Area of and volume enclosed by digital and triangulated surfaces

Ingela Nyström[a], Jayaram K. Udupa[b], George J. Grevera[b] and Bruce E. Hirsch[c]

[a]Centre for Image Analysis, Uppsala University, Uppsala, Sweden
[b]Medical Image Processing Group, Dept. of Radiology, University of Pennsylvania, Philadelphia, PA, USA
[c]Dept. of Anatomy and Cell Biology, Temple University School of Medicine, Philadelphia, PA, USA

ABSTRACT

We demonstrate that the volume enclosed by triangulated surfaces can be computed efficiently in the same elegant way the volume enclosed by digital surfaces is computed by digital surface integration. Although digital surfaces are good for visualization and volume measurement, their drawback is that surface area measurements are inaccurate. On the other hand, triangulated surfaces give more accurate surface area measurements, but volume measurements and visualization are less efficient. The T-shell data structure previously proposed retains advantages and overcomes difficulties of both the digital and the triangulated approaches. We create a lookup table with area and volume contributions for each of the 256 Marching Cubes configurations. When scanning the shell (e.g., while creating it), the surface area and volume are incrementally computed by using the lookup table and the current x co-ordinate, where the sign of the x component of the triangle normal indicates the sign of the volume contribution. We have computed surface area and volume for digital and triangulated surfaces for digitized mathematical phantoms, physical phantoms, and real objects. The computations show that triangulated surface area is more accurate, triangulated volume follows digital volume closely, and that the values get closer to the true value with decreasing voxel size.

Keywords: digital boundary, surface tracking, shell structure, marching cubes, isosurface, shape parameters, boundary measurement, volume measurement, rendering

1. INTRODUCTION

Currently, a number of imaging devices that generate volumetric (3D) images are available and are extensively used especially in medical imaging. There is usually an *object* of interest for which such an image is generated, for example, an organ, a tissue component, a tumour, a physical phantom, or a mathematical phantom. The 3D imaging operations typically performed on these images may be classified into four groups: *preprocessing*, *visualization*, *manipulation*, and *analysis*.[1] The preprocessing operations aim at improving and/or extracting the object of interest. The purpose of visualization is to assist humans in perceiving and comprehending the 3D object. The manipulation operations allow humans to interactively change the objects. Finally, the analysis operations are intended to quantify the object, e.g., geometrically and morphologically. In this paper, we touch on some aspects of visualization, but concentrate mainly on analysis, particularly on estimating surface area and volume.

An object may be represented by its 3D boundary surface. The two classical approaches in extraction of geometric representations for visualization are *digital surface* tracking, as in the approach to boundary detection,[2] and isosurface construction by polygonal surfaces, commonly triangulated by the *Marching Cubes* (abbreviated from now on by MC) algorithm.[3,4] Actually, there is a close relationship between the two approaches, since a digital surface can be transformed directly into a triangulated isosurface.[5,6]

Other author information -
E-mail: `ingela@cb.uu.se`, `jay@mipg.upenn.edu`, `grevera@mipg.upenn.edu`, `bhirsch@tuspm.temple.edu`

Digital boundaries and surfaces have many elegant geometrical and topological properties that lead to their efficient construction (from image data),[7] ultra fast rendering,[8, 9] manipulation,[8] and measurement,[2] and to elegant generalizations of the concepts to higher dimensional spaces.[10] The MC algorithm[3, 4] was an attempt to bridge the gap between digital surface methods (which predate the MC family of algorithms) and the traditional polygonal approaches established in computer graphics. The spirit of this bridging was identified in Ref. 11, wherein it was recognized that the surfaces produced by MC algortihms have a digital embedding, and therefore, can be cast within the same efficient framework, namely that of a *shell*,[7, 12] that is used for efficiently representing, rendering, manipulating, and measuring digital surfaces. This recognition led to the idea of the *T-shell*[11] (short for triangulated shell) for dealing with triangulated surfaces.

In spite of their many desirable properties, one drawback of digital surfaces has been the lack of a simple method to accurately estimate their area. The identity between the T-shell and the shell provides a natural means to overcome this problem. The volume enclosed by a digital surface, on the other hand, can be computed trivially by digital surface integration[2] while tracking the surface in a gray or binary image (or after its creation) without having to visit (and count) any voxel in its interior. Conversely, while area computation of triangulated surfaces is trivial, the computation of the volume enclosed by them is challenging. The main purpose of this paper is to demonstrate that, via T-shell, we can combine the strengths and overcome the difficulties of both approaches and compute surface area and volume trivially and efficiently, and still retain the digital setting. We describe the basic concepts of the two methods of surface representation and of T-shells in Section 2 and the new method of area and volume estimation in Section 3. Our experiments and results comparing the digital and triangulated methods are presented in Section 4, and Section 5 states our conclusions.

2. OBJECT REPRESENTATION BY BOUNDARY

We refer to any gray-level volume (3D) image as a *scene* and represent it by a pair $\mathcal{C} = (C, f)$, where C, called the *scene domain*, is a rectangular 3D array of cuboidal volume elements, called *voxels*, and f is a function that assigns a value $f(v)$ to every voxel $v \in \mathcal{C}$ called *scene intensity*. The range of f is usually a set of integers. When this range is $\{0, 1\}$, we call $\mathcal{C}$ a *binary scene*. Scenes contain information about certain objects of interest, which are defined and delineated in the scene by a scene segmentation operation. The result sought very frequently is either a binary scene or a surface representation of the object. For what we wish to accomplish in this paper, it is the latter form that is relevant. In this section, we shall outline several common methods of representing objects by surfaces, wherein object information is derived from scenes.

2.1. Digital Surface Representations

We will consider two digital surface representations: (i) by a set of voxels,[8] and (ii) by a set of oriented faces of the voxels.[10] In both cases the boundary elements are all of identical size and shape, which makes storage in efficient and clever data structures (called *shells*) possible. Efficiency comes from the fact that the elements need not be stored explicitly in terms of their co-ordinates since they are all identical in size and shape, and have only a small number of distinct orientations.

In the first case, the digital surface is represented by a set of voxels in the object that have a 6-neighbor (i.e., a face neighbor) in the background. This surface fulfills properties such as closure, orientedness, and connectedness. The information that is associated with each boundary voxel, consists of the x co-ordinate of the voxel (y and z co-ordinates are given implicitly by the data structure) and a code describing the configuration of its 6-neighborhood (face connected). Hence, it is not only known that a voxel is facing the background, but also whether it is facing the background in positive or negative (x, y, z) directions. This is useful when computing the volume enclosed by the surface as will be described in Section 3.2.

In the second case, the surface is represented by the set of oriented faces of voxels. The number of boundary elements becomes larger. The faces are assigned an orientation, which means that a face with the normal vector pointing in the $+x$ direction is distinguished from a face at the same location with the normal vector pointing in the $-x$ direction.

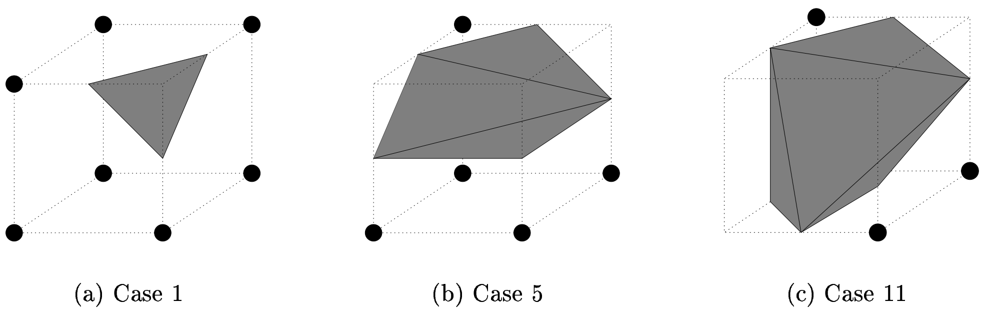


Figure 1. Three m-cubes of $2 \times 2 \times 2$ voxels, where • voxels are inside the object (value 1) and the other voxels are outside (value 0). The surfaces for these configurations consist of one, three, and four triangles, respectively. (Fourth • not visible in (c).)

2.2. Polygonal Surface Representation

A commonly used approach to polygonal surface representation is by a set of triangles, as in the MC algorithm.[3, 4] An *m-cube* (to indicate an individual marching cube, sometimes also called a *cell* in the literature) in a given scene domain, is the cube bounded by the centers of the eight voxels in a $2 \times 2 \times 2$ neighborhood in the scene domain. Hence, each corner of the m-cube corresponds to a voxel. The 6-, 18-, and 26-adjacency between two voxels can be described in terms of the m-cube as follows: connected by an edge of the m-cube, having a face of the m-cube in common, and having the whole m-cube in common, respectively.

If object and background are considered (i.e., the set of voxels with value 1 and 0, respectively), the possible number of configurations of the eight voxels forming an m-cube in a binary scene are $2^8 = 256$. Each configuration consists of zero to four triangles[4] constituting the surface passing through the specific m-cube. See Figure 1 for examples. The configurations are commonly grouped into symmetry and complementary cases, resulting in (14 or) 15 cases.[4, 13] There are two configurations with no triangles, which is when the m-cube is situated completely inside or completely outside the object.

The approximated surface in the m-cube is computed from some interpolation of the scene intensities of the voxels. The interpolation results in *intersection points* on the edges of the m-cube for the triangle vertices. In the simplest case, the intersection points are considered to be midway at the center of the m-cube edges. The midway solution will be used in the rest of the paper for simplicity.

The correct connection among intersection points for forming triangles is tricky for some configurations of voxels. This problem in the original MC algorithm was pointed out by Dürst in 1988.[14] If the same surfaces are used for such m-cube configurations and for their complementary cases (i.e., when object and background are inverted), closed surfaces are not produced. The problem requires careful consideration since any object with finite support must be enclosed by a closed surface to make topological sense. Further, the act of measuring the volume enclosed by a surface is meaningful only when the surface is closed. This problem has been examined by numerous investigators in the literature.[13, 15, 16] We have taken the approach where the problem is avoided by using more complex surfaces for these cases,[15] exemplified in Figure 2.

2.3. T-Shell Representation

In previous shell data structures, the elements have been voxels or oriented faces of voxels[8, 10] (Section 2.1). In the *T-shell*,[11] the elements are m-cubes as described in Section 2.2. In a manner similar to previous shell structures, the T-shell contains only the m-cubes on the boundary of the object, which are stored row-wise. This means that m-cubes completely inside or completely outside the object are not stored. With each element the x co-ordinate of its front-upper-left voxel is stored (other co-ordinates can be derived from this information), as well as the m-cube configuration code and the normal vector computed for the midpoint of the m-cube (one normal per triangle or one normal per triangle vertex may also be stored).

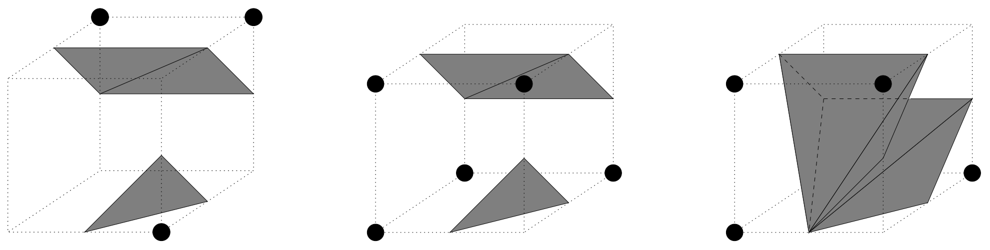

(a) Case 6 (b) Case 6 complementary (c) Solution for closed surface

Figure 2. (a) A tricky m-cube configuration. (b) The complement of (a), which would result in surfaces that are not closed. (c) A set of five triangles that will produce a closed surface for this example.[15] (Fifth • not visible.)

3. SURFACE AREA AND ENCLOSED VOLUME

We are interested in the following measurements for any object defined in a given scene:

- true surface area, A (when known)
- area of the surface of the object approximated by a digital surface, DA
- area of the surface of the object approximated by a triangulated surface, TA
- true enclosed volume, V (when known)
- volume enclosed by a digital surface approximation of the object surface, DV
- volume enclosed by a triangulated surface approximation of the object surface, TV

We will describe the underlying idea in the 2D case first, where perimeter and area are computed in an incremental fashion by scanning the boundary of the object. Understanding the 2D case helps when we later extend the concepts to the 3D case. The intuitive idea extends easily, but as is usually the case when going from two dimensions to three dimensions, it is not a pure generalization as m-cubes consist of contributions from one or more triangles, unlike in the 2D case, where each configuration consists of only one contribution.

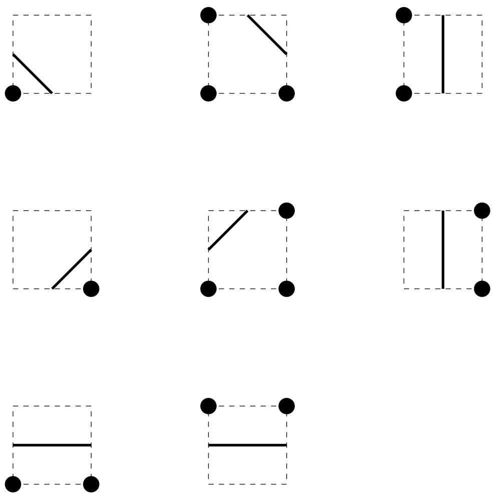
3.1. Surface Area

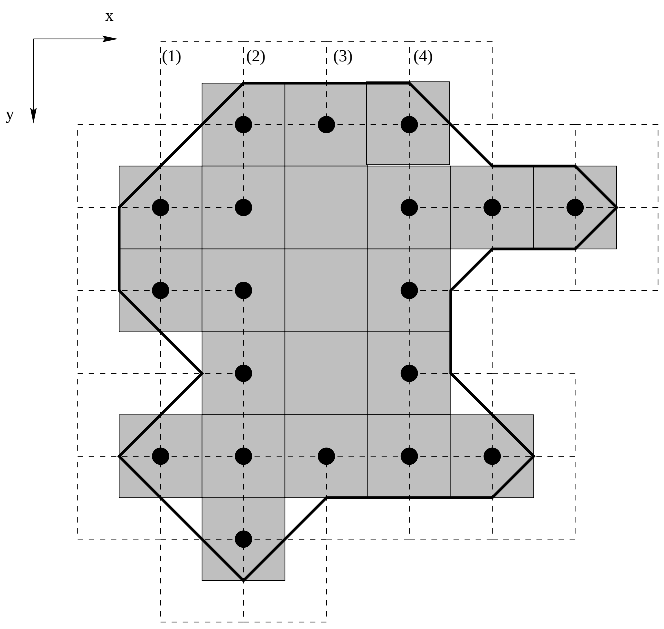
Surface area estimation has been studied earlier,[17, 18] but not with the approach to simultaneously also compute the volume enclosed by the same boundary representation. Our purpose is to use a boundary representation giving accurate results for both measures simultaneously and with minimal computation.

Underlying Idea in Two Dimensions

In the 2D case, the goal is to compute the perimeter of an object, i.e., the boundary length. One approach is to count how many horizontal, vertical, and diagonal steps are taken between pixel centers when scanning the boundary. Each type of step can then be assigned a weight. Note that the weight 1 for horizontal and vertical steps and weight $\sqrt{2}$ for diagonal steps are not optimal when measuring lengths of line segments.[19, 20]

In the purely digital case, one estimate of the perimeter is to count the number of pixel edges between the object and the background. This results in an overestimate. This can be improved by finding a better approximation to the boundary. One such improvement can be seen in Figure 3. An m-square is the 2D analog to the m-cube, i.e., a square bounded by the centers of the four pixels in a 2×2 neighborhood. In this representation, the contribution to the boundary length may be precomputed and stored in a lookup table for the different configurations. This idea will now be extended to the 3D case.


(a) 8 of 16 possible m-squares



(b) A 2D object example

Figure 3. (a) Examples of configurations of marching cubes in two dimensions, i.e., m-squares. (b) Pixels are gray-coloured. The boundary pixels are marked with $\bullet$. Only the m-squares (dashed lines) need to be stored in the shell. The boundary defined by the m-squares is delineated with solid lines.

Extending to Three Dimensions

In the purely digital case, where the surface is represented by the faces of the voxels at the boundary, the number of faces gives an estimation of the surface area, DA. Similar to the 2D case, this is an overestimate. If the surface is represented by triangles, a better surface area estimate, TA, is obtained.

Let S_T be a T-shell that represents a 3D boundary surface of an object in a given scene. Let c denote an m-cube of S_T and let $\gamma(c)$ be the MC configuration of c. Our idea is to create a lookup table that stores the contributions of the different MC configurations to surface area and then use this table to compute the area of the surface represented by S_T. Given the vertices P_1, P_2, and P_3 of any triangle, its area A_t can be computed by the formula

$$A_t = \frac{1}{2}\left|\overrightarrow{P_1P_2} \times \overrightarrow{P_1P_3}\right|, \tag{1}$$

where $|.|$ is the length (the magnitude) of the normal vector to the plane defined by the triangle. Note that the sign of the area will become positive or negative depending on whether the vertices are numbered clockwise or counter-clockwise. Our surface area estimate, TA, is incrementally computed for each m-cube of the object:

$$TA = \sum_{c \in S_T} \sum_{t \in \gamma(c)} A_t. \tag{2}$$

Treating the configurations one by one, according to Ref. 4, is: "possible but tedious and error-prone." We, however, found it more convenient to create a lookup table with entries for each of the 256 configurations (this is done only once any way). Hence, we do not need to take rotation, symmetry, or complementary configurations into account when scanning the T-shell. Recall that m-cubes completely inside or completely outside the object are not stored in the T-shell; they will contribute neither to the area nor to the volume (*sic!*) measurements. That is the beauty of the closed (Jordan) surfaces.

3.2. Enclosed Volume

In Ref. 21, Gauss' theorem is used for volume computations: the integral of the normal component of any vector over any closed surface equals the integral of the divergence of the vector over the volume enclosed. Their volume calculations involve *three* summations along the x, y, and z directions with quite complex weighting of the terms that reflect the orientation of the object. The volume is calculated by summing the vector product of the centroid, area, and normal of all triangles in the mesh surface. We observe that summations along three directions are not necessary. It is sufficient to sum along only one direction, exactly as in the purely digital surface case.[2] We further simplify this process by reducing the computations along the only direction to table lookup operations.

Underlying Idea in Two Dimensions

In the 2D case, the goal is to compute the area of the object. In the purely digital case, a simple way to compute object area is to count all pixels of the object in a connected component labelling algorithm. However, boundary tracking methods are more economical since they visit only elements in the vicinity of the boundary. (Such methods can also label connected components! See Ref. 22.) Therefore it makes sense (even in the digital case) to seek ways of estimating area (volume in the 3D case) from boundary information only. The area is accumulated as a sum of x co-ordinates, where the contribution is $-x$ if the boundary element is facing the background in the negative direction, $+x$ if the boundary element is facing the background in the positive direction, and no contribution if the boundary element is facing the background in the y direction.

We described in Section 3.1 that the perimeter estimation is more accurate if it is computed from the m-squares than if computed from pure pixels. For consistency, we believe that an area estimation should also be computed for the same representation in a quantitative analysis. An m-square is represented by two entities — the x co-ordinate of its upper-left pixel and the configuration of its pixels. In principle, the area contribution for a certain row of the object is simply the x co-ordinate farthest to the right subtracted by the x co-ordinate of the edge farthest to the left. See the 2D example in Figure 3. The contributed area for the different configurations is precomputed and stored in a lookup table.

The edge passing through the m-square is projected onto the y axis. The length of this projection is multiplied by the x co-ordinate at the midpoint of the edge. The sign of the x component of the normal vector for the edge is sufficient to determine whether it is a positive or a negative contribution to the area. If the x component of the normal vector is positive (Figure 3(a) top), the incremental area contribution is positive. If this component is negative (Figure 3(a) middle), the incremental area contribution is negative. If the x component of the normal vector is equal to 0 (i.e., the edge is parallel to the x axis (Figure 3(a) bottom)), then such a configuration will not contribute to the area (but contributes only to the perimeter). From Figure 3(b), it may appear that m-squares (2) and (3) should really contribute to the area, but they will not. Their areas are accounted for by m-squares (1) and (4), which have negative and positive normal components in the x direction, respectively.

Extending to Three Dimensions

In the purely digital case, where the surface is represented by the faces of the voxels at the boundary, the volume DV enclosed by the digital surface can be computed similar to the 2D case by accumulating $-x$ and $+x$ co-ordinate values of only those voxel faces that are orthogonal to the x axis, while scanning the boundary.[2]

For triangulated surface representation, we compute lookup table entries for each of the 256 (rather 254 since the configurations of entirely inside and entirely outside may be disregarded) m-cube configurations. Hence, both incremental area and incremental volume for each configuration will be known for the same representation. We do not distinguish among rotational, symmetrical, and complementary cases. Actually, we take advantage of the fact that some of the configurations do not contribute to the volume, while a rotation of the same configuration does. This is specific to the scanning direction we have chosen, namely x (as opposed to y or z). This is a good reason for storing all configurations instead of storing a reduced number of configurations.

The incremental contribution V_c to the total volume from any m-cube c of configuration $\gamma(c)$ can be expressed as

$$V_c = \Delta V_{\gamma(c)}(x) + \delta(\gamma(c)), \tag{3}$$

where x is the x co-ordinate of the particular m-cube in the T-shell, $\Delta V_{\gamma(c)}$ is the volume contribution from all triangles in configuration $\gamma(c)$, and $\delta(\gamma(c))$ is a constant that depends only on the configuration of c and not on c *per se*. This can be further divided into incremental contribution for all triangles t for the specific configuration $\gamma(c)$. Hence, in a configuration $\gamma(c)$, the net contribution V_c is precomputed as:

$$V_c = \Delta V_{\gamma(c)}(x) + \delta(\gamma(c)) = \sum_{t \in \gamma(c)} [\Delta V_t(x) + \delta_t]. \tag{4}$$

Our enclosed volume estimate, TV, for any T-shell S_T, is incrementally computed for each m-cube of the object:

$$TV = \sum_{c \in S_T} V_c = \sum_{c \in S_T} \sum_{t \in \gamma(c)} V_t. \tag{5}$$

Similar to the 2D case, we compute the signed volume under a triangle. The volume V_t under a triangle t is the projected area of the triangle onto the $y - z$ plane multiplied by the x co-ordinate of the centroid of the triangle. See Section 3.3 for formulas.

The property that the sign and the magnitude of the normal vector describes the contribution is exactly the same as for the digital surface approach.[2] The sign of the x component of the normal vector is sufficient to determine whether it is a positive or a negative contribution to the volume. If the x component of the normal vector is positive, the incremental volume contribution is positive for that triangle. If this component is negative, the incremental volume contribution is negative. If the x component of the normal vector of the triangle is equal to 0 (i.e., the triangle is parallel to the x axis), then the triangle will not contribute to the volume (but contributes only to the area). The volume will be accounted for by other m-cubes that have triangles with non-zero normal component in the x direction.

A rather complex example is the configuration with four voxels inside and four voxels outside the object, as illustrated in Figure 1(c). In this particular configuration, two of the four triangles yield positive and the other two triangles yield negative volume contribution, due to the direction of their normal vectors. Another complexity is that the projected triangle areas are partially overlapping. However, this is not a problem, since the *net* volume and the *net* area contribution for a configuration are only a summation of the respective contributions for each triangle. In this specific case, the net volume contribution actually is 0(!), while for other rotations of the same configuration of voxels this is not true.

3.3. Notes on Triangle Geometry

Given three triangle vertices P_1, P_2, and P_3, the following entities are needed and computed for any triangle t:

The centroid $\mathbf{C}$ for t, where the x component is given by

$$\mathbf{C}_x = \frac{P_{1x} + P_{2x} + P_{3x}}{3}. \tag{6}$$

The normal vector for t

$$\mathbf{N} = \overrightarrow{P_1P_2} \times \overrightarrow{P_1P_3}. \tag{7}$$

The area of t

$$A_t = \frac{1}{2}\left|\overrightarrow{P_1P_2} \times \overrightarrow{P_1P_3}\right|, \tag{8}$$

where $|.|$ denotes the length (the magnitude) of the vector.

The area A_p of the projection of the triangle along the x axis onto the $y - z$ plane

$$A_p = A_t \frac{\mathbf{N}_x}{|\mathbf{N}|} = \frac{1}{2}\mathbf{N}_x, \tag{9}$$

where the last expression certainly simplifies the computations.

Finally, the volume under t

$$\Delta V_t = \mathbf{C}_x A_p. \tag{10}$$

The above equations are valid if the voxels are of size $1 \times 1 \times 1$, i.e., the m-cubes are real cubes. To incorporate the proper resolution and handle anisotropic data, a scale vector $\mathbf{S} = (\mathbf{S}_x, \mathbf{S}_y, \mathbf{S}_z)$ is needed. $\mathbf{S}_x$ and $\mathbf{S}_y$ give the pixel size and $\mathbf{S}_z$ gives the slice spacing, i.e., the voxel size is $\mathbf{S}_x \times \mathbf{S}_y \times \mathbf{S}_z$. The normal vector $\mathbf{N}$ is modified by scaling the components of the vectors in Eq. 7 above: $\mathbf{S}_x(P_{1x} - P_{2x})$, $\mathbf{S}_y(P_{1y} - P_{2y})$, and so on. Also, when computing the volume under the triangle (Eq. 10), the scale in the x direction must be used to modify the x component of the centroid: $\Delta V_t = \mathbf{S}_x \mathbf{C}_x A_p$.

Equations 6–10 above are general for any triangle. We have computed a lookup table specifically for the case where the triangle vertices are midway on the m-cube edges. Equations 8, 9, and 10 are summed for each triangle in the specific configuration resulting in the three net contributions stored in the lookup table. The projected area times the current x co-ordinate plus the volume under all triangles gives the volume contribution for an m-cube. Remember to scale the x co-ordinate with the voxel size in the x direction when computing the contribution for an m-cube configuration.

4. EXPERIMENTS AND RESULTS

We have measured surface area and volume for both digital surfaces as well as triangulated surfaces for mathematically defined synthetic objects, physical phantoms, and real objects. This section describes our experiments and presents the results.

4.1. Mathematical Phantoms

Our first object is a ball. The digitization of a ball centered at $(x_0, y_0, z_0) \in \mathbb{R}^3$ of radius $r \in \mathbb{R}$ is generated by the following equation:

$$f(x,y,z) = \begin{cases} 1, & \text{if } (x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2 \leqslant r^2, \\ 0, & \text{otherwise,} \end{cases} \qquad \text{where} \quad (x,y,z) \in \mathbb{Z}^3. \tag{11}$$

We have studied the performance of surface area and volume estimation methods for 5,310 digitized balls with varying radii. In this work, we chose 177 different sizes in the range from radius 0.45 to 79.65 voxels. For each size, 30 balls were generated (by Eq. 11) with randomized alignment in the digitization grid. Thereafter, surface area and enclosed volume were estimated using the digital and the triangulated approaches. The corresponding mean values (over 30 data sets) are presented for each size in Figure 4. It can be noted that balls with radius less than 10 voxels give unreliable measurements, urging us to be careful when measuring features of digitized small objects. The digital volume, DV, and the triangulated volume, TV, both coincide well with the true volume, TV always being slightly smaller than DV, due to the "cutting of corners" in convex objects. The digital area, DA, is not shown in this plot, because of its large overestimate (close to 50%). The triangulated area, TA, on the other hand, is a much more accurate estimate. The plot shows convergence to an overestimate of 8.8% for large balls. Ways to improve the surface area estimates by choosing different triangulations and assigning optimized local weights have been reported.[18]

The radius expressed in terms of the number of voxels has the same meaning as resolution. Imagine that the ball is 10 cm in diameter and is digitized at 10 different resolutions. Let the voxel size be in the interval 0.2–2.0 mm in x, y, and z directions. The resolution then varies from 250 voxels down to 25 voxels. We have examined how our measurements for different voxel sizes compare with truth, $A = 4\pi r^2 = 314.16$ cm^2, and $V = \frac{4}{3}\pi r^3 = 523.60$ cm^3, see Figure 5.

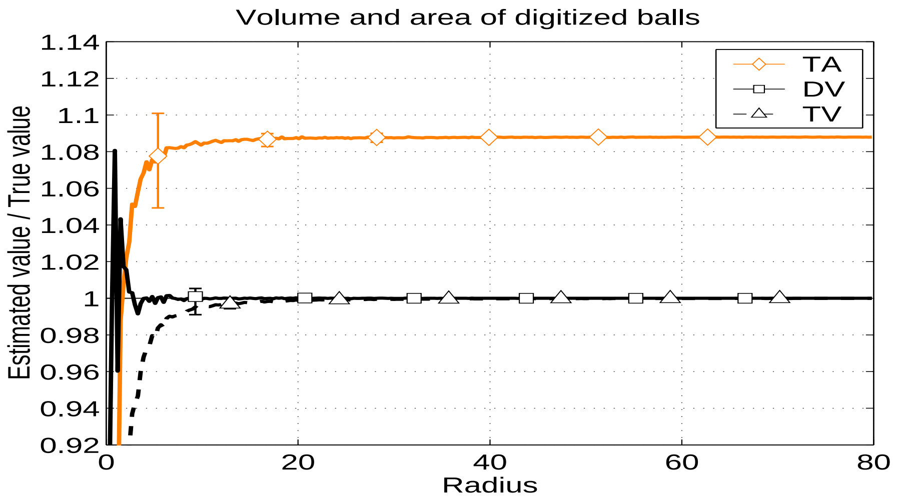


Figure 4. Estimated surface area and volume divided by the true surface area and volume, respectively, for 5,310 digitized balls of increasing radius.

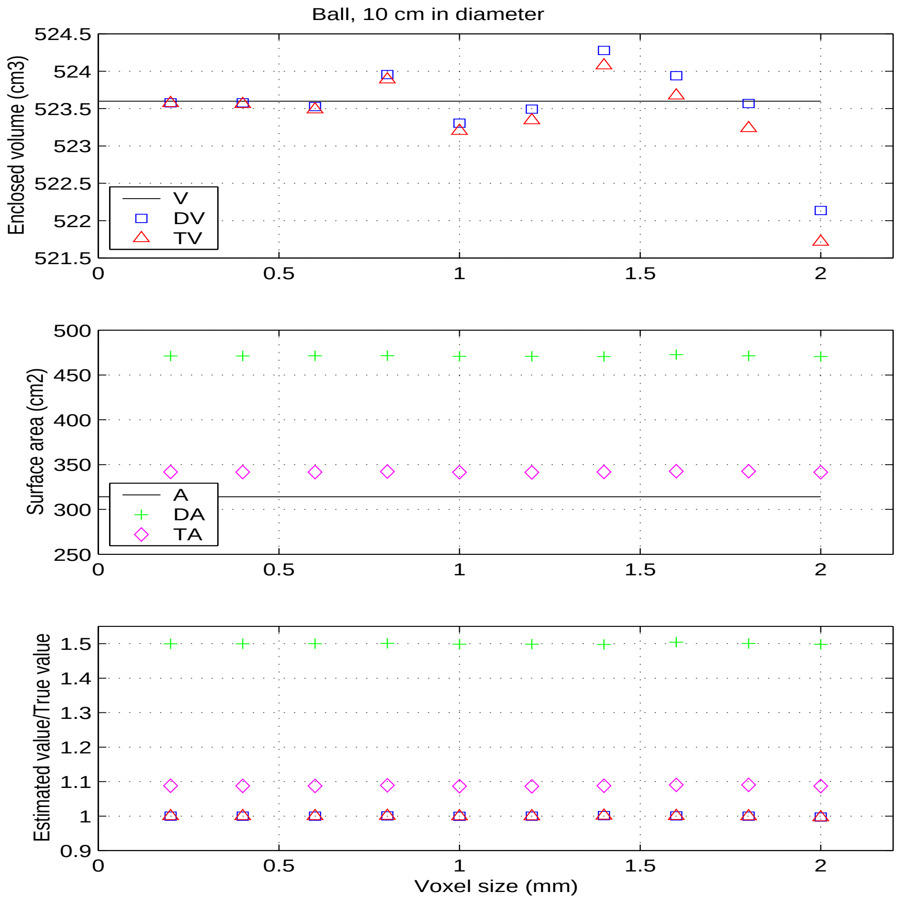


Figure 5. Volume estimates and area estimates are plotted as functions of the voxel size for digital surfaces and triangulated surfaces of a ball.

4.2. Physical Phantoms

The physical phantoms we have used in this work are five dry bone specimens of the talus, i.e., one of the bones in the area of the hindfoot. Their surfaces were sealed with methacrylate and varnish. The true bone volumes were then determined experimentally from water displacement measurements. The water displacement box is filled with water until it flows from the spout coming out from it. The bones were then put inside one by one, and the runoff was collected and measured.

Before scanning via MRI, the talus bones were suspended in the center of plastic boxes. Thereafter, the boxes were filled with the contrast medium Omniscan^{TM} (a gadodiamide solution) diluted 1:1000 in water. The phantoms were scanned twice (in different orientations) at resolution 256×256, where the images obtain a voxel size of $0.31 \times 0.31 \times 1.20$ mm^3. Three of the bones were also scanned twice (in different orientations) at resolution 512×512, where the images obtain a voxel size of $0.16 \times 0.16 \times 1.00$ mm^3.

The bones were thereafter segmented by using the interactive live-wire method.[23] Digital and triangulated measurements of the segmented bones are presented in Table 1. It can be noted that the digital volumes are larger than the measured volumes, still being rather close to these values. The triangulated volumes are a little smaller than the digital volumes, and hence are closer to the measured volumes. The triangulated area measurements are never close to the digital area measurements, always smaller, but this is expected as a summation of voxel faces results in an over-estimate.

Table 1. Talus bones in boxes of contrast medium. The true volumes are known by measuring water displacement. Volumes are in milliliters. Areas are in square centimeters.

Talus	**Enclosed volume** in milliliters (cm^3)			**Surface area** in cm^2		Image size	Voxel size
	Measured	Digital	Triangulated	Digital	Triangulated	in no. of voxels	in mm^3
No. 1	37.3	39.0	38.4	109.2	89.3	$256 \times 256 \times 47$	$0.31 \times 0.31 \times 1.20$
		39.0	38.4	106.7	87.4	$256 \times 256 \times 46$	
No. 2	29.1	30.2	29.9	92.6	79.2	$512 \times 512 \times 52$	$0.16 \times 0.16 \times 1.00$
		29.9	29.6	92.1	79.4	$512 \times 512 \times 47$	
		29.6	29.0	91.7	74.1	$256 \times 256 \times 46$	$0.31 \times 0.31 \times 1.20$
		29.6	29.1	89.4	73.8	$256 \times 256 \times 41$	
No. 3	25.1	25.4	25.2	81.9	70.5	$512 \times 512 \times 43$	$0.16 \times 0.16 \times 1.00$
		25.4	25.1	83.0	71.6	$512 \times 512 \times 55$	
		25.1	24.7	80.1	65.0	$256 \times 256 \times 44$	$0.31 \times 0.31 \times 1.20$
		25.1	24.7	80.1	65.2	$256 \times 256 \times 45$	
No. 4	23.6	23.8	23.6	80.4	69.0	$512 \times 512 \times 47$	$0.16 \times 0.16 \times 1.00$
		23.9	23.6	81.5	70.2	$512 \times 512 \times 46$	
		23.9	23.5	79.5	65.1	$256 \times 256 \times 41$	$0.31 \times 0.31 \times 1.20$
		23.7	23.3	80.6	66.1	$256 \times 256 \times 42$	
No. 5	35.2	35.7	35.2	103.7	85.5	$256 \times 256 \times 39$	$0.31 \times 0.31 \times 1.20$
		35.7	35.1	107.4	87.5	$256 \times 256 \times 38$	

4.3. Real Objects

We have tested our methods on real objects via clinical CT and MRI data sets obtained for various parts of the human anatomy, e.g., head, skull, knee, and blood vessels. Surfaces were created from segmented scene data obtained by applying a simple gray-level threshold specifically tailored to the respective data set. We chose eight objects of different sizes and shapes, as described in Table 2. The triangulated volume estimates follow the digital volume estimates closely, while the triangulated area estimates are not as close, but this deviation can be expected, as described above.

Table 2. Real objects imaged at different resolutions. Volumes are in cubic centimeters. Areas are in square centimeters.

Data sets	**Enclosed volume** in cm^3		**Surface area** in cm^2		Image size	Voxel size
	Digital	Triangulated	Digital	Triangulated	in no. of voxels	in mm
CT head, skin	2416.4	2440.3	2436.5	2094.6	$512 \times 512 \times 97$	$0.49 \times 0.49 \times 1.50$
CT head, skull	396.5	404.4	3013.3	2555.7	$512 \times 512 \times 97$	$0.49 \times 0.49 \times 1.50$
CT head, skull	142.8	147.8	1193.4	991.5	$512 \times 512 \times 32$	$0.49 \times 0.49 \times 1.50$
MR head, skin	2616.4	2607.9	4214.1	4038.3	$256 \times 256 \times 53$	$0.86 \times 0.86 \times 3.00$
CT child skull	314.4	313.8	2710.1	2132.7	$351 \times 465 \times 145$	$0.41 \times 0.41 \times 1.00$
CT dry skull	553.6	555.8	2796.0	2316.0	$193 \times 242 \times 68$	$0.80 \times 0.80 \times 3.00$
CT knee	92.3	91.4	1077.1	1089.4	$171 \times 193 \times 69$	$0.68 \times 0.68 \times 1.00$
MRA vessels	80.5	82.5	581.1	535.5	$256 \times 256 \times 128$	$1.09 \times 1.09 \times 2.20$

5. CONCLUDING REMARKS

It is known that triangulated surface representations give a more accurate area estimate for the original object surface than digital surface representations. In this paper, we have demonstrated that volume enclosed by triangulated surfaces can be computed efficiently in the same elegant way the volume enclosed by digital surfaces is computed by digital surface integration. We have shown that our triangulated shell data structure retains advantages and overcomes difficulties of both the digital and the triangulated approaches.

Because of the way the surface is scanned and the normal components are utilized, not only solid objects can be handled, but also object cavities are dealt with correctly. Cavities are identified by having a negative volume. It is also possible to obtain measurements from several objects in an image in one scan. For this purpose, the described method is very efficient compared to, for example, connected component labelling, which is more costly than surface tracking. See Ref. 22.

In this work, the vertices of the surface triangles have been positioned midway between adjacent voxels. With a more sophisticated interpolation technique, the triangles would be closer to the true surface and the estimates would likely become more accurate. The complexity of the method would increase, however, since precomputed lookup tables may not be feasible to the same extent. Note that, by discretizing the possible locations of the triangle vertices along the m-cube edges, the lookup table idea becomes feasible, although the size of the table will increase. By using even the "hard" (midpoint) triangles, an improvement has been achieved compared to using the voxel faces as was shown in Figures 4 and 5. These figures also indicate a trend that the digital volume approaches the triangulated volume, which in turn approaches the true volume, with decreasing voxel size.

The need to generate the marching cube configuration code while generating digital surfaces leads to future work. A study of the relationship among digital surfaces, marching cubes representation, and T-shell is worth pursuing. Another future study is to verify that our ideas from projective geometry can be utilized to compute the volume enclosed by any polygonal surface.

ACKNOWLEDGMENTS

We express our thanks to Dewey Odhner at MIPG for help in understanding the "secrets of bits" of 3DVIEWNIX and to Joakim Lindblad at Centre for Image Analysis for help with the mathematical phantoms. This work was initiated during the first author's visit to MIPG during February–May, 2001. The support of the following DHHS grants is acknowledged: AR46902 and NS37172.

REFERENCES

1. J. K. Udupa and G. T. Herman, eds., *3D Imaging in Medicine*, ch. 1. CRC Press LLC, Boca Raton, FL, 2nd ed., 2000.
2. J. K. Udupa, "Determination of 3-D shape parameters from boundary information," *Computer Graphics and Image Processing* **17**, pp. 52–59, 1981.
3. G. Wyvill, C. McPheeters, and B. Wyvill, "Data structures for soft objects," *The Visual Computer* **2**(7), pp. 227–234, 1986.
4. W. E. Lorensen and H. E. Cline, "Marching Cubes: A high resolution 3D surface construction algorithm," in *Proceedings of the 14th ACM SIGGRAPH on Computer Graphics*, **21(4)**, pp. 163–169, July 1987.
5. J.-O. Lachaud and A. Montanvert, "Digital surfaces as a basis for building iso-surfaces," in *Proc. of 5th IEEE Int. Conference on Image Processing (ICIP'98)*, **2**, pp. 977–981, (Chicago, IL), 1998.
6. J.-O. Lachaud and A. Montanvert, "Continuous analogs of digital boundaries: A topological approach to iso-surfaces," *Graphical Models* **62**, pp. 129–164, 2000.
7. D. Gordon and J. K. Udupa, "Fast surface tracking in three-dimensional binary images," *Computer Vision, Graphics, and Image Processing* **45**, pp. 196–214, Feb. 1989.
8. J. K. Udupa and D. Odhner, "Fast visualization, manipulation, and analysis of binary volumetric objects," *IEEE Computer Graphics and Applications* **11**, pp. 53–62, Nov. 1991.
9. G. J. Grevera, J. K. Udupa, and D. Odhner, "An order of magnitude faster isosurface rendering in software on a PC than using dedicated, general purpose rendering hardware," *IEEE Transactions on Visualization and Computer Graphics* **6**, pp. 335–345, Oct. 2000.
10. J. K. Udupa, "Multidimensional digital boundaries," *Graphical Models and Image Processing* **56**, pp. 311–323, July 1994.
11. G. J. Grevera, J. K. Udupa, and D. Odhner, "T-shell rendering," in *Medical Imaging 2001: Visualization, Display, and Image-Guided Procedures*, S. K. Mun, ed., pp. 413–425, Proc. SPIE 4319, 2001.
12. J. K. Udupa and D. Odhner, "Shell rendering," *IEEE Computer Graphics and Applications* **13**, pp. 58–67, Nov. 1993.
13. A. Van Gelder and J. Wilhelms, "Topological considerations in isosurface generation," *ACM Transactions on Graphics* **13**(4), pp. 337–375, 1994.
14. M. J. Dürst, "Letters: Additional reference to "Marching Cubes"," in *Proceedings of ACM SIGGRAPH on Computer Graphics*, **22(2)**, pp. 72–73, Apr. 1988.
15. G. M. Nielsen and B. Hamman, "The asymptotic decider: Resolving the ambiguity in marching cubes," in *Proceedings of IEEE Visualization'91*, pp. 83–90, IEEE Computer Society Press, Oct. 1991.
16. Y. Kenmochi, K. Kotani, and A. Imiya, "Marching cubes method with connectivity," in *Proc. of IEEE Int. Conference on Image Processing (ICIP'99)*, pp. 361–365, 1999.
17. Y. Kenmochi and R. Klette, "Surface area estimation for digitized regular solids," in *Vision Geometry IX*, L. J. Latecki, R. A. Melter, D. M. Mount, and A. Y. Wu, eds., pp. 100–111, Proc. SPIE 4117, 2000.
18. J. Lindblad and I. Nyström, "Surface area estimation of digitized 3D objects using local computations," in *Proceedings of Discrete Geometry for Computer Imagery (DGCI 2002)*, A. Braquelaire, J.-O. Lachaud, and A. Vialard, eds., *Lecture Notes in Computer Science*, Springer-Verlag, 2002. Accepted for publication.
19. F. C. A. Groen and P. W. Verbeek, "Freeman code probabilities of object boundary quantized contours," *Computer Graphics and Image Processing* **7**, pp. 391–402, 1978.
20. J. Lindblad, "Perimeter and area estimates for digitized objects," in *Proceedings of SSAB (Swedish Society for Automated Image Analysis) Symposium on Image Analysis*, pp. 113–117, (Norrköping, Sweden), 2001.
21. S. W. Hughes, T. J. D'Arcy, D. J. Maxwell, J. E. Saunders, C. F. Ruff, W. S. C. Chiu, and R. J. Sheppard, "Application of a new discreet form of Gauss' theorem for measuring volume," *Physics in Medicine and Biology* **41**, pp. 1809–1821, Sept. 1996.
22. J. K. Udupa and V. G. Ajjanagadde, "Boundary and object labelling in three-dimensional images," *Computer Vision, Graphics, and Image Processing* **51**, pp. 355–369, Sept. 1990.
23. A. X. Falcão, J. K. Udupa, S. Samarasekera, S. Sharma, B. E. Hirsch, and R. A. Lotufo, "User-steered image segmentation paradigms: Live wire and live lane," *Graphical Models and Image Processing* **60**, pp. 233–260, July 1998.