



# Representing 3-D regions with rational Gaussian surfaces

Marcel Jackowski<sup>a</sup>, Ardeshir Goshtasby<sup>a</sup> and Martin Satter<sup>b</sup>

<sup>a</sup>Computer Science and Engineering Dept., Wright State University, Dayton, OH 45435

<sup>b</sup>Nuclear Medicine/PET, Kettering Medical Center, Kettering, OH 45429

## ABSTRACT

The 3-D regions obtained as a result of a volume image segmentation often need to be geometrically modeled and rendered. In this paper, use of rational Gaussian (RaG) surfaces in modeling and rendering 3-D regions is described. A new parametrization technique is introduced that morphs a sphere in a coarse-to-fine fashion to a 3-D region. Knowing parameters of points on the sphere, parameters of voxels on the region are determined. Having the parameters of the voxels, a RaG surface is then fitted to the voxels to obtain a smooth representation for the region. Examples of the proposed representation on various 3-D regions are presented.

**Keywords:** Image segmentation, rational Gaussian surface, parametric surface, superquadrics, volumetric image

## 1. INTRODUCTION

Image segmentation programs partition an image into regions of interest. In volumetric images, the regions represent 3-D objects or their parts. A 3-D region is usually represented by a set of connected voxels that cover the surface of an object or its part. Representation by voxels is accurate, but it lacks the power to characterize a region or effectively render it.

For rendering purposes, a collection of voxels is often approximated by a triangle<sup>21,29,42</sup> and the triangles are rendered. Optimization algorithms have been developed to minimize the number of triangles representing a discrete 3-D data set.<sup>22</sup> The triangular representation can render 3-D regions better than the voxel representation, but it still lacks the descriptive power. It is desired to represent a region analytically so that various properties of the region could be measured. Implicit and parametric representations have been used to describe 3-D shapes. The representation chosen in this paper is a parametric one, allowing effective characterization and rendering of 3-D regions.

The parametric representation requires that each voxel be assigned a pair of unique parameter values. A method for determining these parameters for a free-form shape will be described. The only restriction imposed on the shape of a 3-D region is that it must not contain a hole. Many human organs have this property, such as the brain, the bones, the hands, the legs, and the entire exterior skin of the body. After parametrizing the voxels in a given shape, we fit a rational Gaussian (RaG) surface to the voxels to obtain a smooth parametric representation for the shape. The RaG formulation has a smoothness parameter that can be selected appropriately to render an object at a desired resolution.

In the following section, the existing 3-D representing methods are reviewed. Next, an algorithm to parametrize voxels in a 3-D region is described. Then, examples of 3-D regions represented by rational Gaussian surfaces are given. Finally, speed and accuracy issues are discussed and concluding remarks are made.

## 2. BACKGROUND

Methods for representing volumetric regions have been around for some time. Roberts<sup>39</sup> used volumetric primitives such as cuboids, wedges, and hexagonal prisms to describe line drawing of polyhedral scenes. Additional support for volumetric primitives based on accessibility, scope and uniqueness, and stability and sensitivity was provided by Marr.<sup>31</sup>

---

Send correspondence to A. Goshtasby, E-mail: agoshtas@cs.wright.edu

A representation that can describe an object with an axis and varying cross-sections is the generalized cylinder.<sup>1,6,9,32-34</sup> A generalized cylinder  $\mathbf{P}(u, v)$  can be formulated as a surface of sweep, obtained by sweeping a curve  $\mathbf{C}(v) = [X(v)Y(v)Z(v)]^t$  along a trajectory  $\mathbf{T}(u)$ <sup>24</sup>:

$$\mathbf{P}(u, v) = \mathbf{T}(u) + X(v)\mathbf{v}_1 + Y(v)\mathbf{v}_2 + Z(v)\mathbf{v}_3, \quad (1)$$

where  $\mathbf{v}_1$  is the tangent vector at point with parameter  $u$ :  $\mathbf{v}_1 = \mathbf{T}'(u)$ , and  $\mathbf{v}_2$  is the principal normal at point with parameter  $u$  on the trajectory. This normal is in the plane normal to the curve and is in the direction the curve bends toward the center of curvature. The principal normal is computed from  $\mathbf{v}_2 = \mathbf{T}''(u)/|\mathbf{T}''(u)|$ , and  $\mathbf{v}_3$  is the binormal, which is the vector normal to both  $\mathbf{v}_1$  and  $\mathbf{v}_2$ :  $\mathbf{v}_3 = \mathbf{v}_1 \times \mathbf{v}_2$ . In the above formula it is assumed that the shape of the cross-section  $\mathbf{C}(v)$  remains constant along the trajectory. If the cross-section varies along the trajectory,  $\mathbf{C}(v)$  should be replaced with  $\mathbf{C}(u, v)$ , where  $u$  is the parameter along the trajectory. In tomographic images, these curves can be considered the cross-sections of the shape with axial, sagittal, or coronal planes.

Biederman<sup>5</sup> argues that just as combinations of 55 phonemes can represent all words in a language, a limited number of geometric primitives should be able to represent all objects. He proposed a set of generalized cylinders as the primitives and developed a process that could recognize objects with the primitives as parts. Based on a similar idea, Pentland<sup>35</sup> used ten generalized cylinders as primitives and modeled a large number of 3-D objects.

Generalized cylinders evolved to superquadrics later. Superquadric surfaces were first formulated by Danish writer and inventor Piet Hein in the early 1960s.<sup>2,14</sup> Their use in representation of volumetric primitives was brought to light by Barr in 1981.<sup>4</sup> Soon after, Pentland<sup>35</sup> and Solina and Bajcsy<sup>44</sup> used superquadrics to represent shapes with global geometric deformations. Superquadrics in parametric form are defined by

$$x(\theta, \phi) = s_1 \text{sign}(\cos \theta \cos \phi) |\cos \theta \cos \phi|^{\alpha_1}, \quad (2)$$

$$y(\theta, \phi) = s_2 \text{sign}(\sin \theta \cos \phi) |\sin \theta \cos \phi|^{\alpha_2}, \quad (3)$$

$$z(\theta, \phi) = s_3 \text{sign}(\sin \phi) |\sin \phi|^{\alpha_3}, \quad (4)$$

where  $s_1, s_2$ , and  $s_3$  are scaling factors along  $x$ ,  $y$ , and  $z$  axes, respectively;  $\alpha_1, \alpha_2$ , and  $\alpha_3$  control the shape of a superquadric, and by varying them, different shapes are obtained. Parameters  $\theta \in [0, 2\pi]$  and  $\phi \in [-\frac{\pi}{2}, \frac{\pi}{2}]$  are spherical parameters of the shape.<sup>20</sup> Parameters  $s_1, s_2, s_3, \alpha_1, \alpha_2$ , and  $\alpha_3$  are determined through a minimization process to fit a superquadric to a set of points.

Since superquadrics have only a few parameters, they can capture only the global properties of a shape. To capture the local shape details, Pentland applied polynomial deformations to superquadrics.<sup>36,37</sup> To provide both global and local shape control, Terzopoulos and co-workers combined superquadrics with generalized splines.<sup>45,46</sup> They proposed a dynamic model in which the local and global shape parameters could be determined by taking into consideration the external forces defined by the image and the internal forces defined by the model. The process starts from an initial state and gradually migrates to the final state through a minimization process.<sup>16</sup>

To represent local shape details in a superquadric, Bardinet *et al.*<sup>3</sup> viewed the problem as one of free-form deformation. They determined a transformation function that could deform a superquadric to approximately fit a given set of points. The process was based on the free-form deformation of Sederberg and Parry,<sup>43</sup> who used Bernstein polynomials as transformation functions. The process required mapping each given point into a unique point on the superquadric. Knowing the parameters of points on the superquadric, they then determine the parameters of given points. This parametrization enables deforming the superquadric to approximate a given set of points. The main task in this method is the mapping of given points to points on the approximating superquadric.

To overcome the limitations of superquadrics, Cohen and Cohen<sup>11</sup> suggested use of a model whose number of free parameters could be adjusted to the complexity of the shape. They used a hybrid hyperquadric with the capability to include a sufficient number of exponentials of hyperquadric terms to obtain a volumetric primitive of a desired shape. The parameters of a hybrid hyperquadric shape were determined by the Levenberg-Marquardt optimization method.<sup>16</sup>

Hyperquadrics are defined by<sup>20</sup>

$$H(x, y, z) = \sum_{i=1}^N |(a_i x + b_i y + c_i z + d_i)|^{\gamma_i} = 1, \quad (5)$$

where  $a_i, b_i, c_i$ , and  $d_i$  are coefficients of a plane that approximates points in a local neighborhood and  $\gamma_i$  is a number larger than 0, determining the local influence of the plane on the obtained shape. For  $\gamma_i$  larger than 1, strictly convex shapes are obtained. For  $\gamma_i$  less than 1, concavities can be produced, but this makes the underlying optimization very unstable, often resulting in singularities.<sup>11</sup> Observing that sums of exponentials of hyperquadrics produce a hilly landscape with bumps or intrusions centered around each component equation, Hanson<sup>20</sup> replaced the equations of planes with the exponentials to create local shape deformations on a hyperquadric surface. A composite hyperquadric surface is defined by<sup>20</sup>

$$CH(x, y, z) = \sum_{k=1}^K c_k \exp(-H_k(x, y, z)) = 1, \quad (6)$$

where  $H_k(x, y, z)$  is the equation of a hyperquadric shown by (5) and  $c_k$  is the contribution of the  $k$ th hyperquadric in local shape of the obtained model. A positive  $c_k$  contributes to the convexity, while a negative  $c_k$  contributes to the concavity of the model in a local neighborhood. Parameters of the composite hyperquadrics are also determined by an optimization process in the same way the parameters of the hyperquadrics are determined.

If a polyhedral mesh is already known approximating a given data set, parametric surface patches that approximate the data can be obtained. Gim<sup>19</sup> used manifolds to stitch B-spline patches together to represent a shape, while Catmull and Clark<sup>10</sup> via subdivision obtained a B-spline approximation to a mesh. Three- and five-sided patches<sup>41</sup> as well as  $n$ -sided patches<sup>27</sup> have also been used to obtain a smooth surface approximation to an irregular mesh. Peters<sup>38</sup> and Loop<sup>28</sup> isolated irregularities in a mesh by one or more refinements and fitted a parametric surface, such as B-splines, to the mesh.

In this paper, the rational Gaussian (RaG) surface,<sup>17,18</sup> which is a parametric surface allowing rendering of a free-form shape at varying resolutions is used. The equation of a RaG surface is:

$$\mathbf{P}(u, v) = \sum_{i=1}^N \mathbf{V}_i g_i(u, v), \quad (7)$$

where  $\mathbf{V}_i$  is the  $i$ th control point on the shape and  $g_i(u, v)$  is the  $i$ th basis function of the surface, defined by

$$g_i(u, v) = \frac{G_i(u, v)}{\sum_{i=1}^N G_i(u, v)}. \quad (8)$$

For a spherically parametrized surface with  $u$  denoting parametrization along the hemisphere,

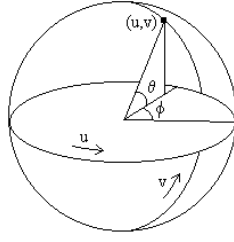
$$G_i(u, v) = \sum_{j=-\infty}^{\infty} \exp\{-[(u - u_i + j)^2 + (v - v_i)^2]/2\sigma_i^2\} \quad (9)$$

where  $\sigma_i$  denotes the standard deviation of the  $i$ th Gaussian. The larger the value of  $\sigma_i$  the smoother the obtained shape in the neighborhood of  $(u_i, v_i)$ , the  $i$ th node of the surface. If the points are parametrized spherically using  $(\theta, \phi)$ , we will have  $u = \theta/2\pi$  and  $v = (\phi + \pi/2)/\pi$ . Although theoretically parameter  $j$  varies from  $-\infty$  to  $+\infty$  because a 2-D Gaussian extends to infinity in all directions and along parameter  $u$  it wraps around the object infinitely, but in practice, a Gaussian approaches zero exponentially and  $j$  needs to be varied from  $-m$  to  $+m$ , where  $m$  is a small number such as 1 or 2.

A single RaG surface can represent a very complex shape. This representation, however, requires that  $\{(u_i, v_i) : 1, 2, \dots, n\}$  be given. Parameters  $(u_i, v_i)$  define the node of the surface at the  $i$ th control point. In the following, we will show how to determine the nodes at voxels on a region and how to represent a region by a RaG surface.

### 3. SPHERICAL PARAMETRIZATION OF A 3-D REGION

In this section an algorithm for parametrizing voxels in the bounding surface of a 3-D region is given. The basic idea of the algorithm is to morph a sphere in a coarse-to-fine fashion to take the shape of the region. Then, by knowing the parameters of points on the sphere, we will determine the parameters of the voxels. We will use the following definitions to describe the algorithm.



**Figure 1.** Each point on the sphere can be referenced by parameters  $u, v \in [0, 1]$ .

**Parametric Equation of a Sphere:** The sphere of radius  $r$  centered at  $(x_0, y_0, z_0)$  is parametrically defined by<sup>24</sup>

$$x = x_0 + r \cos \theta \cos \phi, \quad \theta \in [-\pi/2, \pi/2], \quad (10)$$

$$y = y_0 + r \cos \theta \sin \phi, \quad \phi \in [0, 2\pi], \quad (11)$$

$$z = z_0 + r \sin \theta. \quad (12)$$

By setting  $\theta = \pi v - \frac{\pi}{2}$  and  $\phi = 2\pi u$ , we see that when varying  $u$  and  $v$  from 0 to 1 we will be able to draw the sphere in its entirety. In other words, each point on the sphere can be uniquely referenced by parameters  $u, v \in [0, 1]$ . (See Fig. 1.)

**Polyhedral Approximation of a Sphere:** A sphere can be approximated by a polyhedron whose vertices lie on the sphere. As the number of polyhedral vertices increase, the approximation becomes more accurate. We will denote the vertices of the polyhedron approximating a sphere by  $\mathbf{V} = \{\mathbf{V}_i : i = 1, 2, \dots, N_v\}$ . Note that since parameters  $(u, v)$  at each point on the sphere are known, parameters at the vertices of the approximating polyhedron will be known. We assume all polyhedral faces are triangular and denote parameters at vertex  $\mathbf{V}_i$  by  $(u_i, v_i)$ .

**A Digital Shape:** We will call a 3-D region in a volumetric image a digital shape, a digital object, or simply a shape or an object. A digital shape is defined by a set of voxels defining the bounding surface of a 3-D region. Assuming  $N_o$  voxels form a shape, we will denote the voxels by  $\mathbf{p} = \{\mathbf{p}_i : i = 1, 2, \dots, N_o\}$ .

**Polyhedral Approximation of a Shape:** A digital shape can be approximated by a polyhedron in such a way that the vertices of the polyhedron lie on the shape. The approximation can be made more accurate by increasing the number of faces or vertices of the polyhedron. We assume polyhedral faces are triangular and a polyhedron has  $N_v$  vertices, denoted by  $\mathbf{v} = \{\mathbf{v}_i : i = 1, 2, \dots, N_v\}$ . Note that  $\mathbf{v}$  is a subset of  $\mathbf{p}$ .

**Parametrizing Points on a Shape:** By determining a one-to-one correspondence between vertices of a polyhedron approximating a sphere and vertices of a polyhedron approximating a shape, we can parametrize the points (voxels) on the shape by knowing the parameters of points on the sphere. To achieve this mapping, approximation of the sphere and the shape is carried out in parallel in such a way that in each iteration the same vertex configuration is obtained in polyhedral approximations of both the sphere and the shape.

**Edge Contour:** Suppose that  $\mathbf{v}_i \mathbf{v}_j$  is an edge of the polyhedron approximating a shape. Projection of this edge to the shape is obtained by finding the closest path on the shape that connects  $\mathbf{v}_i$  and  $\mathbf{v}_j$ . We denote this contour by  $\widehat{\mathbf{v}_i \mathbf{v}_j}$ . Therefore, contour  $\widehat{\mathbf{v}_i \mathbf{v}_j}$  represents a sequence of voxels on the shape from  $\mathbf{v}_i$  to  $\mathbf{v}_j$ , connecting the two points in a minimal path.

**Midpoint of an Edge Contour:** Given an edge contour  $\widehat{\mathbf{v}_i \mathbf{v}_j}$ , the midpoint of the contour, denoted by  $\mathbf{m}_{ij}$ , is the point on the contour that has the same length paths to points  $\mathbf{v}_i$  and  $\mathbf{v}_j$ . Parameters  $(u_{ij}, v_{ij})$  at  $\mathbf{m}_{ij}$  are set to the averages of corresponding parameters at  $\mathbf{v}_i$  and  $\mathbf{v}_j$ . Therefore, assuming parameters at  $\mathbf{v}_i$  and  $\mathbf{v}_j$  are  $(u_i, v_i)$  and  $(u_j, v_j)$ , respectively, we find  $u_{ij} = (u_i + u_j)/2$  and  $v_{ij} = (v_i + v_j)/2$ .

An easy way to determine the midpoint of contour  $\widehat{\mathbf{v}_i \mathbf{v}_j}$  is to mark all voxels adjacent to  $\mathbf{v}_i$  and  $\mathbf{v}_j$ . This will mark small circles around  $\mathbf{v}_i$  and  $\mathbf{v}_j$ . If we mark voxels that are adjacent to the marked voxels around  $\mathbf{v}_i$  and  $\mathbf{v}_j$ , we

will obtain larger circles around  $\mathbf{v}_i$  and  $\mathbf{v}_j$  on the shape. If we continue doing this in parallel from the two ends, we will see that at some point the circles centered at  $\mathbf{v}_i$  and  $\mathbf{v}_j$  will grow large enough to meet at a point (voxel). This is detected when a voxel is marked twice. If the two circles touch at more than one voxel, the mid voxel is taken to represent the center point of contour  $\widehat{\mathbf{v}_i\mathbf{v}_j}$ . It should be mentioned that it is not necessary to find the shortest path from  $\mathbf{v}_i$  to  $\mathbf{v}_j$  in order to find  $\mathbf{m}_{ij}$ . As a matter of fact, it is not necessary to find any of the edge contours in the proposed algorithm. It is only necessary to find the midpoints of the contours to implement the algorithm.

**A Triangular Patch:** This is a patch obtained by projecting the edges of a polyhedral face to the shape that it is approximating. We will denote the patch obtained as a result of projecting a polyhedral face with vertices  $\mathbf{v}_i, \mathbf{v}_j$ , and  $\mathbf{v}_k$  to a shape by  $\mathbf{t}_{ijk}$ . Note that patch  $\mathbf{t}_{ijk}$  is bounded by edge contours  $\widehat{\mathbf{v}_i\mathbf{v}_j}$ ,  $\widehat{\mathbf{v}_j\mathbf{v}_k}$ , and  $\widehat{\mathbf{v}_k\mathbf{v}_i}$ .

**A Triangular Face:** This is one face of a polyhedron approximating a shape. If the vertices of a triangular face are  $\mathbf{v}_i, \mathbf{v}_j$ , and  $\mathbf{v}_k$ , we will denote the face by  $\mathbf{T}_{ijk}$ . Note that triangular face  $\mathbf{T}_{ijk}$  has the same vertices as triangular patch  $\mathbf{t}_{ijk}$ .

**Subdivision of a Triangular Patch:** A triangular patch is subdivided into four by finding shortest paths on the shape between the midpoints of the edge contours defining the patch. Therefore, assuming triangular patch  $\mathbf{t}_{ijk}$  is given and if midpoints on the edge contours of the patch are denoted by  $\mathbf{v}_K = \mathbf{m}_{ij}$ ,  $\mathbf{v}_I = \mathbf{m}_{jk}$ , and  $\mathbf{v}_J = \mathbf{m}_{ki}$ , we will denote the four triangular patches obtained as a result of the subdivision by  $\mathbf{t}_{iJK}$ ,  $\mathbf{t}_{Ijk}$ ,  $\mathbf{t}_{IJk}$ , and  $\mathbf{t}_{IJK}$ .

**Subdivision of a Triangular Face:** Since a triangular face has a corresponding triangular patch, by knowing the vertices of the triangle:  $\mathbf{v}_i, \mathbf{v}_j$ , and  $\mathbf{v}_k$ , and the midpoints of the edge contours:  $\mathbf{v}_K, \mathbf{v}_I, \mathbf{v}_J$ , we find a triangular face for each triangular patch. Therefore, after subdividing triangular face  $\mathbf{T}_{ijk}$ , we will obtain  $\mathbf{T}_{iJK}$ ,  $\mathbf{T}_{Ijk}$ ,  $\mathbf{T}_{IJk}$ ,  $\mathbf{T}_{IJK}$ . Note that when a triangular patch is subdivided, the subdivision does not affect the adjacent patches because an edge contour is only split into two. Its position does not change. However, when a polyhedral face is subdivided, since segments of an edge before and after a subdivision are not the same, when one polyhedral face is subdivided and its adjacent face is not, a hole will appear in the polyhedron. To avoid this from happening, when a triangle is subdivided, triangles adjacent to it that are not subdivided, are automatically subdivided. Such triangles do not need to be divided into four, but rather into two by connecting the new vertex obtained from the midpoint of an edge contour to the triangular vertex facing it.

**Relations between Number of Vertices, Number of Edges, and Number of Faces in a Polyhedron:** Assuming in a polyhedron there are  $N_v$  vertices,  $N_e$  edges, and  $N_f$  faces, since each face contains 3 vertices, if all triangular faces are obtained by subdividing a triangle into four, each vertex will be shared by four faces, and we have

$$N_v = \frac{3}{4}N_f. \quad (13)$$

If  $N_{f1}$  of the triangles are obtained by subdividing a triangle into four and  $N_{f2} = N_f - N_{f1}$  of the triangles are obtained by subdividing a triangle into two, we have

$$N_v = \frac{3}{4}(N_{f1} + \frac{N_{f2}}{2}). \quad (14)$$

If there are  $N_e$  edges in the polyhedron, since each edge is shared by exactly two triangles, but each triangle uses three edges, we have

$$N_f = \frac{2}{3}N_e. \quad (15)$$

**Parallel Subdivision of a Sphere and a Shape:** As a shape is subdivided into smaller triangular patches, the corresponding polyhedral faces are subdivided into smaller triangles. Imagine starting with a similar polyhedral approximation of a sphere and a shape. Then, as each triangular patch in the shape is subdivided, the corresponding triangle in the polyhedral approximation of the shape and the sphere are also subdivided. The subdivision in the shape is straightforward since a triangular patch has a corresponding triangular face. The subdivision in the sphere is achieved by connecting the center of the sphere to the midpoints of the edges of the triangular face under consideration

and determining their intersections with the sphere. The intersections enable subdivision of a face in the polyhedral approximation of the sphere.

To start, we find the largest vertical line segment inside the shape. Let's suppose points on the shape that are also at the end points of this line are  $\mathbf{v}_1$  and  $\mathbf{v}_2$  and the length of the line is  $2r$ . We will then initialize a sphere with radius  $r$  at  $\mathbf{C} = (\mathbf{v}_1 + \mathbf{v}_2)/2$ . Suppose also that we obtain an octahedral approximation to the sphere and the vertices of the octahedron are  $\{\mathbf{V}_i : i = 1, 2, \dots, 6\}$ . Note that  $\mathbf{V}_1 = \mathbf{v}_1$  and  $\mathbf{V}_2 = \mathbf{v}_2$ . To find points on the shape corresponding to vertices of the octahedron approximating the sphere, we connect the center of the sphere  $\mathbf{C}$  to  $\{\mathbf{V}_i : i = 3, 4, \dots, 6\}$  and find the intersections of obtained lines with the shape. Let shape points corresponding to octahedral vertices be  $\{\mathbf{v}_i : i = 1, 2, \dots, 6\}$ . Therefore, before starting any subdivision, vertices of the octahedron approximating a sphere  $\{\mathbf{V}_i : i = 1, 2, \dots, 6\}$  will correspond to vertices of polyhedron approximating a shape:  $\{\mathbf{v}_i : i = 1, 2, \dots, 6\}$ . Subdivision is carried out in parallel in such a way that always the same polyhedral configuration is obtained by approximating a shape and a sphere. By knowing parameters of vertices of the polyhedron approximating the sphere we will then know parameters of corresponding vertices in polyhedral approximation of the shape.

Note that if the objective is to approximate a shape by a polyhedron, we should follow the rules of subdivision for a triangular face. However, if we want to represent a shape by a smooth parametric surface, we should follow the rules of subdivision for a triangular patch.

**Parameters at Voxels in a Triangular Patch:** Parameters at the vertices of a triangular patch are the same as parameters at vertices of the approximating polyhedron. Since there is a one-to-one correspondence between the polyhedral approximation of a shape and the polyhedral approximation of a sphere, we will know parameters at the vertices of the polyhedron approximating a shape from parameters at corresponding polyhedral vertices approximating a sphere. The parameters at midpoints of edge contours are the same as parameters at midpoints of corresponding polyhedral edges. We recursively subdivide a triangular patch into four subtriangles and determine parameters at the vertices of newly obtained triangles until parameters at all voxels in a triangular patch are determined.

**Correspondence between Points in a Triangular Patch and Points in the Approximating Polyhedral Face:** A point (voxel) in a triangular patch and a point in the corresponding polyhedral face that have the same parameters  $(u, v)$  are considered corresponding points.

**Accuracy in Polyhedral Approximation:** The error  $E$  between a shape and its polyhedral approximation is defined by the maximum error between triangular patches in the shape and the corresponding triangular faces. Assuming maximum distance between triangular patch  $\mathbf{t}_{ijk}$  and corresponding triangular face  $\mathbf{T}_{ijk}$  is  $d_l$ , we define

$$E = \max_l d_l. \quad (16)$$

If there are  $N_f$  faces in the polyhedral approximation of a shape, then,  $l = 1, 2, \dots, N_f$ . The maximum distance between a triangular patch and its corresponding triangular face is determined from the maximum distance between corresponding points in the patch and the face. If there are  $N_p$  points (voxels) in a triangular patch and the  $i$ th corresponding points in the patch and the face are  $\mathbf{p}_i$  and  $\mathbf{P}_i$ , we define

$$d_l = \max_i \|\mathbf{p}_i - \mathbf{P}_i\|, \quad (17)$$

where  $i = 1, 2, \dots, N_p$ .

**Accuracy in Surface Approximation:** The error  $e$  between a shape and its approximation by a RaG surface is defined by the maximum distance between corresponding points in the shape and in the surface. Assuming parameters at point (voxel)  $\mathbf{p}_i$  in the shape are  $(u_i, v_i)$ , we define

$$e = \max_i \|\mathbf{p}_i - \mathbf{P}(u_i, v_i)\|. \quad (18)$$

Assuming there are  $N_o$  voxels in the shape, in the above equation we have  $i = 1, 2, \dots, N_o$ .

Equipped with these definitions, we now describe algorithms for approximating a digital shape with a polyhedron or a RaG surface with a required accuracy.

**Algorithm 1 (Polyhedral Approximation):** Given voxels on a digital shape, this algorithm finds a polyhedral approximation to the shape with a required accuracy.

1. **Initialize:** Find the octahedral approximation to the shape and compute the error between each obtained triangular patch and corresponding triangular face of the octahedron. Enter into a list those faces that have errors above the specified tolerance.
2. **Main Step:** If the list is empty, stop. Otherwise, take a triangular face from the list (call it **T**) and subdivide its corresponding triangular patch into four smaller patches. Determine the error between each obtained patch and the corresponding polyhedral face. Enter all subtriangles that have errors above the required tolerance into the list. For triangles adjacent to **T** that are not in the list, subdivide each to two subtriangles and compute the error between each and the corresponding patch. Enter an obtained subtriangle into the list if its error is above the required tolerance.
3. **Continue:** Go to the Main Step.

In this algorithm an adjacency list is kept, showing triangles adjacent to each triangle. This list is continuously revised as the triangles are subdivided. Also not mentioned in the algorithm is the computation of parameters at the vertices of the approximating polyhedron. By knowing the parameters at the octohedron approximating a sphere and the corresponding polyhedron approximating a shape, and by carrying out subdivision of the sphere and the shape in parallel, parameters at the vertices of the polyhedron are computed at each step of the algorithm.

The proposed algorithm produces triangles of various sizes approximating a shape. Complexity in one area in a shape results in localized subdivision in that area. Note that if a given shape has branches, the subdivision will allow approximation of the branches by growing the subdivision into the branches. Therefore, subdivision will stop early in flat areas in a shape, while it will continue in detailed and branching areas until the desired tolerance in approximation is reached. The output of this algorithm will be a triangular mesh, which can be used for rendering and other purposes. The algorithm is not optimal since the initial octahedron that approximates a shape is always oriented vertically irrespective of the geometry of the shape. However, obtained approximation considerably compresses data and produces far fewer triangles than voxels in a shape.

**Algorithm 2 (RaG Surface Approximation):** This subdivision algorithm is similar to that of the polyhedral approximation except that error is measured between points (voxels) in a shape and corresponding points in the approximating RaG surface. Also, note that when a triangular patch is divided into four smaller patches, there is no need to check for adjacent patches and divide them into two (see discussion on Subdivision of a Triangular Patch).

Polyhedral approximation of irregularly-spaced data has been attempted before. Hoppe *et al.*<sup>21</sup> developed a method for triangulating irregularly spaced points on the surface of an object to recover its geometry. Later the method was optimized<sup>22</sup> to minimize the number of triangles while reproducing the same geometry with a required tolerance. An algorithm to represent iso-valued surfaces in a volumetric image was given by Lorensen and Cline.<sup>29</sup> This algorithm, which was called the marching cube, represents a surface by a large number of very small triangles. A method to combine many triangles to larger ones based on distance and angle tests was later developed by Schroeder *et al.*,<sup>42</sup> reducing the number of the triangles considerably while preserving the appearance of the surface. A similar algorithm was given by Turk,<sup>47</sup> which combined triangles in flat areas without changing triangles in high curvature areas to preserve shape while reducing the number of the triangles approximating a shape with a required tolerance. In an algorithm developed by Garland and Heckbert,<sup>15</sup> adjacent vertices were combined by removing the edge between them and reducing the number of triangles in a triangular mesh.

If the objective in mesh simplification is for rendering purposes, viewpoint should be taken into consideration. Based on a selected view, some triangles may become completely hidden, in which case they can be removed, or become very small when viewed, in which case they can be combined with adjacent triangles. View-dependent mesh simplification methods have been proposed by Hoppe<sup>23</sup> and Luebke and Erikson.<sup>30</sup>

Parametrization of mesh vertices is very important for smooth representation of a mesh by a parametric surface and for texture mapping. To determine parameters at mesh vertices, Lee *et al.*<sup>26</sup> simplified a mesh to a base mesh. Parameters were then assigned to vertices in the base mesh, and parametrization of the original mesh was determined through conformal mapping of the base mesh to the original mesh. Eck *et al.*<sup>12</sup> developed a method for

parametrizing a polyhedral mesh using the theory of harmonics. In this method, parameters at vertices of a simple polyhedron having an opening was mapped to a polygonal domain in the plane. For more complex polyhedral shapes with possible holes, parameters were assigned locally to ensure continuity between patches that replaced adjacent polyhedral faces. Later, this method was extended to combine triangular faces into quadrilateral and determined local parametrization of the polyhedral mesh for representation by B-spline patches.<sup>13</sup> A nonlinear optimization method for determining the parameters of a B-spline surface fitting an irregular mesh was also given by Rogers and Fog.<sup>40</sup> Brechbühler *et al.*<sup>7,8</sup> developed an optimization method for a one-to-one mapping of closed, simply connected shapes to a sphere, thus enabling spherical parametrization of closed shapes. The algorithm proposed here determines a spherical parametrization of an irregular mesh approximating a digital region with a required accuracy.

#### 4. RESULTS

Examples of surface fitting to 3-D regions are shown in Figures 2–5. The left-column images show the 3-D regions obtained by an image segmentation method, and the middle-column images show results of RaG surfaces fitting to the regions. The surface fitting process preserves the geometry of the regions while smoothing noise and unnecessary details. The smoothness of a RaG surface can be varied by changing the standard deviation of the Gaussians used to define the surface. The third-column images show surfaces obtained by increasing the smoothness of the surfaces shown in the second column.

The region in the first row represents the left ventricular blood pool obtained by segmenting a cardiac MR image, the region in the second row shows a brain tumor obtained by segmenting a brain MR image, the region in the third row shows a brain obtained by segmenting an MR image, and the region in the fourth row shows the skin of the head obtained by segmenting an MR image. The surface fitting process removes excess noise and details and enables visualization and manipulation of the regions effectively.

#### 5. CONCLUSIONS

Parametric representation of 3-D regions is required in many situations. Parametric representation provides an effective means to render and manipulate the regions. It also enables determination of the local and global properties of a region such as surface normals and surface gradients. In addition, it allows reproduction of the region in a resolution that is most suitable for viewing. Even close up of a region represented by a surface will appear sharp. This is in contrast to the digital representation, which appears blurred at close up.

A key advantage of representing a 3-D region with a parametric surface is that the region can be easily revised by moving the control points of the surface. In a system being developed jointly by Wright State University and Wallace-Kettering Neuroscience Institute, the result of an automatic image segmentation is represented by a parametric surface and the surface is overlaid with the image.<sup>25</sup> The user is provided with the ability to revise the surface interactively while observing image data. In this manner, segmentation errors obtained by an automatic method are quickly corrected by the user.

#### ACKNOWLEDGMENTS

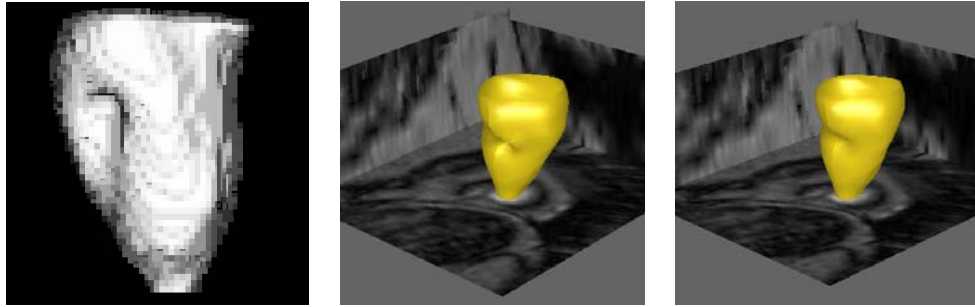
Funding and images for this work were provided by the Wallace-Kettering Neuroscience Institute, Kettering Medical Center, Kettering, Ohio.

#### REFERENCES

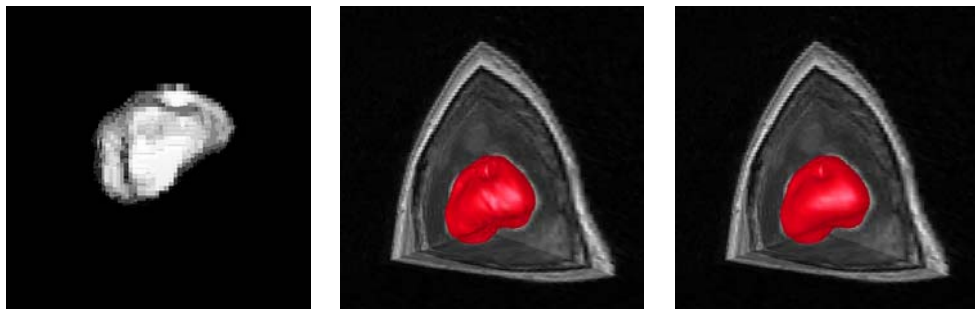
1. G. Agin and T. Binford, Computer description of curved objects, *IEEE Trans. Computers*, vol. 25, no. 4, 1976, 439–449.
2. J. Allard, Note on squares and cubes, *Math. Mag.*, vol. 37, 1964, 210–214.
3. E. Bardinet, L. D. Cohen, and N. Ayache, A parametric deformable model to fit unstructured 3-D data, *Computer Vision and Image Understanding*, vol. 71, no. 1, 1998, 39–54.
4. A. H. Barr, Superquadrics and angle-preserving transformations, *IEEE Computer Graphics and Applications*, vol. 1, 1981, 11–23.
5. I. Biederman, Human image understanding: recent research and a theory, *Computer Vision, Graphics, and Image Processing*, vol. 32, 1985, 29–73.

6. T. Binford, Visual perception by computer, in *Proceedings, IEEE Conf. Systems and Control*, Miami, FL, 1971.
7. Ch. Brechbühler, G. Grrig, and O. Kübler, Surface parametrization and shape description, *SPIE Workshop on Visualization in Biomedical Computing*, vol. 1808, 1992, 80–89.
8. Ch. Brechbühler, G. Gerig, and O. Kübler, Parametrization of closed surfaces for 3-D shape description, *Computer Vision and Image Understanding*, vol. 61, no. 2, 1995, 154–170.
9. R. Brooks, Model-based 3-D interpretation of 2-D images, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 5, no. 2, 1983, 140–150.
10. E. Catmull and J. Clark, Recursively generated B-spline surfaces on arbitrary topological meshes, *Computer Aided Design*, vol. 10, no. 6, 1978, 350–355.
11. I. Cohen and L. D. Cohen, A hybrid hyperquadric model for 2-D and 3-D data fitting, *Computer Vision and Image Understanding*, vol. 63, no. 3, 1996, 527–541.
12. M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle, Multiresolution analysis of arbitrary meshes, *Proc. SIGGRAPH '95*, 1995, 173–182.
13. M. Eck and H. Hoppe, Automatic reconstruction of B-spline surfaces of arbitrary topological type, *Proc. SIGGRAPH '96*, 1996, 325–334.
14. M. Gardner, Mathematical games, *Scientific American*, vol. 213, 1965, 222–234.
15. M. Garland and P. Heckbert, Surface simplification using quadratic error metrics, *Computer Graphics Proceedings*, 1997, 209–216.
16. P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*, Academic Press, London, 1981.
17. A. Goshtasby, Design and recovery of 2-D and 3-D shapes using rational Gaussian curves and surfaces, *Int'l J. Computer Vision*, vol. 10, no. 3, 1993, 233–256.
18. A. Goshtasby, Geometric modeling using rational Gaussian curves and surfaces, *Computer-Aided Design*, May 1995, 363–375.
19. C. M. Grimm and J. F. Hughes, Modeling surface of arbitrary topology using manifolds, *Proc. SIGGRAPH '95* 1995, 359–368.
20. A. J. Hanson, Hyperquadrics: Smooth deformable shapes with convex polyhedral bounds, *Computer Vision, Graphics, and Image Processing*, vol. 44, 1988, 191–210.
21. H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, Surface reconstruction from unorganized points, *Computer Graphics*, 1992, 71–78.
22. H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle, Mesh optimization, *Computer Graphics*, 1993, 19–26.
23. H. Hoppe, View-dependent refinement of progressive meshes, *Computer Graphics Proceedings*, 1997, 189–198.
24. J. Hoschek and D. Lasser, *Fundamentals of Computer Aided Geometric Design*, A. K. Peters, 1993.
25. M. Jackowski, A. Goshtasby, and M. Satter, A computer-aided design environment for segmentation of volumetric image, in preparation.
26. A. W. F. Lee, W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin, MAPS: Multiresolution adaptive parametrization of surfaces, *Computer Graphics Proceedings*, 1998, 95–104.
27. C. Loop and T. DeRose, Generalized B-spline surfaces of arbitrary topology, *Proc. SIGGRAPH '90* 1990, 347–356.
28. C. Loop, Smooth spline surfaces over irregular meshes, *Proc. SIGGRAPH '94* 1994, 303–310.
29. W. Lorensen and H. Cline, Marching cube: A high resolution 3-D surface construction algorithm, *Computer Graphics Proceedings*, vol. 21, no. 4, 1987, 163–169.
30. D. Luebke and C. Erikson, View-dependent simplification of arbitrary polygonal environment, *Computer Graphics Proceedings*, 1997, 199–208.
31. D. Marr, *Vision*, W. H. Freeman, San Francisco, 1982.
32. D. Marr and H. Nishihara, Representation and recognition of the spatial organization of three-dimensional shapes, *Proc. Royal Society of London*, vol. B-200, 1978, 269–294.
33. R. Nevatia and T. Binford, Description and recognition of curved objects, *Artificial Intelligence*, vol. 8, 1977, 77–98.
34. H. K. Nishihara, Intensity, visible-surface and volumetric representations, *Artificial Intelligence*, vol. 17, 1981, 265–284.

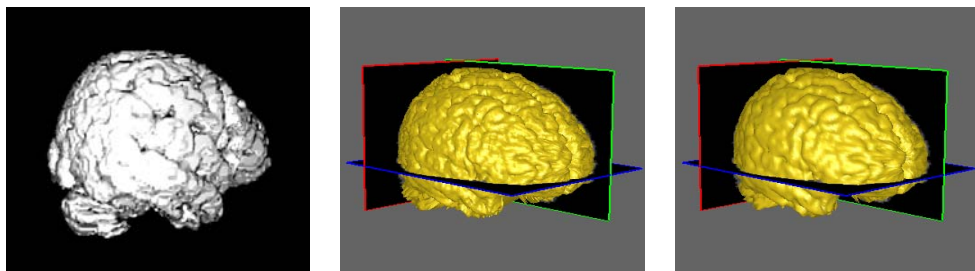
35. A. Pentland, Perceptual organization and the representation of natural form, *Artificial Intelligence*, vol. 28, 1986, 293–331.
36. A. Pentland, Canonical fitting of deformable part models, in *Sensing and Reconstruction of Three-Dimensional Objects and Scenes*, *Proc. SPIE 1260*, B. Girod (ed.), 1990, 216–228.
37. A. Pentland and S. Sclaroff, Closed-form solutions for physically based shape modeling and recognition, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 13, no. 7, 1991, 715–729.
38. J. Peters, Smooth free-form surfaces over irregular meshes generalizing quadratic splines, *Computer Aided Geometric Design*, vol. 10, 1993, 347–361.
39. L. Roberts, Machine perception of three-dimensional solids, in *Optical and Electro-Optical Information Processing*, J. T. Tippett *et al.* (eds.), MIT Press, Cambridge, MA, 1965, 159–197.
40. D. F. Rogers and N. G. Fog, Constrained B-spline curve and surface fitting, *Computer Aided Geometric Design*, vol. 21, 1989, 641–648.
41. M. Sabin, Non-rectangular surface patches suitable for inclusion in a B-spline surface, *Proc. Eurographics '83*, North-Holland, 1983, 57–69.
42. W. Schroeder, J. Zarge, and W. Lorensen, Decimation of triangle meshes, *Computer Graphics*, vol. 26, no. 2, 1992, 65–70.
43. T. W. Sederberg and S. R. Parry, Free-form deformation of solid geometric models, *Computer Graphics*, vol. 20, no. 4, 1986, 151–160.
44. F. Solina and R. Bajcsy, Recovery of parametric models from range images: The case of superquadrics with global deformations, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 12, no. 2, 1990, 131–147.
45. D. Terzopoulos, A. Witkin, and M. Kass, Constraints on deformable models: Recovering 3D shape and non-rigid motion, *Artificial Intelligence*, vol. 36, 1988, 91–123.
46. D. Terzopoulos and D. Metaxas, Dynamic 3-D models with local and global deformations: deformable superquadrics, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 13, no. 7, 1991, 703–713.
47. G. Turk, Re-tiling polygonal surfaces, *Computer Graphics*, vol. 26, no. 2, 1992, 55–64.



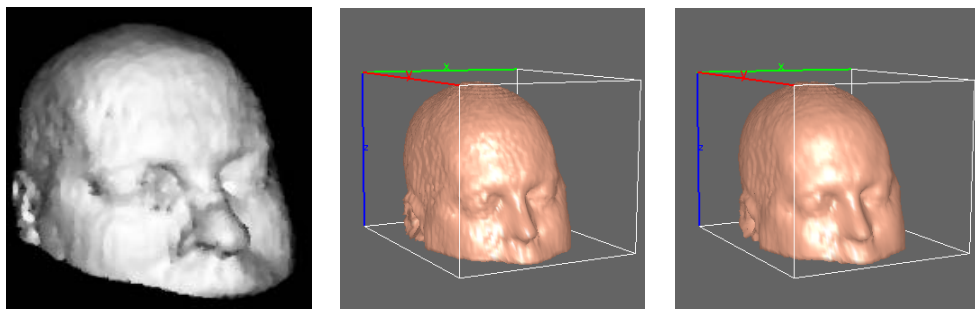
**Figure 2.** Left ventricular blood pool in digital form, obtained by segmenting a cardiac MR image. Middle and right images show the blood pool represented by a RaG surface at two different smoothness levels.



**Figure 3.** Representing a brain tumor with a RaG surface at two different smoothness levels.



**Figure 4.** Representing a brain with a RaG surface at two different smoothness levels.



**Figure 5.** Representing the skin of a head with a RaG surface at two different smoothness levels.