

Conversion of a Surface Model of a Structure of Interest into a Volume Model for Medical Image Retrieval

Sarmad ISTEPHAN*, Mohammad-Reza SIADAT

Oakland University, Department of Computer Science and Engineering, 2200 N Squirrel Rd Rochester, MI 48309, USA.

E-mails: saistep3@oakland.edu; siadat@oakland.edu

* Author to whom correspondence should be addressed; Tel.: 7346348755

Received: 19 April 2015 / Accepted: 12 June 2015 / Published online: 23 June 2015

Abstract

Volumetric medical image datasets contain vital information for noninvasive diagnosis, treatment planning and prognosis. However, direct and unlimited query of such datasets is hindered due to the unstructured nature of the imaging data. This study is a step towards the unlimited query of medical image datasets by focusing on specific Structures of Interest (SOI). A requirement in achieving this objective is having both the surface and volume models of the SOI. However, typically, only the surface model is available. Therefore, this study focuses on creating a fast method to convert a surface model to a volume model. Three methods (1D, 2D and 3D) are proposed and evaluated using simulated and real data of Deep Perisylvian Area (DPSA) within the human brain. The 1D method takes 80 msec for DPSA model; about 4 times faster than 2D method and 7.4 fold faster than 3D method, with over 97% accuracy. The proposed 1D method is feasible for surface to volume conversion in computer aided diagnosis, treatment planning and prognosis systems containing large amounts of unstructured medical images.

Keywords: Medical image analysis; Medical image querying; Medical image database; Medical informatics; Content-based image retrieval

Introduction

There has been and continues to be an enormous amount of medical image data acquisitions from minimally invasive examinations. Image modalities include brain Magnetic Resonance Imaging (MRI), Computed Tomography (CT), Positron Emission Tomography (PET), Single-Photon Emission Computed Tomography (SPECT), function MRI (fMRI), Diffusion Tensor Imaging (DTI) [1-3]. In fact, five billion medical images have been acquired up to 2010 worldwide [4]. According to the Organisation for Economic Co-operation and Development (OECD), there were 97.7 MRI exams and 265 CT exams performed per 1,000 population in the United States in 2010 [5]. This tremendous number of exams results in tens of terabytes of medical image data storage in an average size radiology department per year and hence petabytes of data storage across all radiology departments [6].

Medical imaging data are typically stored in archival systems such as Picture Archiving and Communication System (PACS). PACS systems are excellent at storing the images. Furthermore, PACS systems retrieve the unstructured medical images using structured data such as the imaging acquisition information [7-9]. However, they lack the ability to retrieve the unstructured medical images using the contents of the unstructured image data directly such as the volume of a brain [10, 11].

Quantitative analysis of the unstructured image data stored in PACS systems is extremely important in noninvasive diagnosis, treatment planning and prognosis [12]. For instance, volumetric and texture analysis of brain MRI can be used to lateralize and localize epileptogenic hippocampi in temporal lobe epilepsy [12, 13]. This task is not only challenging for PACS systems, but it is also challenging for human experts who tend to be more qualitative than quantitative. Moreover, the vast volume of data makes such a task beyond human capabilities in many cases [14]. Therefore, there is a need for a system to automatically query and retrieve images using unstructured image data.

In the current literature, there are two major types of methods used to query unstructured image data. The first method is text-based, which was originally developed in the 1970s [15]. This method uses the images' structured data (i.e., image annotations) for querying and retrieving images. Systems that rely on text-based searches include iBase, Index+, Digital Catalogue, Fastfoto Picdar, FotoWare and Signpost [16]. Furthermore, annotations may not be specific enough to the location in the image being referenced [16].

In the medical domain, text-based methods are executed on image annotations as dictated by a radiologist. This method has several shortcomings. Annotations may be biased due to the radiologist's mindset at the time of annotating [11]. In addition, annotations are narrow scoped and do not lend themselves to future querying with different criteria to obtain additional insights on the image which is due to the abundance and complexity of image data [13, 6]. Furthermore, annotations may not be specific enough to the location in the image being referenced [16]. Finally, it is difficult to obtain the semantics of the images from the basic annotations [17].

The second method used for querying image data is Content Based Image Retrieval (CBIR), which was originally developed the 1980s. CBIR overcomes the shortcomings of a text-based system because it operates on the unstructured image data directly. Some systems that rely on CBIR include QBIC, Virage and Blobworld [18, 19]. CBIR consists of three levels [15]. First level operates on low-level features such as color, texture, and shape. The values from these features are used to perform image retrieval (e.g. [20]). The second level operates on the semantics of the image such as recognizing the overall scene and the objects within it. The third level also leverages the semantics of the images but at a much higher level in order to answer queries such as determining the mood of the picture [7, 14, 15].

In this paper, we are interested in the first level of CBIR methods that operate on low-level shape features. Shape features are placed in two categories, global features, and local features. Global features describe overall features of the image such as aspect ratio. Local features describe the shape of a specific Structure of Interest (SOI) within the image, which is the focus of this paper [21].

An SOI is used to query local low-level features instead of the entire image [22]. In a medical application, a SOI can be a physiological or anatomical structure segmented in a medical image modality (e.g. T1-weighted MRI). The result of the segmentation can be a set of 2D binary image slices that it will be referred to as the *volume model*. Furthermore, the volume model can be transformed into the outer 3D surface model of the SOI that we refer to as the *surface model*.

In order to extract the most information (e.g. features) from a given SOI, both the volume model and the surface model must be available. This is because some features are extracted easier using one model representation than the other. Unfortunately, both models are not always present.

The volume model representation of the medical SOI is needed since some features are retrieved easier with it. For example, the volume model can be used to query human brain changes, which happens due to a variety of reasons such as age and disease [22]. Specifically, as a person ages, the volumes of grey and white matter, the size of some brain structures and the size of the brain are reduced. In order to measure the reduction, a brain volume model must be used [23]. The volume model makes it possible to apply methods such as cross-sectional analysis of deformations, cross-sectional analysis of tissue concentrations, longitudinal analysis of boundary shift, pattern classification methods, and cortical pattern matching, to identify brain changes [24]. Furthermore, in order to visualize the volume model, direct volume rendering techniques can be used [25] and ray-leaping techniques can be used [26]. Therefore, the volume model is needed for a subset of queries against the medical SOI.

A more concrete example to solidify the link between theory and practice for using the volume model is given here. Suppose in a clinical data repository, we have conventional MRI and DTI images of hundreds of patients with temporal lobe epilepsy where the hippocampi are already segmented on pre-op conventional MRI and their surface models are available. Given this data repository we would like to know whether the average Fractional Anisotropy (FA) within the white matter adjacent to hippocampus has any indication regarding epileptogenicity. This can potentially help in diagnosis and prognosis as to whether the hippocampal resection will be an effective treatment plan [27-29]. Using morphological image processing would provide an efficient method to generate the adjacent models. This method can only be applied to the volume model and not the surface model. Moreover, using surface model based methods cannot provide as efficient approach as given by morphology. In this scenario, having the volume model is necessary and therefore having the capability of converting the surface models to the volume models would be essential. Note that one may derive many different variations of the example given above (e.g. by looking into different brain structures such as the Deep Perisylvian Area or different diffusion maps) that show the link between theory and practice for the proposed work.

The surface model representation of the SOI is also needed since some features are more easily retrieved with the help of it. For example, the surface model is used to perform shape and curvature analyses [24, 30], which can then be used to identify hippocampal abnormalities [13, 31-34]. In addition, the surface model enhances the ability to simplify segmentation, label the various structures, and reduce the effect of noise [35]. Furthermore, the surface model can be visualized using surface rendering techniques [25]. Therefore, the surface model is needed for a subset of queries against the medical SOI.

Motivation

As noted above, both the surface model and the volume model are needed to fully query medical image data. As a result, conversion methods between the surface and volume models are important. Constructing a surface model from a volume model is most commonly accomplished using Marching Cubes [36-38]. In this paper, we propose and develop the reverse conversion from a surface model to a volume model that is starred in Figure 1. There are some practical advantages for such a conversion. For example, the storage of surface models uses less space than that of volume models, which is especially true in older archived images. In addition, surface models can be under-sampled or over-sampled when converting to a volume model to fulfill the given task.

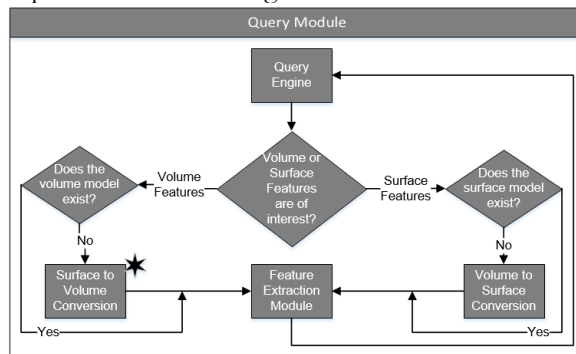


Figure 1. Overall system for querying an unstructured medical Structure of Interest. The contribution of this paper is the Surface to Volume Conversion method

Surface to Volume Methods Background

Surface model to volume model conversion entails finding all the voxels inside the surface model. In the literature, this problem is reduced to classifying voxels as inside or outside a 2D polygon. However, in the medical field we deal with 3D surface models so this approach is not sufficient. Therefore, the 3D surface model must first be cut into a set of 2D polygons whose voxels are then classified. The general approach to accomplish the classification task is ray casting. Ray casting literature can be grouped into two categories:

- a) Methods with preprocessing
- Jimenez et al. propose a preprocessing method of tri-trees followed by two different classification tests including ray casting and triangle based [39].
 - Zalik et al. propose a preprocessing method of a grid of cells followed by a classification test based on a modified flood-fill method [40].
- b) Methods without preprocessing
- Ye et al cast a ray through the test point and classify it by using a substitution method [41].
 - Jian et al cast a ray from the test point and classify the test point based on the sum of a position function between the ray and the edges [42].
 - Siadat et al use traditional ray casting with coagulation effect within their database management system [13].

As summarized above, ray casting is a well-known approach for classifying a point as inside or outside of a 2D polygon. Since this work is related to the medical field, which has 3D surface models, we extended the 2D ray casting to be done in 3D on the surface model directly, implemented the 2D method, and developed a novel approach for 1D. The development of the 1D method was inspired by our observation of how the 3D method can be constrained into a 2D method.

Method

A surface model is a closed surface made out of triangle patches. A volume model consists of all the voxels inside the surface model. Therefore, to convert a surface model to a volume mode, all the voxels inside the surface model must be found. To do this, a 3D method has been proposed which is an extension of traditional 2D ray casting; also, the traditional 2D ray casting has been mathematically formulated; finally, a novel 1D method has been proposed. Also, please note that to convert the other way, a volume model to a surface model, then a marching cubes can be used as depicted in Figure 2. The rest of the method section will explain the surface model to volume model 3D, 2D and 1D conversion methods, respectively.

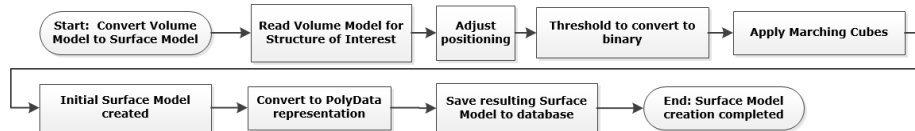


Figure 2. Steps of converting a volume model to a surface model

3D Method

At a high level, this method commences by casting 3D rays from every test voxel within the 3D space and classifying the voxel inside or outside the surface model based on the number of intersections the rays have with the surface model's triangle patches. The high level steps are as below followed by a detailed explanation of each step.

1. Calculate a plane for each triangle patch.
2. Determine the bounding box of the surface model.
3. For each voxel in the bounding box, cast a set of rays.
4. Intersect each ray with all the triangle patches' planes.
5. Count the number of intersections that each ray encountered.
6. Using a formula, classify the voxel as inside or outside the surface model.
7. Create a volume model using all the voxels classified as inside the surface model.

Step 1, for each triangle patch $T_i, i \in \{1, \dots, b\}$, a plane P_T is calculated using the plane equation (Eq. 1). Without loss of generality, d' is set to -1, which simplifies Eq. 1 into Eq. 2 by substitution. Parameters a , b , and c comprise a 3D normal vector n of P_T for each T_i . Since T_i 's three vertices are

on its corresponding P_T , Eq. 2 is transformed into a linear system of equations as in Eq. 3 where matrix M contains the vertices' coordinates and the vector $1 = [1 \ 1 \ 1]$ represents the location of P_T .

$$P_T: a'x + b'y + c'z + d' = 0 \tag{1}$$

$$P_T: ax + by + cz = 1 \tag{2}$$

$$P_T: M^{-1} 1 = n \tag{3}$$

Step 2 is to create a 3D bounding box around the surface model to reduce the number of voxels that need to be classified. The 3D bounding box is calculated by finding the minimum and maximum values of x, y and z of the surface model and enlarging it by a few voxels so that it is not touching the surface model.

Step 3 casts a set of rays, each denoted by R , from every test voxel to be classified as inside within the 3D bounding box, denoted by $P_b(x_b, y_b, z_b)$ to a neighboring structure's voxels, denoted by $P_{bni}(x_{bni}, y_{bni}, z_{bni})$, using Eq. 4. An example of a neighboring structure is the set of 26 points that neighbor $P_b(x_b, y_b, z_b)$ formed by various combinations of x_b , y_b and z_b incremented by 1 and decremented by -1 as listed in Table 1. Using Table 1, a unique set of values to increment/decrement from each column is selected (e.g. see bolded line segments in Table 1) because intersections are counted on one side of the voxel only. Note that more than one ray is used because there are extreme cases in which casting only one ray may result in an inaccurate classification.

$$R: (x-x_b)/(x_{bni}-x_b) = (y-y_b)/(y_{bni}-y_b) = (z-z_b)/(z_{bni}-z_b) \tag{4}$$

Table 1. Showing a set of 26 possible neighboring points to point (0,0,0)

1, -1, 1	0, -1, 1	-1, -1, 1	-1, 0, 1	-1, 1, 1	0, 1, 1	1, 1, 1	1, 0, 1	0, 0, 1	1, -1, 0	0, -1, 0	-1, -1, 0	-1, 0, 0
-1, 1, -1	0, 1, -1	1, 1, -1	1, 0, -1	1, -1, -1	0, -1, -1	-1, -1, -1	-1, 0, -1	0, 0, -1	-1, 1, 0	0, 1, 0	1, 1, 0	1, 0, 0

Step 4 finds all the intersection points $P_k(x_k, y_k, z_k)$ of R and all the planes P_T using equation set 1. This equation set is derived by plugging the line equation (Eq. 4) into the plane equation (Eq. 3) from the perspective of x .

Equation Set 1:

$$x_k = -\frac{d - b\left(y_b - \frac{y_{bni}-y_b}{x_{bni}-x_b} x_b\right) - c\left(z_b - \frac{z_{bni}-z_b}{x_{bni}-x_b} x_b\right)}{\left(a + b\frac{y_{bni}-y_b}{x_{bni}-x_b} + c\frac{z_{bni}-z_b}{x_{bni}-x_b}\right)} = -\frac{(d - y_b - z_b)(x_{bni}-x_b) - b(y_{bni}-y_b)x_b - c(z_{bni}-z_b)x_b}{a(x_{bni}-x_b) + (b(y_{bni}-y_b) + c(z_{bni}-z_b))}$$

$$y_k = \frac{y_{bni}-y_b}{x_{bni}-x_b} x_k + \left(y_b - \frac{y_{bni}-y_b}{x_{bni}-x_b} x_b\right)$$

$$z_k = \frac{z_{bni}-z_b}{x_{bni}-x_b} x_k + \left(z_b - \frac{z_{bni}-z_b}{x_{bni}-x_b} x_b\right)$$

It is important to note that if the end points (P_b and P_{bni}) of the ray casted have the same values for x then the intersections will not be defined using equation set 1 due to dividing by zero error. Therefore, equation set 2 is needed which is derived in the same manner as equation set 1 but from the perspective of y .

Equation Set 2:

$$x_k = \frac{x-x_b}{y-y_b} (x_{bni} - x_b) + x_{bni}$$

$$y_k = -\frac{d + cz_b + ax_b - c\frac{z_{bni}-z_b}{y_{bni}-y_b} y_b - a\frac{x_{bni}-x_b}{y_{bni}-y_b} y_b}{b + c\frac{z_{bni}-z_b}{y_{bni}-y_b} + a\frac{x_{bni}-x_b}{y_{bni}-y_b}} = -\frac{(d + cz_b + ax_b)(y_{bni}-y_b) - c(z_{bni}-z_b)y_b - a(x_{bni}-x_b)y_b}{b(y_{bni}-y_b) + c(z_{bni}-z_b) + a(x_{bni}-x_b)}$$

$$z_k = z_b$$

A similar problem will occur as noted above if the endpoints of the ray have the same values of y since division by zero is not allowed. Therefore, equation set 3 is needed which is derived in the same manner as equation set 2 but from the perspective of z .

Equation Set 3:

$$x_k = x_b$$

$$y_k = [(y_{bni}-y_b)/(z-z_b)](x-z_b) + y_b$$

$$Z_k = - \frac{d+ax_b+by_b-a \frac{x_{bni}-x_b}{z_{bni}-z_b} z_b - b \frac{y_{bni}-y_b}{z_{bni}-z_b} z_b}{c+a \frac{x_{bni}-x_b}{z_{bni}-z_b} + b \frac{y_{bni}-y_b}{z_{bni}-z_b}} = - \frac{(d+ax_b+by_b)(z_{bni}-z_b) - a(x_{bni}-x_b)z_b - b(y_{bni}-y_b)z_b}{c(z_{bni}-z_b) + a(x_{bni}-x_b) + b(y_{bni}-y_b)}$$

To summarize the choice of how to select an equation for calculating the intersection, Table 2 is created. It lists all eight cases that can exist on whether the values of x, y or z of R's endpoints are equal. For each case, the equation set(s) that are appropriate to use are listed.

Table 2. List of equation sets that may be selected based on coordinates of P_b and P_{bni}

Case #	$X_1=X_2$	$Y_1=Y_2$	$Z_1=Z_2$	Equation set to use
1	false	false	false	1 or 2 or 3
2	false	false	true	1 or 2
3	false	true	false	1 or 3
4	false	true	true	1
5	true	false	false	2 or 3
6	true	false	true	2
7	true	true	false	3
8	true	true	true	Not possible

There is one exception where all three equations sets fail, which happens when P_T is parallel to R. In this case, either there is no intersecting point since P_T and R are parallel or there are an infinite number of intersections because R is on P_T . No intersection points can be found in either case.

Given the intersection point $P_k(x_k, y_k, z_k)$ calculated above, two checks must take place. One is that only one side of the ray going through $P_b(x_b, y_b, z_b)$ is used to determine intersecting voxels. The second check is to ensure that $P_k(x_k, y_k, z_k)$ is inside the triangle patch whose plane intersected the ray.

The second check is done as follows. Given triangle patch T_i with vertices (A, B, C) and line segments \overline{AB} , \overline{AC} and \overline{BC} as in Figure 3, $P_k(x_k, y_k, z_k)$ is inside T_i if it passes the following three conditions.

- I) The point is on the same side of segment AB as that of vertex C,
- II) the point is on the same side of segment AC as that of vertex B, and
- III) The point is on the same side of segment BC as that of vertex A.

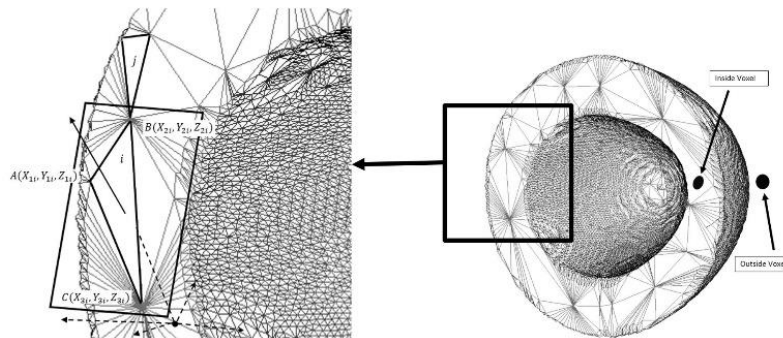


Figure 3. Figure on the left is an enlarged portion of the surface model on the right. The image on the left shows a set of rays being casted from a voxel and intersected with the plane of a triangle patch in order to classify the voxel as inside or outside the surface model.

The three conditions are mathematically satisfied by the corresponding equations below followed by the dot product of each condition's result given by $D = \vec{S}_1 \cdot \vec{S}_2$. When the dot product is greater than or equal to zero, then \vec{S}_1 and \vec{S}_2 are pointing in the same direction which means the point is inside the triangle, otherwise, it is outside.

Condition 1 Equation:

$$\vec{S}_1 = \vec{AB} \times \vec{AP}_k(x_k, y_k, z_k)$$

$$\vec{S}_2 = \vec{AB} \times \vec{AC}$$

Condition 2 Equation:

$$\vec{S}_1 = \vec{AC} \times \vec{AP}_k(x_k, y_k, z_k)$$

$$\vec{S}_2 = \vec{AC} \times \vec{AB}$$

Condition 3 Equation:

$$\vec{S}_1 = \vec{BC} \times \vec{BP}_k(x_k, y_k, z_k)$$

$$\vec{S}_2 = \vec{BC} \times \vec{BA}$$

Step 5 accumulates the number of intersections that each ray had. Step 6 classifies a voxel as inside or outside the surface model using Eq. 5 [11], which takes in the number of rays R_n , and the number of intersections for each ray I_n to calculate a score for $P_b(x_b, y_b, z_b)$, which is used to classify it as inside or outside the surface model when the score is 0 or 1, respectively [11].

$$Score = Round\left(\frac{1}{R_n} \sum_{i=1}^{R_n} \frac{1+(-1)^i}{2}\right) \quad (4)$$

Finally, in step 7, all voxels classified as inside are assigned a nonzero intensity (e.g. 100) and all outside voxels are given an intensity of 0 creating a volume model.

3D Computational Complexity Analysis

The above method iterates through all the bounding box's voxels ($m*n*r$) and intersects the casting rays with all T_i triangle patches of the surface model. Therefore the complexity is $O(m*n*r*T_i)$. As a side note, methods in other literature look at only $O(T_i)$ because their goal is to classify one test voxel and not to find all the inside voxels. An example of the 3D method's complexity would be, given a bounding box with dimensions $90*90*102 = 826,200$ and a surface model with 52,000 patches then the required number of operations is approximately 43 billion operations. On the laptop we tested on, calculating an intersection with all T_i for each voxel takes approximately 500 milliseconds. This means that for the given 826,200 operations, calculating an intersection will take $826,200*500 = 413,100,000$ milliseconds. Therefore, using the 3D method to convert the example surface model to a volume model can take up to 5 days (~115 hours).

2D Method

The complexity computed in the previous section for the 3D method is high. The number of triangle patches T_i is usually huge. The problem is that there is no way to reduce T_i except using decimation. However, even after performing such a step, in a reasonable way, it will still result in T_i being large for a fairly complicated real surface model. A feasible way to lower this complexity is to reduce the number of times the triangle patches are visited (the number of times a ray is intersected with all triangle patches). This is the premise for the 2D method.

A good way to reduce the computational complexity is to reduce the space from a 3D space to a 2D space prior to performing ray casting. To do so, the model is cut by planes in an arbitrary direction and at an arbitrary sampling rate resulting in 2D contours (i.e. polygon). Each voxel in the 2D contour space is now classified as inside the contour, and hence inside the surface model, by casting a ray and counting the number of intersections with the edges of the contour which is a significantly less than the number T_i . Therefore, the 2D method could result in significantly less complexity.

At a high level, this method commences by cutting the 3D surface model resulting in a set of 2D contours. Rays are then casted from every test voxel within the 2D contour's space classifying the voxel as inside or outside the contour, hence inside or outside the surface model, based on the number of intersections the rays have with the contour's edges. The high level steps are as below followed by a detailed explanation of each step.

1. Create a 3D bounding box for the surface model.
2. Select a direction and a sampling rate for that direction to cut the surface model.

3. Intersect the surface model by parallel plane, P_i at each sampling point forming 2D contour(s).
4. Calculate a 2D bounding box for each 2D contour.
5. For each voxel on the cutting plane P_i cast a set of 2D rays with random slopes and count the number of intersections.
6. Steps 6 and 7 from the 3D method and then used to create the final volume model.

Step 1, a 3D bounding box for the surface model is created in the same manner as the 3D method. Step 2, the 3D bounding box is sampled in one direction. Step 3, at each sampling step, a plane P_i , $i \in \{1, \dots, q\}$, is intersected with the surface model, as illustrated in Figure 4. In other words, P_i is intersected with all T_i . The intersection process occurs against the three line segments of T_i using equation set 4. The intersection points are then connected forming a 2D contour (see Figure 5 for an example).

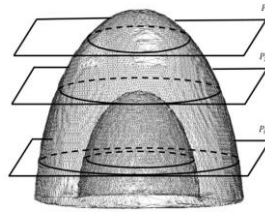


Figure 4. Plane slicing - Surface Model M_s being sliced by plane P_i



Figure 5. A contour from a surface model

Equation Set 4:

$$x = - \frac{\left(d + by_i + cz_i - b \frac{y_j - y_i}{x_j - x_i} x_i - c \frac{z_j - z_i}{x_j - x_i} x_i \right)}{a + b \frac{y_j - y_i}{x_j - x_i} + c \frac{z_j - z_i}{x_j - x_i}} = \frac{(d + by_i + cz_i)(x_j - x_i) - b(y_j - y_i)x_i - c(z_j - z_i)x_i}{a(x_j - x_i) + b(y_j - y_i) + c(z_j - z_i)}$$

$$y = y_i + [(y_j - y_i) / (x_j - x_i)](x - x_i)$$

$$z = z_i + [(z_j - z_i) / (x_j - x_i)](x - x_i)$$

The equation has the same exceptions as that of the 3D method and hence similar equations can be derived using the same manner of that in the 3D method with the perspectives of y and z . Hence, the same equation set selection process needs to be used here as that one in the 3D method when performing an intersection.

Step 4, a 2D bounding box is computed for each contour. While iterating through the 2D bounding box, step 5 classifies each voxel as inside or outside the contour. This is done by casting a set of rays, with random slopes m , from each point in the bounding box, $P_b(x_b, y_b)$ and then intersected with contour's edges. Let each contour edge have two endpoints edge start (X_{es}, Y_{es}) and edge end (X_{ee}, Y_{ee}) . To test whether there is an intersection voxel $P_k(x_k, y_k)$, equation set 5 is used.

Equation Set 5:

$$x = [(m * X_{poi} - Y_{poi} + Y_{es})(X_{ee} - X_{es}) - X_{es}(Y_{ee} - Y_{es})] / [m(X_{ee} - X_{es}) - (Y_{ee} - Y_{es})]$$

$$y = Y_{poi} + m * (x - X_{poi})$$

A few checks must take place to ensure that the P_k can be counted as an intersection voxel. First, P_k must be within the edge's endpoints (X_{es}, Y_{es}) and (X_{ee}, Y_{ee}) . The second test is to verify whether all the intersecting points are on one side of (X_{poi}, Y_{poi}) . This is because a ray could cause intersections on both sides of (X_{poi}, Y_{poi}) if for example (X_{poi}, Y_{poi}) is in the middle of the model. To do this, the dot product must be calculated. Given the first intersected point (X_{j1}, Y_{j1}) , and the new intersect point, (X_{ni}, Y_{ni}) , calculate the dot product using Equation 6.

$$dotProductValue = [(X_{j1} - X_{poi}) * (X_{ni} - X_{poi})] + [(Y_{j1} - Y_{poi}) * (Y_{ni} - Y_{poi})] \tag{6}$$

Using the same process as that of 3D with the result of the dot product, the intersection voxel is either counted or not. Finally, in step 6, using the same scoring method and volume model creation technique as that of the 3D method, the volume model is created.

Special Case when Cutting by Z

Typically in medical imaging, the surface model will be intersected with a special plane P_i , $i \in \{1, \dots, q\}$ which goes through the z -axis. In plane P_i , the value for z is set to a constant k and the values of a and b are 0. Therefore, the equation for P_i is simply $z=k$ and the intersection point can be calculated using Equation Set 6.

Equation Set 6:

$$\begin{aligned}
 P_i: z &= k \\
 x &= [(k-z_j) / (z_i-z_j)](x_j-x_i) + x_i \\
 y &= [(k-z_j) / (z_i-z_j)](y_j-y_i) + y_i
 \end{aligned}$$

2D Computational Complexity Analysis

The above method has two major steps. First is cutting the model (i.e. intersecting all triangle patches) by q number of planes, which has complexity $O(q * T_i)$. Note that q is significantly less than $m * n * r$, which was the number of times the triangle patches were visited in the 3D method. Using the example in the last part of the previous section and assuming $q=100$, this means $m * n * r / q \cong 8,100$ times reduction in visiting triangle patches.

Second is classifying a voxel as inside or outside within the 2D contour whose complexity is the number of edges. The number of edges e in a 2D contour is significantly lower than the number of triangle patches T_i since only a very small fraction of triangle patches intersect with each cutting plane. Therefore, the complexity of step 2 is $O(m * n * e)$. The overall complexity is $O(q * T_i) + O(m * n * e)$ which is much smaller than the 3D complexity of $O(m * n * r * T_i)$.

1D Method

The motivation of the 1D method is to create a volume model from a surface model with less computational complexity than that of the 2D method by reducing the number of edge visits for each 2D contour. Similar to how the number of triangle visits were significantly reduced in the previous approach, limiting the number of edge visits has promising potential. In the previous case, only the triangles intersecting a plane were considered, so in 1D we only visit edges intersecting a line. For the sake of simplicity, this line is chosen parallel to the y -axis as previously the plane was chosen to be parallel to the xy -plane. Analogous to the previous approach, the number of 2D rays casted would be significantly reduced using the 1D approach.

The 1D method works as follows. In the same manner as the 2D method, a set of 2D contours are created. Given the 2D contours, an arbitrary sampling direction (see the horizontal axis in Figure 6 for an example) and an arbitrary sampling rate are determined. A ray perpendicular to the sampling direction is casted at each sampling step, denoted by R_i with slope m_i and y -intercept b_i (see Equation Set 7). The ray R_i is then intersected with all the edges of the contour, where each edge is represented as R_m with slope m_m and y -intercept b_m . In order to calculate the x intersection, R and R_m are intersected resulting in equations for x and y (see equation set 7) which are the intersection point.

Equation Set 7:

$$\begin{aligned}
 R_i: y &= m_i x + b_i \\
 R_m: y &= m_m x + b_m \\
 x &= (b_m - b_i) / (m_i - m_m) \\
 y &= m_i [(b_m - b_i) / (m_i - m_m)] + b_i
 \end{aligned}$$

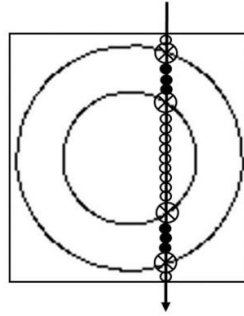


Figure 6. 1D Method - Filled in circles are voxels that are classified as inside the contour while open circles are voxels classified as outside the contour. The circles with an “x” are the intersection points and are counted as inside

Using the intersection points calculated above, all the voxels along the ray can be easily classified as inside or outside the surface model using the following three cases. In the first case, no edges intersect the ray and therefore all the voxels along the ray are classified as outside. In the second case there is only one intersection with the ray and therefore only the intersection voxel is classified as inside and the rest are classified as outside. Finally, the third case is that there is more than one intersection with the ray. In the case of more than one intersection, all the voxels from the beginning of the ray to the first intersection are classified as outside and all the voxels after the first intersection until the second intersection are classified as inside and this pattern alternates until the end of the ray is reached. See Figure 6 above for an illustration of this method. In this case, we assume that all the voxels on the border of the field of view are outside. Finally, using the same scoring method and volume model creation technique as that of the 3D method, volume model is created. There are four special cases provided below.

- I) m_i is undefined and hence no intersection point exists.
- II) if $m_m = m_i$, then the edge and ray casted are parallel and hence no intersection point exists.
- III) if the x value of the ray casted is outside the endpoints of the edges then no intersection is checked.
- IV) There is a special case in which the direction of the ray selected is perpendicular to an axis, which simplifies the method. Let us assume that it is perpendicular to the x -axis as Figure 6 above shows. In this case, ray casting will happen as follows. Given this special case, Equation Set 7 reduces to Equation Set 8 in which y is simply the value of the sampling step.

Equation Set 8:

$$\begin{aligned}
 L_i: y &= x_o \\
 L_m: y &= m_m x + h_m \\
 x &= \frac{x_o - h_m}{m_m} \\
 y &= m_m \left(\frac{x_o - h_m}{m_m} \right) + h_m
 \end{aligned}$$

Given all the intersection points, the classification and volume creation processes proceed as in the general case.

1D Computational Complexity Analysis

The above method has two major steps. The first major step is cutting the model which is the same as that of the 2D method and has a complexity of $O(q * T_i)$. The second major step is classifying a voxel as inside or outside the 2D contour whose complexity is the number of rays casted parallel to the y -axis. The number of rays casted is constant (resolution of the image in the x direction) and limited to the edges where the x value of the casted ray is between the endpoints of the edges. Thus, the number of edges to be intersected with the casted ray is independent of the

total number of contour edges and is a small and fairly constant number. Hence this step has the complexity of $O(1)$ as compared to 2D's voxel classification step's complexity of $O(m*n*e)$. Therefore, the overall complexity to find all the points inside a surface model using the 1D method is $O(q*T_i) + O(1)$.

Implementation Details, Technologies Utilized and Datasets

This program is created using .NET Framework 4.0. Within this project, various technologies were integrated together to streamline the conversion of a surface model to a volume model. These technologies include Visualization Toolkit (VTK), Imaging Toolkit (ITK) and ClearCanvas. All three libraries, VTK, ITK and ClearCavas, have been built to run on C# .NET framework 4.0. It should be noted that both the VTK library and the ITK library are written in C++ and therefore, CMAKE was used to create the C# DLLs from the native C++ code. Figure 7 depicts the overall system architecture.

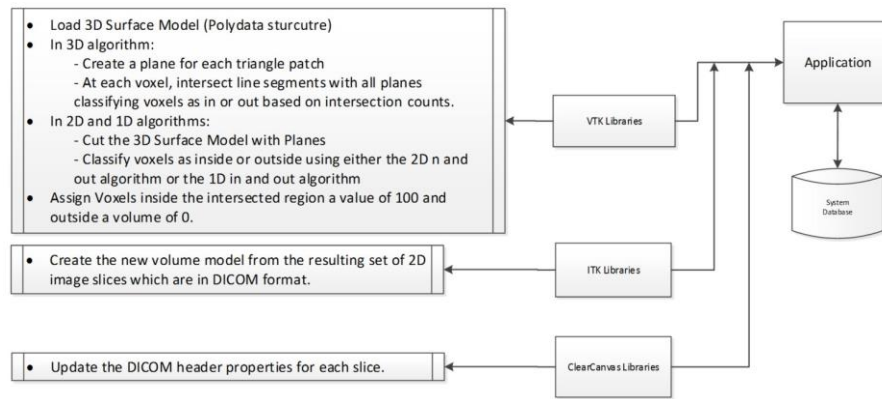


Figure 7. Overall system architecture

Simulated Dataset

The simulated dataset consists of models generated with varying number of triangle patches and complexities. The number of triangle patches is simply the number of triangles that make up the surface model. The models' complexity is measured using the surface model representation. One way to do this is to calculate the ratio of the total surface area of the surface model divided by the volume of the surface model; generally referred to as the *specific surface area* which we simply consider as a measure of Complexity (Eq. 7).

$$Complexity = [Surface Area (M_i)] / [Volume (M_i)] \tag{7}$$

In order to generate the models, the following parameters were used and can be depicted in Figure 8(a).

1. Radius 1, R_1 , used to generate the inner radius
2. Radius 2, R_2 , used to generate the outer radius
3. Number of stripes, m , where each stripe $S_i, i \in \{1, \dots, m\}$, has radius R_{3i} and thickness b

Real Dataset

The Deep Perisylvian Area (DPSA) Segmentation and Surface Reconstruction from MR data were acquired on a GE Signa 3 Tesla scanner using a T1-weighted spoiled gradient echo (SPGR) pulse sequence with the following parameters: TE = 4.288 ms, TR = 9.872 ms, matrix size = 256 × 256; FOV = 240 mm × 240 mm; slice thickness = 2 mm, 124 slices without gaps. A semi-automated thresholding method was used to segment the gray matter. The threshold was set so that apparent contiguity between the DPSA and nearby subcortical structures (e.g., external capsule, putamen) was minimized. Where any scant contiguity remained on the segmented image as a consequence of partial volume and signal-to-noise effects, a manual separation was performed to

isolate the DPSA. The marching cubes method as described earlier was then applied to construct the surface model of the structure.

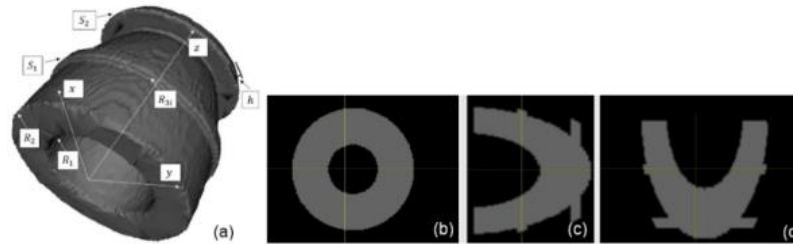


Figure 8. Simulated Data Creation Parameters. Surface (a) and volume model cross sections (b-d) representations

The evaluation consists of accuracy and computational complexity analyses (duration). For computational complexity, we simply consider duration of how long the method ran for. The duration was calculated using a laptop with 8 GB of memory, 1 socket (2 cores) with 4 logical processors at 2.10 GHz and a solid state drive.

Results

In order to evaluate the methods proposed, eleven simulated models were generated using the parameter values given in Table 3; and 20 real models of the human brain DPSA were used. Figure 8(b-d) above contains orthogonal cross sectional views of the simulated volume model.

Table 3. The parameters and complexity of the simulated models

Models ID	R1**	R2***	S _i ****	R3 _i [†]	Triangles
SM*1	30	50	0	N/A	21,148
SM 2	30	50	1	55	23,030
SM 3	30	50	2	55	24,840
SM 4	30	50	2	Random sizes	25,852
SM 5	30	50	3	Random Sizes	23,921
SM 6	30	50	2	60	48,898
SM 7	30	50	4	60	76,868
SM 8	30	50	5	60	89,980
SM 9	30	50	7	60	115,691
SM 10	45	50	5	75	94,171
SM 11	45	50	7	75	121,724

*SM: Simulated Model

**R1: Radius 1 used to generate the inner radius of the simulated data

***R2: Radius 2 used to generate the outer radius of the simulated data

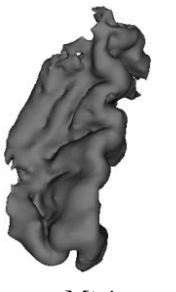
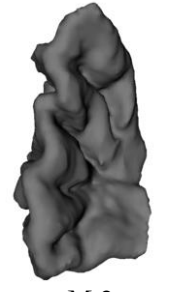
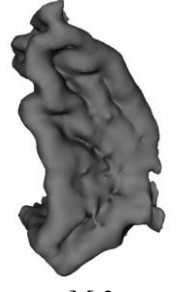
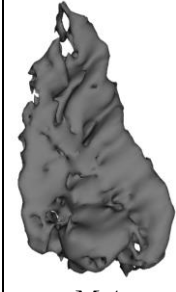
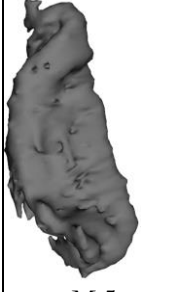
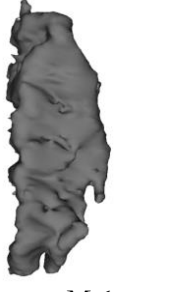
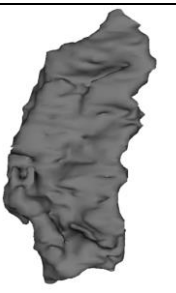
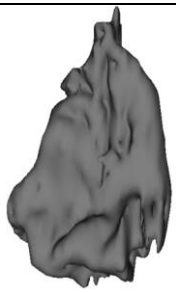
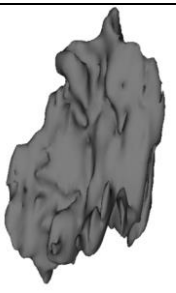

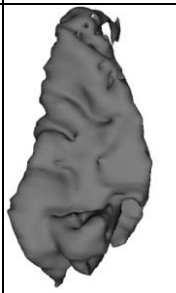

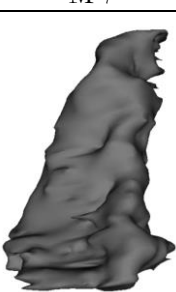


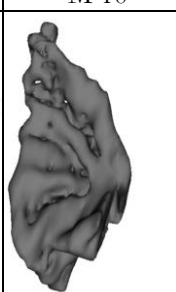

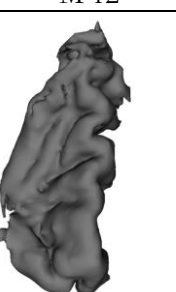
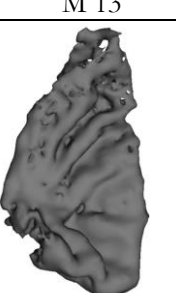

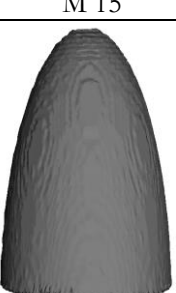

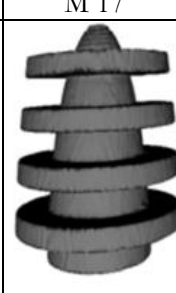

****S_i: Number of stripes included in the simulated data

†R3_i: Radius of the stripes S_i

1. Radius 2, R₂, used to generate the outer radius
2. Number of stripes, w , where each stripe $S_i, i \in \{1, \dots, w\}$, has radius R_{3_i} and thickness h

Table 4 contains visualizations of all the real surface models and four of the simulated surface models.

Table 4. M 1 - M 20 represent the surface model of the real dataset. SM 1, 2, 7 and 11 represent a surface model of a subset of the simulated dataset

					
M* 1	M 2	M 3	M 4	M 5	M 6
					
M 7	M 8	M 9	M 10	M 11	M 12
					
M 13	M 14	M 15	M 16	M 17	M 18
					
M 19	M 20	SM** 1	SM 2	SM 7	SM 11

*M: Model; **SM: Simulated Model

In order to calculate the accuracy of the method, four models are used; the original surface model (S_1), the original volume model (V_1), the generated surface model (S_2) and the generated volume model (V_2). Using those four models, accuracy is measured by calculating the ratio of the intersection of V_1 and V_2 and the union of V_1 and V_2 . Note that intensities that were 0 in V_1 and stayed 0 in V_2 are considered background voxels and hence are not counted. The formal equation to measure accuracy is provided in Eq. 8.

$$Accuracy = (V_1 \cap V_2) / (V_1 \cup V_2) \quad (8)$$

The results of accuracy and duration of applying 2D and 1D methods on the simulated dataset are presented in Table 5. The Simulated Model (SM) IDs in Table 5 match those in Table 3. Note

that there are no results for the 3D method because when it was tested for SM 1, it took several hours to finish a few of the many slices. Extrapolating that duration, we estimated that it would take about 9,867 minutes which is equivalent to 164 hours, and hence slightly under a week to complete the conversion. Therefore, the 3D method is not feasible and so the timings were not recorded. Furthermore, both the 2D and the 1D methods had an average accuracy of over 99.7%. However, the 1D method was 3.75 times faster than the 2D method of 300 and 80 seconds, respectively, in average. Therefore, the 1D method is preferable and is the only method tested against the real dataset. In addition, for the real dataset, the duration in seconds were all either one or two seconds and hence milliseconds was used to evaluate the 1D method's accuracy and duration for the real dataset instead of seconds. Table 6 contains the accuracy and duration in milliseconds of applying only the 1D method to the real dataset.

Table 5. Results of 2D and 1D methods applied to the simulated dataset

SM ID	2D A (%)	2D D (sec)	1D A (%)	1D D (sec)	Complexity	Triangles
1	99.92	144	99.72	27	10.16	21,148
2	99.90	150	99.73	28	10.62	23,030
3	99.92	159	99.72	31	10.97	24,840
4	99.93	160	99.73	32	11.23	25,852
5	99.88	153	99.66	33	10.61	23,921
6	99.91	270	99.76	68	16.36	48,898
7	99.91	405	99.78	105	18.32	76,868
8	99.78	386	99.79	119	20.85	89,980
9	99.90	414	99.72	152	29.35	115,691
10	99.87	495	99.62	129	31.58	94,171
11	99.65	537	99.42	149	46.26	121,724

SM = simulated model; A = accuracy; D = duration

Table 6. Results of 1D method applied to the real dataset

Model (M) ID	1D Accuracy (%)	1D Duration (ms)	Complexity	Triangles
M 1	98.25	1,936	52.86	7,126
M 2	98.83	1,535	40.46	6,352
M 3	98.12	1,629	50.00	7,242
M 4	97.10	2,376	80.40	8,666
M 5	97.31	1,348	45.92	6,740
M 6	99.12	1,563	40.16	7,104
M 7	98.13	1,373	41.55	4,707
M 8	99.01	1,551	43.27	6,410
M 9	98.52	1,408	69.72	5,605
M 10	98.79	1,080	44.67	3,966
M 11	98.50	1,944	60.55	8,524
M 12	98.73	1,374	50.22	6,254
M 13	98.11	1,421	52.44	4,634
M 14	98.60	1,258	48.83	3,459
M 15	97.78	1,212	46.49	5,018
M 16	97.31	1,403	65.45	5,535
M 17	97.38	1,428	72.94	7,049
M 18	97.86	2,064	54.35	8,460
M 19	97.26	1,679	63.83	7,546
M 20	98.83	2,294	47.97	9,324

In order to analyze the data from Tables 5 and 6, a set of figures have been generated. Figures 9 – 12 plot the effect that the number of triangle patches a model contains and the effect of the a model's complexity has on the the surface to volume conversion duration for both simulated and real data, respectively.

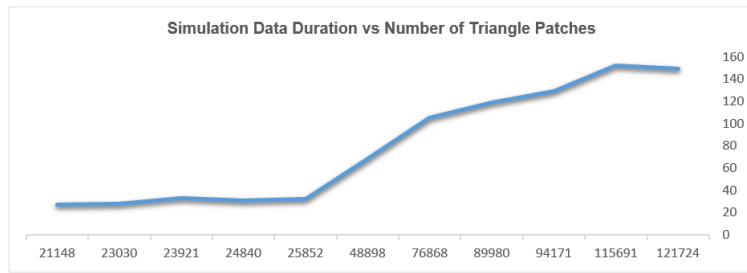


Figure 9. Trend line illustrating simulated data duration vs number of triangle patches

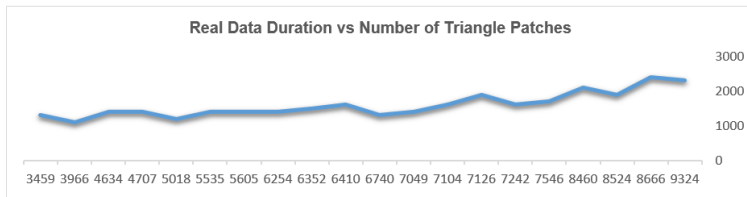


Figure 10. Trend line illustrating real data duration vs number of triangle patches

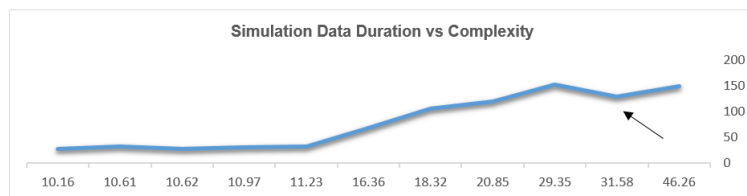


Figure 11. Trend line illustrating simulated data vs complexity

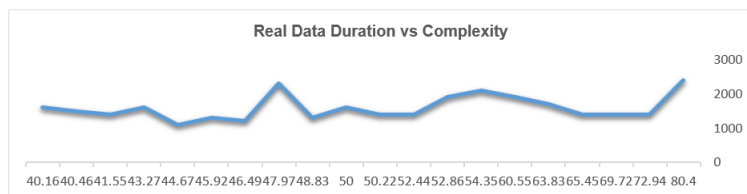


Figure 12. Trend line illustrating real data vs complexity

Figures 13 and 14 are used to examine whether there is a trend between a model's complexity and the number of triangle patches it contains for both simulated and real data, respectively.

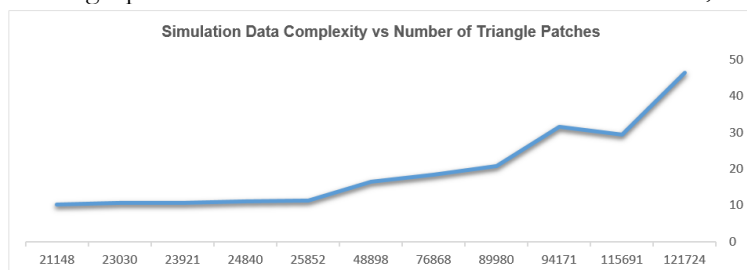


Figure 13. Trend line illustrating simulated data complexity vs number of triangle patches

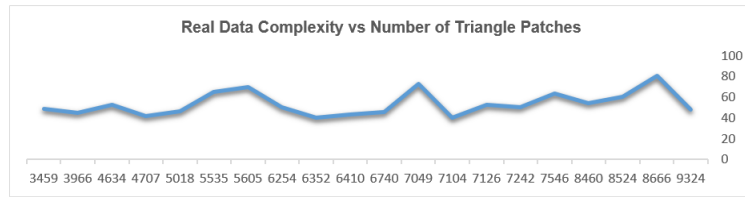


Figure 14. Trend line illustrating real data complexity vs number of triangle patches

Finally, Figures 15 – 18 visualize the effect of a model’s complexity and the effect of the number of triangle patches a model contains on the accuracy of the surface to volume conversion for the real and simulated data, respectively.

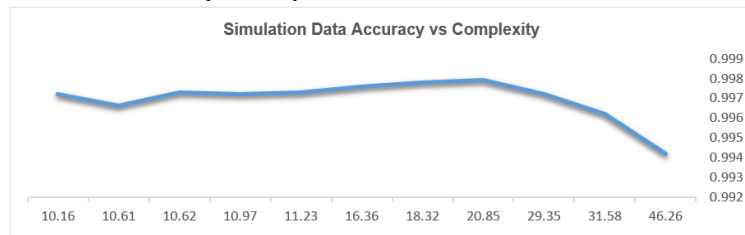


Figure 15. Trend line illustrating simulated data accuracy vs complexity

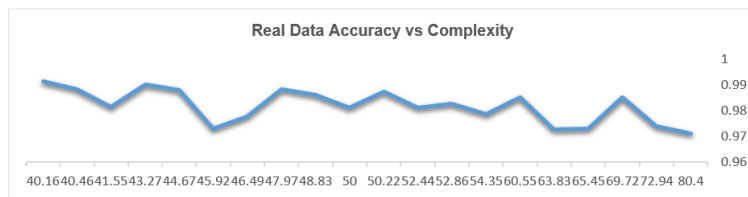


Figure 16. Trend line illustrating real data accuracy vs complexity

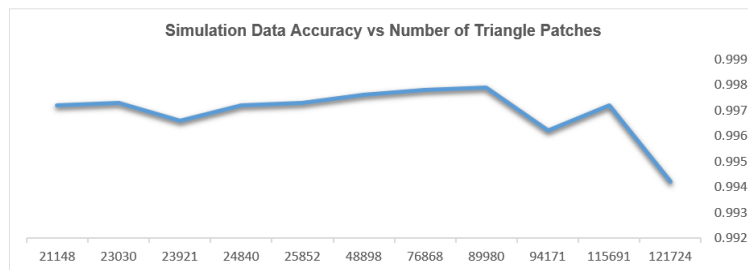


Figure 17. Trend line illustrating simulated data accuracy vs number of triangle patches

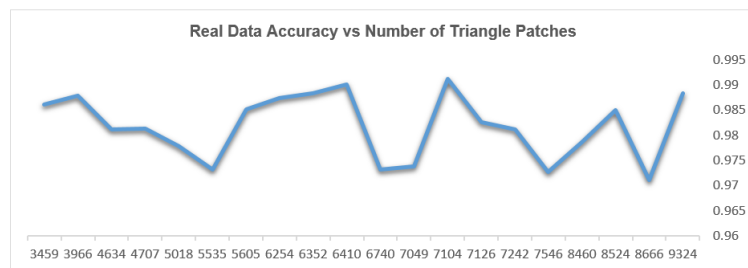


Figure 18. Trend line illustrating real data accuracy vs number of triangle patches

Since sampling is used to convert from surface space to volume space, a discretization effect will take place which is illustrated in Figure 19.

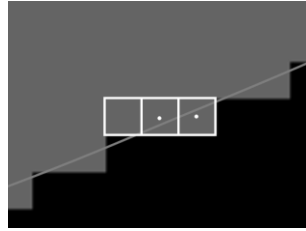


Figure 19. Discretization effect of converting a surface model to a volume model. The border of the continuous model in light gray and the borders of the voxels in white.

1D method compared to state of the art point ray casting method

In this section the duration of the 1D method is compared to VTK’s PointInPolygon method (simply referred to as Ray Casting).

The data in Table 7 provides a summary of the results for comparing the 1D method to Ray Casting. Figure 20 demonstrates the conversion duration of the 1D method compared to Ray Casting.

Table 7. Results of Ray Casting compared to 1D method

Number of Vertices	Ray Casting (ms)	1D (ms)	Ray Casting / 1D Ratio
432	208.24	32.76	6.36
730	639.91	30.57	20.93
962	1,598.96	32.35	49.43
1,050	1,653.59	36.53	45.27
1,276	3,648.83	37.85	96.40
1,312	3,255.02	32.26	100.90
1,330	4,525.55	33.16	136.48
1,486	4,049.78	32.81	123.43
1,492	3,169.1	31.54	100.48
1,560	5,942.87	30.13	197.24
16,137	--	84.8	--

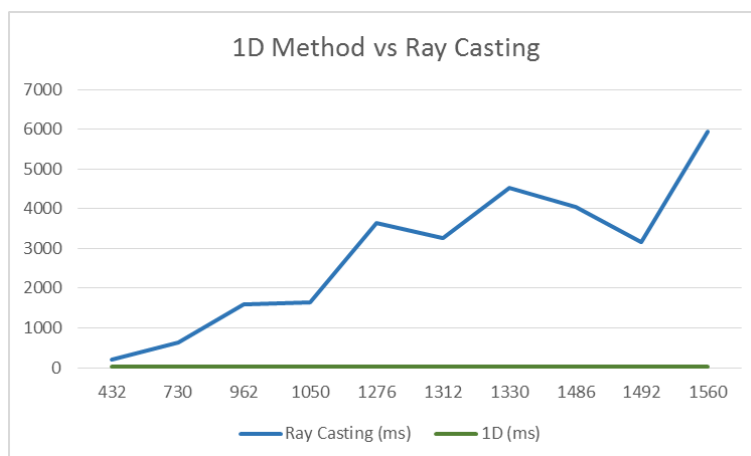


Figure 20. Trend line illustrating the duration of Ray Casting vs 1D Method as the number of vertices increase

Discussion

Surface to Volume Conversion using Proposed Method

Analyzing the results from Tables 5 and 6, the following observations can be made for the 1D method. First, the more triangle patches the simulated model contains, the longer the duration for surface to volume conversions (Figure 9). In general, this is also true for the real dataset (Figure 10). Second, the simulated models' complexity loosely follows the duration pattern with one anomaly (indicated by an arrow in Figure 11). However, the real data does not follow such a pattern (Figure 12).

The two observations above can be explained by the fact that a simple (non-complex) model can be generated using many triangle patches and vice versa. Thus, there is not a strong correlation between the complexity of the model and the number of triangle patches, as illustrated in Figure 13, for the simulated dataset. This is even more apparent in the real dataset where there is no correlation (Figure 14). As discussed in the method section, the dominant factors influencing duration in the proposed method are the number of triangle strips and the number of triangle patches that need to be cut as shown in the first observation. Therefore, in general, we do not expect the duration to have a linear correlation with complexity as shown in the second observation.

The third observation is that the more complex the simulated model is, the lower the accuracy of the surface to volume conversions (Figure 15). The anomaly in Figure 15 can be discarded as the first three complexity values are practically the same. For the real data (Figure 16), an overall decreasing trend is evident. The correlation between the accuracy and the model complexity can be explained by the fact that the more complex the model is the more surface area it would have for a given volume. More surface area means that there are more voxels that make up the border of the surface model. Figure 19 shows a few voxels (white boxes) on the border (light gray) of a model where the effect of discretization is evident. If the center of a given voxel is inside of the border, the voxel is classified as inside. This can obviously present an inaccuracy as part of the voxel is in fact outside of the model. More border voxels can imply higher inaccuracies and therefore more complex models would show higher inaccuracies. This is observed in Table 5 with an average complexity of 19.7 resulting an average accuracy of 99.7% compared to the results in Table 6 with an average complexity of 53.6 resulting in an average accuracy of 98.2%. One may note that other factors like asymmetry can contribute to higher inaccuracy rates.

The fourth observation is that the simulation model's accuracy versus the number of triangle patches are not correlated (Figure 17), which is even more evident for the real dataset (Figure 18). Therefore, among the two measures of model complexity and the number of triangle patches, the former correlates well with accuracy and the latter correlates better with the duration of the surface to volume conversion using the 1D method.

The worst performance, from the accuracy point of view, shows less than 1% deviation from the original volume model for simulated datasets. Even for a relatively large SOI such as DPSA, the method shows less than 3% deviation. Given the fact that the segmentation accuracy that is either manual or automatic is much lower [38], this method would be adequate to be used in a medical application.

1D Method Compared to State of the Art Point Ray Casting Method

Analyzing the results in this table indicate that finding all of the points inside a given polygon with less than 2,000 vertices is up to 200 times faster using the 1D method than using the Ray Casting method. Furthermore, as the number of vertices is increased, the 1D method stays fairly constant while the Ray Casting method increases at least linearly, which is apparent in Figure 20.

It is important to discuss when ray casting can result in false negatives. Two primary scenarios that we have seen in the literature are the tangent line and rounding errors issues. First, if the ray being casted is tangent to the model, then false negatives could occur. However, this has an extremely low likelihood in the order of 10^{-400} [13]. Therefore, this is not of major concern for 3D and 2D methods due to the extremely low likelihood. The proposed 1D method does not suffer

from the tangent line case because we only count an edge intersection if the intersection is between the endpoints of the edge and not the vertex itself. Second scenarios for false negatives can occur due to rounding errors, which may occur for points very close to the boundary. Within the literature we have seen, all of the methods employ a rounding factor causing potential classification errors. However, this is a common issue for any method implemented on a finite precision arithmetic machine.

In this work, the way we have arrived at the 2D method from the 3D method using complexity analysis provided the pattern that we later applied on 2D method to arrive at 1D. In other words, we have proposed a natural progression that results in 1D method which is a novel and extremely fast approach. As discussed in the method section, the complexity of the 1D algorithm is $O(q \cdot T) + O(1)$. Most literature that finds points inside a 2D polygon express complexity in terms of the complexity of classifying a specific voxel inside a 2D polygon [39-42]. In our 1D method, this specific task is done in $O(1)$. Furthermore, the 1D method is efficient because our requirement is to find all the points inside the surface model, which is a more relaxed requirement compared to classifying a specific voxel as inside. The latter is the requirement in all the literature mentioned above [39-42]. To the best of our knowledge there exists no method with such an efficiency.

Overall Query

A typical query consists of two parts. The first part of the query is to narrow down the number of subjects being examined based on structured data such as date of birth, gender and other demographics information. The second part of the query is based on the unstructured content of the image and is usually more time consuming. Let us consider a hypothetical case of a medical database with hundreds of thousands of patients, a typical query may first be filtered to the top 100 subjects using structured data. The unstructured data within the returned 100 subjects narrowed down to five structures of interest along with their contralateral part result in 1,000 models to be converted and queried. Based on Table 6, a relatively complicated model, such as the DPSA, requires an average of 1.5 seconds to be converted. Therefore, converting all 1,000 models in the given scenario takes under half an hour. One may note that the times we reported in Table 6 are based on a laptop with limited computational power and could be significantly faster when using a powerful server. More importantly, it must be noted that this step could be an offline step.

Conclusion and Future Work

Having a system that takes an input of a volume model or a surface model and is able to convert between them is critical. The need to have both forms of the model is important because there are some quantitative calculations that are easier done using the volume model and some that are easier done using the surface model. The main contribution of this work was to provide a 1D method that is able to convert a surface model to a volume model with better efficiency that what is found in the current literature. In addition, the methodology proposed by the natural progression from 3D to 2D was quite novel and inspired the development of the 1D method. Overall, the results indicate that the conversion process is not lossless. However, the amount of data lost is very little and may not jeopardize the surface model's features. The 1D method proposed in this paper is sufficient in converting the surface model to a volume model in terms of accuracy and time requirements. Therefore, it can be used in real life radiological systems capable of querying different aspects of the structure of interest and thus the content of medical images.

With the ability to efficiently convert a surface model to a volume model, as a future work we can build a general framework of an unlimited query module where feature extraction occurs on demand, rather than precalculated, and stored in the database as shown in Figure 1. Depending on the features of interest (volume features or surface features), the query engine will invoke the proper conversion method if the model is not available in the desired format (volume or surface). The query engine communicates with the feature extraction module to calculate specific features on demand using the converted models.

Equations' Symbols

- P_T : Plane for Triangle Patch T
 M : Matrix consisting of vertices' coordinates
 $\mathbf{1}$: vector containing elements [1 1 1]
 \mathbf{n} : normal vector
 R : Set of Rays
 (x_b, y_b, z_b) : A point within the 3D bounding box
 $(x_{bni}, y_{bni}, z_{bni})$: A point within the 3D neighboring structure to the bounding box
 $P_k(x_k, y_k, z_k)$: Intersection Point k
 AB : Segment of the given triangle patch
 AC : Segment of the given triangle patch
 BC : Segment of the given triangle patch
 R_n : Number of rays
 T_i : Number of triangle patches
 P_i : Plane used to cut the surface model
 $(x_i, y_i, z_i), (x_j, y_j, z_j)$: Endpoints of a triangle patch's segment
 m : Random slope
 (X_{poi}, Y_{poi}) : coordinates of the point of interest being tested
 $(X_{es}, Y_{es}), (X_{ee}, Y_{ee})$: Edge start and Edge end points of the 2D contour
 (X_{fi}, Y_{fi}) : First intersection point found
 (X_{ni}, Y_{ni}) : New intersection point found
 P_i : Special plane which goes through the Z-axis
 R_i : Ray generated at various sampling steps
 m_i : Slope of ray R_i
 h_i : y-intercept of ray R_i
 R_m : Edge point of 2D contour
 m_m : Slope of edge point of 2D contour
 h_m : y-intercept of edge point of 2D contour
 x_o : Value of x at the x-axis sample location
 M_i : The given model
 V_1 : Original volume model generated from simulation or segmentation
 V_2 : Volume model generated from conversion method

Conflict of Interest

The authors declare that they have no conflict of interest.

References

1. Dimitrovski I, Kocev D, Kitanovski I, Loskovska S, Dzeroski S. Improved Medical Image Modality Classification using a Combination of Visual and Textual Features. *Comput Med Imaging Graph* 2015;39:14-26.
2. Srinivas M, Mohan C. Medical Images Modality Classification using Multi-Scale Dictionary Learning. 19th International Conference on Digital Signal Processing 2014:621-625. doi: 10.1109/ICDSP.2014.6900739
3. Deliabsis KK, Kechriniotis A, Maglogiannis I. A Novel Tool for Segmenting 3D Medical Images Based on Generalized Cylinders and Active Surfaces. *Comput Methods Programs Biomed* 2013;111:148-165.
4. Roobottom CA, Mitchell G, Morgan-Hughes G. Radiation-Reduction Strategies in Cardiac Computed Tomographic Aangiography. *Clin Radiol* 2010;65:859-867.
5. Organisation for Economic Co-operation and Development. [Internet] [cited 2 March 2015]

- Available from: <http://www.oecd.org>
6. Glatard T, Montagnat J, Magnin IE. Texture Based Medical Image Indexing and Retrieval: Application to Cardiac Imaging. ACM SIGMM International Workshop on Multimedia Information Retrieval 2004;135-143.
 7. Muller H, Michoux N, Bandon D, Geissbuhler A. A Review of Content-Based Image Retrieval Systems in Medical Applications- Clinical Benefits and Future Directions. *Int J Med Inform* 2004;73:1-23.
 8. Lehmann TM, Güld MO, Thies C, Fischer B, Key-Sers M, Kohlen D, et al. Content-Based Image Retrieval in Medical Applications for Picture Archiving and Communication Systems. *Proceedings of the SPIE Conference on Medical Imaging* 2003;5033.
 9. Siegel E, Kolodner RM. *Filmless Radiology*. Springer; 2001.
 10. Medina JM, Jaime-Castillo S, Barranco CD, Campana JR. On the Use of a Fuzzy Object-Relational Database for Flexible Retrieval of Medical Images. *IEEE Transactions Fuzzy Systems* 2012;20:786-803.
 11. Bueno JM, Chino F, Traina AJM, Traina Jr C. How to Add Content-Based Image Retrieval Capability in a PACS. *IEEE Computer-Based Medical Systems Proceedings of the 15th IEEE Symposium* 2002:321-326. doi:10.1109/CBMS.2002.1011397
 12. Jafari-Khouzani K, Siadat MR, Soltanian-Zadeh H. Texture Analysis of Hippocampus for Epilepsy. *Proceedings of SPIE Image Processing Conference* 2003;279-288.
 13. Siadat MR, Soltanian-Zadeh H, Fotouhi F, Elisevich K. Content-Based Image Database System for Epilepsy. *Comput Methods Programs Biomed* 2005;79:209-226.
 14. Liu Y, Zhang D, Lu G, Ma W. A Survey of Content-Based Image Retrieval with High-Level Semantics. *Pattern Recognit* 2007;40:262-282.
 15. Eakins J, Graham M, Franklin T. *Content-Based Image Retrieval*. Library and Information Briefings 1999.
 16. Antani S, Lee DJ, Long LR, Thoma GR. Evaluation of Shape Similarity Measurement Methods for Spine X-Ray Images. *J Vis Commun Image Represent* 2004;15:285-302.
 17. Ghosh P, Antani S, Long LR, Thoma GR. Review of Medical Image Retrieval Systems and Future Directions. *IEEE Computer-Based Medical Systems 24th International Symposium* 2011;1-6.
 18. Hsu W, Antani S, Long LR, Neve L, Thoma GR. SPIRS: A Web-Based Image Retrieval System for Large Biomedical Databases. *Int J Med Inform* 2009;78:13-24.
 19. Chandrakar A, Thoke AS, Singh BK. Indexing and Retrieval of Medical Images using CBIR Approach. *Advances in Parallel Distributed Computing* 2011;393-403.
 20. Ramamurthy B, Chandran R. CBMIR: Content Based Medical Image Retrieval Using Multilevel Hybrid Approach. *Int J Comput Comm Control* 2015;10:3:382-389.
 21. Chowattanakul W, Rai HG, Krishna PR. An Efficient Shape Based Feature for Retrieval of Healthcare Literatures Using CBIR Technique. *ACM Proceedings of the Fourth Annual ACM Bangalore Conference* 2011;24.
 22. Thanellas AK, Pollari M. Compact Review of Structural and Microstructural Brain Image Analysis Methods. *IEEE 23rd International Symposium Computer-Based Medical Systems* 2010;378-382.
 23. Bozzali M, Cercignani M, Caltagirone C. Brain Volumetrics to Investigate Aging and the Principal Forms of Degenerative Cognitive Decline: A Brief Review. *Magn Reson Imaging* 2008;26:1065-1070.
 24. Thanellas AK, Pollari M. Sensitivity of Volumetric Brain Analysis to Systematic and Random Errors. *IEEE 23rd International Symposium Computer-Based Medical Systems* 2010;238-242.
 25. Rodriguez J, Ayala D, Grau S. VolumeEVM: A New Approach for Surface/Volume Integration. *Computers & Graphics* 2005;29:217-224.
 26. Celebi OC, Cevik U. Accelerating Volume Rendering by Ray Leaping with Back Steps. *Comput Methods Programs Biomed* 2010;97:99-113.
 27. Cruces R, Concha L. White Matter in Temporal Lobe Epilepsy: Clinico-Pathological Correlates of Water Diffusion Abnormalities. *Quant Imaging Med Surg* 2015;5:2:264-278.
 28. Spittler K, Tirol F, Fried I, Engel J, Salamon N. Diffusion Tensor Imaging Correlates of

- Hippocampal Sclerosis and Anterior Temporal Lobe T2 Signal Changes in Pharmacoresistant Epilepsy. *Int J Epilepsy* 2014;1:1-7.
29. Liu M, Chen Z, Beaulieu C, Gross D. Disrupted Anatomic White Matter Network in Left Mesial Temporal Lobe Epilepsy. *Epilepsia* 2014;55:5:674-682.
 30. Styner M, Gerig G, Lieberman J, Jones D, Weinberger D. Statistical Shape Analysis of Neuroanatomical Structures Based on Medial Models. *Med Image Anal* 2003;7:207-220.
 31. Hosung K, Tommaso M, Bernasconi N, Bernasconi A. Surface-Based Multi-Template Automated Hippocampal Segmentation: Application to Temporal Lobe Epilepsy. *Med Image Anal* 2012;16:1445-1455.
 32. Heimann T, Meinzer H. Statistical Shape Models for 3D Medical Image Segmentation: A Review. *Med Image Anal* 2009;13:543-563.
 33. Jiang T, Shi F, Zhu W, Li S, Li X. Shape Analysis of Human Brain with Cognitive Disorders. *Digital Human Modeling* 2007;409-414.
 34. Shen D, Zhan Y, Davatzikos C. Segmentation of Prostate Boundaries from Ultrasound Images Using Statistical Shape Model. *IEEE Transactions Medical Imaging* 2003;22:539-551.
 35. Cootes TF, Taylor CJ. Anatomical Statistical Models and Their Role in Feature Extraction. *Br J Radiol* 2004;77:113-139.
 36. Lorensen WE, Cline HE. Marching Cubes: A High Resolution 3D Surface Construction Method. *Computer Graphics* 1987;21:4.
 37. Narkbuakaew W, Sotthivirat S, Gansawat D, Yampri P, Koonsanit K, Areeprayolkij W, et al. 3D Surface Reconstruction of Large Medical Data Using Marching Cubes in VTK. *The 3rd International Symposium on Biomedical Engineering* 2008;64-67.
 38. Narkbuakaew W, Sotthivirat S, Gansawat D, Yampri P, Koonsanit K, Areeprayolkij W, et al. 3D Surface Modeling and Clipping of Large Volumetric Data Using Visualization Toolkit Library. *13th International Conference on Biomedical Engineering* 2009;1144-1148.
 39. Jimenez J, Feito F, Segura R. A New Hierarchical Triangle-Based Point-in-Polygon Data Structure. *Comput Geosci* 2009;35:1843-1853.
 40. Zalik B, Kolingerova I. A Cell-Based Point-in-Polygon Algorithm Suitable for Large Sets of Points. *Comput Geosci* 2001;27:1135-1145.
 41. Ye Y, Guangrui F, Shiqi O. An Algorithm for Judging Points Inside or Outside a Polygon. *Seventh International Conference on Image and Graphics* 2013:690-693. doi: 10.1109/ICIG.2013.140
 42. Jian W, Zongyan C. A Method for the Decision of a Point Whether In or Not in Polygon and Self-Intersected Polygon. *Eighth International Conference on Fuzzy Systems and Knowledge Discovery* 2011:16-18. doi: 10.1109/FSKD.2011.6019594