the surface shown in Fig. 14a. Fig. 14b represents another example of the same configuration with different values for the vertex gray-levels. \(B_2\) is chosen farther from the isovalue than the other vertices and the surface is pushed away from the vertex. Fig. 14c represents a pattern 8. It can be very different from a flat polygon. Fig. 14d–f represent, respectively, patterns 9, 5c, and 11. They can be compared with their equivalent patterns in Fig. 8.

It can be shown that the trilinear interpolation technique generates the 23 patterns of the MC algorithm. But how does the trilinear interpolation solve the ambiguity problem? Fig. 15a represents a pattern 3 designed with two triangles. When the gray-levels \(B_0\) and \(B_1\) are changed to a value farther from the isovalue than \(B_1\) and \(B_t\), the result is shown on Fig. 15b: the two surfaces converge closer and closer together. In Fig. 15c, they join into a single surface. The direct design of the ambiguous face of Fig. 15b has become a reverse design in Fig. 15c. Going a little farther, Fig. 15d is equivalent to the reverse design of pattern 3 in Fig. 9. Figs. 15e and f show the same change for a pattern 6. It is thus shown that the trilinear technique automatically handles the ambiguity problem.

How the trilinear interpolation allows a facial decision is shown in Fig. 16 that represents the four possible designs of a pattern 10. By changing the values of the vertex gray-levels, one can obtain the four different situations. The curves that represent the intersection of the surface with the ambiguous faces are the hyperbolae used by Nielson and Hamann to decide between direct and reverse cube-face design \[48\]. Fig. 16 can be compared with Fig. 10. It can be shown that all additional configurations introduced by Nielson and Hamann are automatically handled by the trilinear interpolation technique.

Finally, one may worry about how these surfaces will connect between the cubes. Since the bilinear interpolation at a face is nothing less than the restriction of the trilinear interpolation to the face, two adjacent cubes will converge to the same hyperbola at the connecting face. Therefore, the surface is continuous. Fig. 17a shows how a pattern 3 is connected to a pattern 3c. The surface is continuous and smooth. The smoothness of the surface is not a guaranty since the two cubes have only four vertices in common. Fig. 17b shows a slight angle

---

**Fig. 13.** Localization of the eight vertices of a marching cube. The \(B_n\) values are the gray-levels of the vertices. The index \(n\) represents the vertex number as used to calculate the cube configuration from the binary state of each vertex.
Fig. 14. Trilinear interpolated isosurfaces within unity cubes for different patterns proposed by the MC algorithm. The numbers in parentheses are the configuration numbers. The gray-levels used to obtain these surfaces are given in Table 2.
Fig. 15. Trilinear interpolated isosurfaces within unity cubes. (a)–(d) show how a pattern 3 evolves from a direct design ((a), then (b)) to a reverse design ((c), then (d)). (e) Direct pattern 6. (f) Reverse pattern 6. The numbers in parentheses are the configuration numbers. The gray-levels used to obtain these surfaces are given in Table 2.
between the two surfaces of a pattern 2 on top of a pattern 5c. In Fig. 17c, four cubes are represented together. The surface is not completely smooth, but it is continuous. For all the configurations presented in Figs. 14–17, the vertex gray-levels used to produce the isosurfaces are reported in Table 2.

The trilinear interpolation suffers a few singularities. First, if one vertex of the cube were equal to the isovalue, the surface would resume to a single point at this vertex. This is avoided by the use of a non-integer isovalue, like 127.5 for the mathematical bone sample. Second, symmetric configurations of the vertex gray-levels would lead to a surface that would no longer be a hyperboloid. As an example, one can imagine a pattern 13 with all its dotted vertices equal to 255 and all its non-dotted vertices equal to 0. This perfect symmetry would transform the surface into three planes that intersect at the center of the cube and divide it into eight sub-cubes: four of them being above the isovalue and four being below. These non-hyperboloid situations are not critical for a computer program as will be seen in Section 3.4.

Due to the complexity of Eq. (6), we will not try to evaluate the surface area of the isosurface within each MC. Our study will be limited to the chord-length distribution measurement to show that the trilinear technique is suitable for image-based computer calculations.
3.4. Intersection of a straight line with a hyperboloid

In both the particle transport code and the chord-length distribution measurements, one is interested in the intersection of a straight line (i.e. chord) with the hyperboloid surface given by Eq. (6). The parametric equation of a straight line characterized by a point \((X_0, Y_0, Z_0)\) and a direction given by the direction cosines \(U\), \(V\), and \(W\) is:

\[
X = X_0 + Ut, \\
Y = Y_0 + Vt, \\
Z = Z_0 + Wt,
\]

where \(t\) is a parameter. To find the intersection with the hyperboloid surface:

\[
aX^2/a^2 - bZ^2/b^2 = 1,
\]

we need to substitute the parametric equations into this equation and solve for \(t\). This will yield the points of intersection between the straight line and the hyperboloid surface.

Fig. 17. Trilinear interpolated isosurfaces within several adjacent cubes, showing how the surface connects. (a) A pattern 3 on top of a pattern 3c. (b) A pattern 2 on top of a pattern 5c. (c) Four cubes of patterns 2, 11, 10, and 2c, from left to right. The numbers in parentheses are the configuration numbers. The gray-levels used to obtain these surfaces are given in Table 2.
\[ W u^2 + v^2 + w^2 = 1 \]
can be substituted into Eq. (6) to find the intersection points. Eqs. (5) and (6) are given for a cube of unit size and with the coordinate system origin at \( B_0 \). Therefore, the parametric equation of the straight line must be corrected to take into account the cube size. With \( V_x \), \( V_y \), and \( V_z \) being the voxel sizes of the initial image, the straight line is characterized by

\[
\begin{align*}
  x &= \frac{X_0 + DU}{V_x} \\
  y &= \frac{Y_0 + DV}{V_y} \\
  z &= \frac{Z_0 + DW}{V_z}
\end{align*}
\]

In Eq. (7), \( D \) is the parameter of the straight line, that is the distance from any point \((x, y, z)\) on the line to \((X_0, Y_0, Z_0)\). If \( D \) is negative, the point is in the backward direction. Note that \((x, y, z)\) and \((X_0, Y_0, Z_0)\) are relative to the cube coordinate system with origin at \( B_0 \). Therefore, \((X_0, Y_0, Z_0)\) must be translated from its absolute location prior to be injected into Eq. (7). Using Eq. (7) into Eq. (6) gives a cubic equation with \( D \) as unknown, i.e.

\[
aD^3 + bD^2 + cD + d = 0.
\]
4. Results and discussion

4.1. Surface-area measurement

The total surface area of the marrow spheres within the mathematical bone sample was measured within the voxel-based image as well as in images with both the DIR-MC- and the REV-MC-based techniques. The results are listed in columns 3, 4, and 5 of Table 1. The convergence of the area to the exact value is shown in Fig. 18. The solid straight line is the exact value given by Eq. (2). The dotted straight line corresponds to the exact value multiplied by 1.5. One can see, as predicted in previous studies [26,27], that the measurement of the surface area through the voxel-based images overestimates the true value by 50% at high resolution (small voxel size). On the other hand, both the DIR-MC- and the REV-MC-based techniques converge to the exact value. At 60 µm, which is about what is currently used for NMR microscopy imaging of trabecular bone, the approximation is within 1.5%. At 30 µm, it is down to only 0.4%.

The DIR-MC- and the REV-MC-based techniques differ slightly as shown in Fig. 18. At large voxel sizes, the space between the spheres of the model (representing the bone trabeculae) is thin compared to the voxel size. Therefore, an ambiguous face with a direct segmentation will generate two long segments next to one another. A reverse segmentation will generate two short segments located at two opposite corners of the voxel. This explains why the REV-MC-based surface area is smaller than the DIR-MC-based surface area at large voxel sizes. At small voxel sizes, the situation is inverted and the REV-MC-based surface area is smaller than the DIR-MC-based surface area at large voxel sizes. At 60 µm, these spikes tend to disappear, but a sharp peak (not entirely shown in Fig. 19) remains present at the beginning of the distribution (short chords). The voxel effects increase the frequency of the short chords and the mean chord length is reduced by 30% at image resolutions of both 60 and 24 µm.

At large voxel sizes (e.g. 375 µm), the three MC-based techniques do not give better results than the voxel-based technique. They are even worst if one compares mean chord lengths. A poor image resolution, even using a good interface-definition technique will not give good results. At 150 µm, the shape of the three MC-based distributions become close to the true distribution. The mean chord lengths are all within 5% of the true mean. Below 60 µm, they are within 1% of the true mean, and the curves match almost perfectly, except for a residual voxel effect discussed later.

Let us first compare the two polygon techniques (DIR-MC and REV-MC). They both show a voxel effect that overestimates the frequency of short chords. The effect extends from 0 to a chord length about twice the voxel size. The angles between the triangles are not flat. Even though they tend to become flat as the resolution improves, there still exists a small angle and these zigzag-like surfaces will always increase the frequency of short chords for the trajectories traveling close and parallel to the boundary. The voxel effect of the DIR-MC-based model seems to be different than for the REV-MC-based model, especially at resolutions of 150 and 60 µm. As discussed previously, and as can be seen in Fig. 4b, the direct technique better suits the geometry composed of dotted cavities surrounded by a non-dotted network. That is exactly the situation of the mathematical bone sample where the spheres are the dotted regions. Furthermore, Fig. 5b and c will help explain the differences of short chord frequencies. The three direct examples of Fig. 5b look more like a sharp piece of bone (non-dotted region) that penetrates into a block of marrow (dotted region). This sharpness will produce an excess of short bone chords. In Fig. 5c, on the other hand, the reverse segmentation of the same examples look more like a sharp piece of marrow that penetrates a block of bone, and will produce an excess of short marrow chords. That is exactly what happens within the mathematical bone sample.
The reverse segmentation produces more short marrow chords than the direct segmentation. This explains the difference at the 150 and 60 \( \mu m \) image resolutions between the DIR-MC- and REV-MC-based techniques. The asymptotic decider discussed previously would probably give a voxel effect somewhere in between the two techniques, as it combines both models. At 24 \( \mu m \), the voxel size becomes so small compared to the object size that the ambiguous faces become rare and there are few differences noted between the DIR-MC- and the REV-MC-based curves.

The TRI-MC-based technique should eliminate the angles between the triangles, at least within region interior to the MCs. Fig. 17 shows that there are still angles at the cube interfaces and that a small voxel effect is expected. However, in Fig. 19, it seems that the effect is worst (even though it remains small) than with the traditional MC-based techniques (DIR or REV). This observation is attributed to the concavity of the isosurface. In Fig. 14a, the surface is generated from a configuration 4 that has one vertex set to 255 and seven vertices set to 0. The dotted vertex is on the same side of the surface than the center of curvature. Configuration 251 is the complement of configuration 4. With one vertex set to 0 and 7 vertices set to 255, it would generate the exact same surface as configuration 4. In this

Fig. 19. Chord-length distributions measured through the images of the mathematical bone sample for four different voxel sizes. The solid line is the exact distribution calculated from Eq. (3). The three MC-based techniques (DIR-MC, REV-MC, and TRI-MC) are compared with the measurement through the voxel image.
case, however, the center of curvature is the non-dotted vertex. As a consequence, the concavity of the isosurface depends also on the configuration of the cube whereas it should depend only on the curvature of the object. The TRI-MC-based model artificially curves the flat surfaces. Since, at small voxel size, all surfaces tend to appear flat (relatively to the voxel size), this effect is expected to be more important at high resolution (small voxel size), as seen at 24 μm in Fig. 19. The result is that a flat surface that extends over a large number of cubes is likely to be represented by an undulated surface. An example is shown in Fig. 20 for a 2D image. Even though the real boundary may be a straight line, the generated curve is undulated. These undulations generate artificial short chords that are responsible for the small voxel effects seen at high resolution with the use of the TRI-MC-based model.

One can see in Fig. 19 that the voxel effect introduced by the TRI-MC-based technique affects the distribution from 0 to a chord length of about two voxel sizes. At high resolution, all distributions measured within smooth surfaces should drop to zero at zero chord length [24,26]. Besides, the shape of the exact distribution is almost linear at short chords. Therefore, a solution to remove this effect is to change the beginning of the distribution into a linear distribution that smoothly connects to the measured distribution at a chord length equal to two voxel sizes. This procedure was done for the four images of Fig. 19 and they are compared to the exact distribution in Fig. 21. Fig. 21 represents the corrected distributions measured with the trilinear technique. At 375 and 150 μm, the correction is not a significant improvement. However, at 60 μm, the distribution is very close to the exact one. At 24 μm, no distinct differences are noted between the exact and the measured distributions.

5. Conclusions

Three techniques based on the MC algorithm were developed to better represent the boundary between bone trabeculae and marrow cavities when 3D images of trabecular bone are coupled with computer programs for radiation transport or chord-length distribution measurements. A mathematical sample of trabecular bone was used to test these techniques. The DIR-MC and REV-MC techniques are based on the extended MC algorithm that solves the ambiguity problem by providing a specific design for each complement pattern. The two methods give an excellent preservation of the surface area of the boundary. The third technique is based on the trilinear interpolation of the gray-level over each MC. This technique eliminates the data size problem of the MC algorithm as well as the generation of the triangle list. It uses the original gray-level image and the isosurface is constructed locally by the computer code processing the image. It solves the ambiguity problem at the MC-face level as does the asymptotic decider [48], and it simplifies the computer calculations within the image since only one equation is used per MC. The surface area was not measured using this technique, but it is expected to give similar results as the DIR-MC- and the REV-MC-based methods. The three models were used to measure chord-length distributions through the simulated images. When the voxel size reaches values typically attained via NMR microscopy [19,75], the distributions are in good agreement with the reference distribution. A slight voxel effect is still present, but it can be easily removed, as it only affects the very short chords (below two voxel sizes) where a majority of distributions should be approximately linear.

As a conclusion, one of the three models should be used each time a computer simulation is to be performed in direct connection with a digital image. The trilinear interpolation technique is the simplest and will be used to couple future NMR microscopy images of trabecular bone with Monte Carlo simulations.
Carlo radiation transport codes. It will also be chosen to measure new chord-length distributions through these same samples. The discrepancy found when using the initial voxel-based model will be reduced from more than 30% [25,27] to only a few percent.

Another application of the trilinear technique is for visualization purposes. In the case of a trabecular bone image, the current MC surfaces necessitate ∼ 7 times the original image size to store the triangles. A visualization program based on the trilinear technique would reduce this memory problem. However, the display of each individual cube as a hyperboloid surface may pose other challenges. The first concerns the computer time required to calculate the isosurface. A second problem is the presence of singularities among hyperboloid surfaces. A visualization program must detect these singularities prior to display. The examples proposed in Figs. 14–17 have been carefully chosen so that the singularities fall outside the cube so that they do not appear within the figures.

Finally, the current technique was developed within an image containing two media separated by a single isosurface. When imaging techniques are used to show organs of the body, segmentation can be applied to separate the different body tissues. The result is an image for which the gray-levels are used to represent different regions of the body instead of a distance from the isosurface. With a 256 gray-level image, up to 256 different tissues or organs can be segmented within that image. Each half integer can be seen as the isovalue that separates two organs and each isosurface would be on a two gray-level scale. The connecting points along the edges of the cubes no longer need to be interpolated, but the trilinear technique can still be used to represent the isosurface. A problem is that the same cube may be crossed by several isosurfaces and a more complex algorithm would need to be developed; nevertheless, the technique is perfectly suitable for these imaging applications.

Acknowledgements

This work was supported by the US DOE Nuclear Engineering Education Research (NEER) Grants DE-FG07-99ID13764 and DE-FG07-02ID14327 with the University of Florida.

References


Didier A. Rajon attended Pierre Mendes France University in Grenoble, France and graduated in 1984 with a Diplôme Universitaire de Technologie in computer programming and analysis. After several years of experience in software companies specializing in industrial processes for the manufacturing industry, he attended a University-level preparation for entrance to the French Engineering Schools. He graduated with Honors in 1997 from the Institut National Polytechnique de Grenoble with an Engineering degree in Nuclear Physics. After a 6-month internship at the Nuclear and Radiological Engineering Department of the University of Florida, he enrolled in the Health Physics program at UF in January 1998. He graduated in December 1999 with a Masters of Science and went on to receive his PhD in December 2002. His doctoral research focused on image-voxelization effects in bone marrow dosimetry using particle transport simulations through NMR images of trabecular bone. Didier is currently working as a Postdoctoral Research Associate at the Department of Neurological Surgery at the University of Florida. His current research focuses on designing customized positioning fixtures for neurological-guided surgery from high-resolution multiplanar imaging data such as CT.

Wesley E. Bolch is a Professor of Radiological and Biomedical Engineering at the University of Florida. He completed his doctoral degree in health physics from the University of Florida in 1988. His PhD research was performed in the area of microdosimetry at the Oak Ridge National Laboratory. From 1989 to 1995, Bolch held the position of Assistant Professor and then Associate Professor in the Department of Nuclear Engineering at Texas A and M University. In 1995, he joined the Department of Nuclear and Radiological Engineering at the University of Florida in Gainesville, teaching and performing research in both the health physics and medical physics graduate degree programs. His current areas of research include (1) development of patient-specific skeletal dosimetry models using advance imaging techniques such as NMR microscopy and microCT, (2) assessment of organ doses to patients in pediatric radiology and pediatric cardiology through the development of tomographic anatomic models, (3) uncertainties in internal dosimetry models, and (4) radiation damage to DNA. He has been a member of the Society of Nuclear Medicine’s Medical Internal Radiation Dose (MIRD) committee since 1993, and currently serves as a member of the International Commission on Radiological Protection’s (ICRP) Task Group on Dose Calculations (DOCAL).