



Approximating Digital 3-D Shapes by Rational Gaussian Surfaces

Marcel Jackowski*, Martin Satter[†], and Ardeshir Goshtasby^{‡§}

Abstract—A method for approximating spherical topology digital shapes by rational Gaussian (RaG) surfaces is presented. Points in a shape are parametrized by approximating the shape with a triangular mesh, determining parameter coordinates at mesh vertices, and finding parameter coordinates at shape points from interpolation of parameter coordinates at mesh vertices. Knowing the locations and parameter coordinates of the shape points, the control points of a RaG surface are determined to approximate the shape with a required accuracy. The process starts from a small set of control points and gradually increases the control points until the error between the surface and the digital shape reduces to a required tolerance. Both triangulation and surface approximation proceed from coarse to fine. Therefore, the method is particularly suitable for multiresolution creation and transmission of digital shapes over the Internet. Application of the proposed method in editing of 3-D shapes is demonstrated.

Index Terms—digital 3-D shape, triangular mesh, multiresolution representation, rational Gaussian (RaG) surface, shape editing

1 Introduction

Tomographic images are widely used in medical applications. These images are commonly represented in volume form by 3-D arrays of numbers. The rendering of volumetric images often reduces to the rendering of objects within them. The objects, which we call digital

*Department of Diagnostic Radiology, Yale University School of Medicine, New Haven, CT

[†]Wallace-Kettering Neuroscience Institute, Kettering Medical Center, Kettering, OH

[‡]Department of Computer Science and Engineering, Wright State University, Dayton, OH

[§]Corresponding author: ardeshir@cs.wright.edu

3-D shapes or digital shapes, are inherently smooth objects that have been discretized in the domain of volumetric images. In this paper, a surface recovery method is described that approximates a digital 3-D shape by a rational Gaussian (RaG) surface. The obtained surface, which is obtained from coarse to fine, enables efficient transmission, rendering, and editing of the shape.

Considerable work has been done extracting and representing isovalued surfaces in volumetric images using triangular meshes [34, 51]. Except for very simple images, however, isovalued surfaces cannot accurately delineate object boundaries and more elaborate methods are needed to isolate objects from each other and from the background. Some of the existing methods use locally maximum intensity variations [7], zero-crossings of the second intensity derivatives [5], intensity creases [33], energy-minimizing or optimization methods [46, 47], rule-based classification [44], fuzzy connectedness theory [42], and pattern recognition approaches [3]. The method proposed here is not limited to isovalued surfaces and can represent any digital shape obtained by any image segmentation method. The only requirement is that the shape have a spherical topology.

Due to the increased need for sharing of medical data over the Internet, it is desired that a representation be in multiresolution and in compact form. The proposed method approximates a digital shape first by a triangular mesh and then by a RaG surface with a required accuracy. A subdivision algorithm is described that positions large triangles in flat areas and small triangles in detailed areas in a shape to achieve a high compression rate. The approximation is achieved progressively, gradually reproducing more shape details. The method creates a shape from coarse to fine and stops when the desired accuracy in approximation is reached.

In the following, in Section 2 related work is reviewed, and in Section 3 terminologies used in the paper are defined. In Section 4, a coarse-to-fine approach to subdivision of a digital shape to a triangular mesh is given, and in Section 5, parameter coordinates at shape points are determined. In Section 6, an algorithm for approximating a digital shape by a

RaG surface with a required accuracy is described, and in Section 7, the characteristics of the proposed surface approximation method are examined. In Section 8, an interactive editing tool for revising a shape represented by a RaG surface is described, and finally in Section 9, concluding remarks are made.

2 Related Work

Digital 3-D shapes have been represented by finite elements, superquadrics, and hyperquadrics. McInerney and Terzopoulos [36] develop a balloon model that extracts and represents anatomic structures in medical images. The model is composed of triangular surface elements that join smoothly while responding to forces derived from shape points. Solina and Bajcsy [45] and Pentland [39] approximate a digital 3-D shape by a superquadric surface using an energy function that depends on the distance of shape points to the surface to be recovered. The surface is then determined by minimizing the energy function. Because of the limited number of free parameters of superquadrics, such surfaces can only reproduce global shape details. Terzopoulos and Metaxas [48] combine membrane splines with superquadrics to create both local and global shape details. To capture local details in a recovered surface, Bardinet *et al.* [1] first fit a superquadric surface to the data and then carry out a free-form deformation of the surface. Cohen and Cohen [10] and Hanson [22] use hyperquadric models with a large number of parameters to represent free-form shapes. The parameters of the hyperquadrics are determined through a nonlinear optimization method. All these methods represent free-form shapes with spherical topology. In addition, they allow local deformation of recovered shapes. We introduce a parametric surface approximation method that achieves the same without using computationally intensive optimization processes.

Various implicit surface methods to recover 3-D shapes among irregularly spaced points have been reported. Turk and O'Brien [49, 50] use thin-plate splines as radial basis functions to fit an interpolating surface to a set of irregularly spaced points. The coefficients of the interpolating spline are determined by solving a sparse system of linear equations. A signed

distance approach to the method is also proposed that produces the surface in discrete form without solving a system of equations. Carr *et al.* [8] also use radial basis functions, in particular thin-plate splines, to recover shapes from irregularly spaced points. In addition, they show how to reduce the number of radial basis function centers without significantly reducing the accuracy of the fit. Morse *et al.* [37] use compactly supported radial basis functions instead of thin-plate splines to do the same. Since compactly supported radial basis functions span over a limited approximation domain as opposed to thin-plate spline bases that cover the entire domain, a very sparse matrix of coefficients is obtained, enabling a more efficient solution of the system of equations.

A large body of work exists approximating digital shapes by triangular meshes. Hoppe *et al.* [23] estimate tangents at the points, create a weighted signed distance at each tangent plane in a volumetric image, and locate the zero-crossings in the image to produce a digital 3-D shape. The digital shape is then approximated by a triangular mesh using a marching cubes algorithm. Hoppe *et al.* [24] later provide a means to simplify a mesh through an optimization process. Hoppe also describes a method to create a multiresolution mesh starting from the finest mesh obtained for the marching cube toward the base mesh [25, 26]. Since the triangular mesh obtained from the marching cube algorithm has more vertices than the voxels in the shape itself, this multiresolution method has very high memory and computational requirements. Recently Wood *et al.* [51] proposed a method for the coarse-to-fine representation of iso-valued surfaces with much smaller computational and memory requirements. Since all these methods find iso-valued surfaces in volumetric images, and as mentioned earlier iso-valued surfaces rarely find true object surfaces in medical images, these methods have limited uses in medical applications. In this paper, a method is proposed that can take in a digital shape irrespective of how it is obtained and approximate the shape by a parametric surface in multiresolution.

Considerable work exists describing meshes in multiresolution. Eck *et al.* [12], through a remeshing process, transform an arbitrary mesh to a mesh with subdivision connectivity in

multiresolution. Zorin *et al.* [52] provide a means to vary the local resolution of a mesh on demand to reveal details in some areas while smoothing details in other areas. Lee *et al.* [30] introduce a multiresolution remeshing method with the objective of smoothly parametrizing the mesh points. Hoppe [26], Khodakovsky *et al.* [28], Cohen-Or *et al.* [11], and Pajarola and Rossignac [38] develop multiresolution approaches that produce highly compressed meshes.

An approach that produces considerable details by using only a coarse mesh is based on the displaced subdivision idea [31]. From a coarse control mesh, a smooth surface is obtained using a subdivision surface scheme [9]. Then, a scalar field representing the displacement between the subdivision surface and the given points in the direction normal to the surface is determined. The scalar field is then used to displace the surface or create a bump map for the surface to imitate local details in the underlying shape. By selecting an appropriate resolution for the mesh, a desired level of detail can be produced in the reconstructed shape. This idea is extended to meshes as opposed to smooth surfaces [21] by letting the displacement fields measure distances of points to the mesh faces rather than to the surface.

We propose a method that starts from a coarse mesh and subdivides the mesh until maximum distance between the given digital shape and the approximating mesh reaches a desired tolerance. The subdivision requires very little computation if a rather coarse mesh is what is required. As more details are demanded, finer meshes are progressively generated with some additional computation time. Computations are carried out on the digital shape directly, therefore, a marching cubes algorithm is not needed to convert a digital shape to a triangular mesh.

3 Definitions

In this section, terminologies used in the paper are described.

- **Shape point:** A voxel in a digital shape. Voxels in isotropic images are cubic. If a volumetric image is not isotropic, it can be made isotropic by an interpolation and resampling process [17]. Shape points will be denoted by \mathbf{p} 's.

- **Digital shape:** The bounding surface of a region in an isotropic volume image with thickness of one voxel. Fig. 1a shows a digital shape.
- **Adjacent points:** Two points \mathbf{p}_i and \mathbf{p}_j are adjacent if $1 \leq \|\mathbf{p}_i - \mathbf{p}_j\| \leq \sqrt{3}$. It is assumed that the length of a side of a voxel represents the unit of measurement. Adjacent points are considered **connected** points.
- **Path:** A path between points \mathbf{p}_i and \mathbf{p}_j is a connected set of points starting from \mathbf{p}_i and ending at \mathbf{p}_j where no point is repeated. A path is also called a **contour**. A path is shown in Fig. 1b.
- **Triangular mesh approximation of a shape:** A triangular mesh that interpolates some points and approximates the rest in a digital shape. The mesh vertices will be denoted by \mathbf{P} 's. Note that the mesh vertices are a subset of the shape points. A triangular mesh approximation of the shape in Fig. 1a is shown in Fig. 1c.
- **Edge contour:** A contour in a digital shape that is delimited by the end points of a mesh edge and lies in the plane passing through the edge and bisecting the angle between the two triangles sharing the edge. Note that due to the digital nature of shape points, some points in an edge contour may not fall exactly in the bisecting plane; however, they will be closer to the plane than points in any other path connecting the edge end points. An edge contour is shown in Fig. 1d.
- **Distance of point \mathbf{p}_i to edge $\mathbf{P}_j\mathbf{P}_k$:** Assuming the plane passing through the point and normal to the edge intersects the edge at \mathbf{P}_l , if \mathbf{P}_l is between \mathbf{P}_j and \mathbf{P}_k , the distance will be $\|\mathbf{p}_i - \mathbf{P}_l\|$. Otherwise, if \mathbf{P}_l is closer to \mathbf{P}_j than to \mathbf{P}_k , the distance will be $\|\mathbf{p}_i - \mathbf{P}_j\|$, and if \mathbf{P}_l is closer to \mathbf{P}_k than to \mathbf{P}_j , the distance will be $\|\mathbf{p}_i - \mathbf{P}_k\|$ (see Fig. 2a).
- **Distance of an edge contour to an edge:** Assuming \mathbf{p}_i is a contour point with distance d_i to the associating edge, we will take the maximum distance from points

on the contour to the edge as the distance of the contour to the edge. That is, $D_e = \max_i\{d_i\}$.

- **Distance of point \mathbf{p}_i to triangular face $\mathbf{P}_j\mathbf{P}_k\mathbf{P}_l$:** Assuming the line passing through \mathbf{p}_i and normal to the triangle intersects the plane of the triangle at \mathbf{P}_m , if \mathbf{P}_m is inside the triangle, the distance will be $\|\mathbf{p}_i - \mathbf{P}_m\|$. If \mathbf{P}_m is outside the triangle and assuming \mathbf{P}_n is the point on the triangle edge closest to \mathbf{P}_m , the distance will be $\|\mathbf{p}_i - \mathbf{P}_n\|$. Point \mathbf{P}_m will be called the **projection** of point \mathbf{p}_i onto the plane of the triangle.
- **Triangular patch:** A connected set of points in a digital shape delimited by three edge contours whose end points are the vertices of a triangle (see Fig. 2b).
- **Distance of a triangular patch to the associating triangle:** Assuming point \mathbf{p}_i belongs to the triangular patch and the distance between \mathbf{p}_i and the triangle is h_i , the distance to be determined is the maximum of such distances when all points in the patch are tested. That is, $D_t = \max_i\{h_i\}$.
- **Geometric axes of a digital shape:** The axes whose directions are the eigenvectors of the inertia matrix [16] of shape points and which pass through the center of gravity of the shape. The axis obtained from the largest eigenvalue is called the **major axis** of the shape.
- **Subdivision of edge $\mathbf{P}_j\mathbf{P}_k$:** Replacing the edge with edges $\mathbf{P}_j\mathbf{P}_i$ and $\mathbf{P}_i\mathbf{P}_k$, where \mathbf{P}_i is the farthest point on the contour associated with the edge and distance of \mathbf{P}_i to the edge is larger than the given tolerance ε_t (see Fig. 2c). Note that the new edges fall between the edge contour and the old edge. Therefore, distances of the new obtained contour segments to the associating edges will be smaller than the distance of the original contour to the associating edge.

4 Subdivision of a Shape to a Triangular Mesh

Using the above definitions, next we describe an algorithm that subdivides a triangle into 2, 3, or 4 smaller triangles. Then, we use this subdivision algorithm to find a triangular mesh that approximates a digital shape with a required accuracy, ε_t .

Algorithm 1: Subdivision of a single triangle

A triangle is subdivided into 2, 3, or 4 smaller triangles according to the following rules:

1. If distances between all three edges of the triangle and the corresponding contours are larger than ε_t , then each edge is subdivided into two and the subdivision points are connected to each other to obtain four smaller triangles (see Fig. 3a). Many subdivision algorithms use only this rule to maintain subdivision connectivity [12, 52].
2. If distances between two of the edges and the corresponding contours are larger than ε_t , those two edges are subdivided and the subdivision points are connected to obtain a triangle and a quadrilateral. The quadrilateral is then divided into two triangles using the shorter of the two diagonal lines (see Fig. 3b).
3. If the distance between only one of the edges and the corresponding contour is larger than the required tolerance, only that edge is subdivided into two and the subdivision point is connected to the opposing triangle vertex to obtain two smaller triangles (see Fig. 3c).
4. If distances between all three edges and the corresponding contours do not reach ε_t , the distance between the patch and the triangle is determined, and if that distance is larger than ε_t , the point in the patch farthest from the triangle is connected to the vertices of the triangle to obtain three smaller triangles (see Fig. 3d).

In this way, a triangle is subdivided into 2, 3, or 4 smaller triangles. Subdivision will take place in the order specified. That is, only if subdivision by case 1 is not possible will

subdivision by case 2 be considered, and subdivision by case 3 will be considered only when subdivision by case 2 is not possible, and so on. This ordering prefers subdivision of edges first, ensuring that the final subdivision will not contain acute triangles. By subdividing an edge contour at the point farthest from the corresponding edge and subdividing a patch at the point farthest from the corresponding triangle, the algorithm ensures that at each subdivision the maximum error between an edge and its contour or a patch and its triangle is reduced.

Algorithm 2: Approximation of a digital shape by a triangular mesh

We first approximate the shape by an octahedron and then subdivide the triangular faces of the octahedron to smaller triangles and repeat the process until maximum error in subdivision reaches the required tolerance, ε_t .

1. **Initialization:** Approximate the shape by an octahedron. This involves placing an octahedron inside the shape and extending its axes until they intersect the shape and replacing the octahedral vertices with the obtained intersection points. The center of the octahedron is placed at the center of gravity of the shape and its axes are aligned with the axes of the shape. If the shape is concave so that the center of gravity of the shape falls outside the shape, the intersection of the major axis of the octahedron with the shape is found and the midpoint of the longest segment of the axis falling inside the shape is taken as the center of the octahedron. The triangular faces of the obtained octahedron are entered into a list. If a base mesh is given instead, the triangles in the base mesh are entered into the list.
2. **Main step:** Remove a triangle from the list. If the distance between triangle edges and the associating edge contours or the distance between the triangle and the associating triangular patch is larger than the required tolerance, subdivide it according to Algorithm 1 and enter the newly obtained triangles into the list.

3. **Stopping criterion:** If the list is empty, stop. Otherwise, go back to Step 2.

The reason for orienting the octahedron so that its axes lie on the axes of the shape is to maximize overlap between the shape and the octahedron, thereby minimizing the distance between the approximating mesh and the shape. A triangle is subdivided according to Algorithm 1. Therefore, if the distance between an edge and the associating edge contour is greater than the required tolerance, that edge is subdivided. Since an edge is shared by two triangles, when it is subdivided, the two triangles that share the edge are both subdivided. This means, if one triangle is subdivided at an edge, the other triangle sharing the edge, if not subdivided already, will be subdivided later. This ensures that when required tolerance is reached, no cracks will remain between adjacent triangles in the obtained mesh. This also implies that the triangles can be processed independent of each other and in parallel. Therefore, the list that keeps the unprocessed triangles can be implemented by a queue or a stack. The stack implementation completes subdivision of an octahedral face before moving to the next. The queue implementation subdivides all octahedral faces in sequence until required accuracy in approximation is reached. Our implementation uses a queue. In Fig. 4, the intermediate results in subdivision of a volumetric brain are shown. In this figure ε_t is 1 voxel; that is, the maximum distance between the digital shape and the approximating mesh is equal to the length of a side of a voxel.

Some of the properties of the proposed subdivision are: **a)** A unique subdivision is obtained for a non-symmetric shape independent of the orientation and position of the shape. This is achieved by aligning the geometric axes of the shape with axes of the octahedron. **b)** The process avoids creation of long triangles. This is ensured by subdividing the edge contours first and then the patches. **c)** Compression rate depends on the complexity of the shape. The algorithm produces a high compression rate by adjusting triangle sizes to local details in a shape. It produces large triangles in flat areas and small triangles in detailed areas in a shape.

Since the focus of the paper is on shapes with spherical topology, an octahedron was

used as the base mesh. The proposed subdivision, however, is not limited to shapes with spherical topology, and it can be applied to shapes with arbitrary topologies if base meshes representing the topologies under consideration are given. When the base mesh for a shape is provided, instead of starting the subdivision with the triangles representing the octahedral faces, we start with the triangles representing the faces in the base mesh.

The proposed subdivision may be used to approximate a digital 3-D shape by a triangular mesh as demonstrated in Fig. 4. Our goal in this subdivision is to only determine parameter coordinates at the shape points. For this reason, the subdivision process can be stopped very early, obtaining a rather coarse mesh. Subdivision needs to continue until foldovers no longer exist in the triangular patches. This is explained in more detail next.

5 Parametrizing the Shape Points

To parametrize a digital shape with spherical topology, an octahedron is used as the base mesh. First, parameter coordinates at octahedral vertices are determined by fitting an octahedron to the sphere and mapping parameter coordinates at the obtained octahedral vertices to corresponding octahedral vertices approximating the shape. After this initial step, the edge contours and triangular patches corresponding to the base mesh are determined and points in each patch are identified. Since parameter coordinates of points in a triangle can be determined through barycentric coordinates [27] of parameters at vertices of the triangle, parameter coordinates of points in a triangular patch can be determined by projecting them to the associating triangle and assigning parameter coordinates of the triangle points to the corresponding patch points.

An example of a shape that has a spherical topology is shown in Fig. 5. The figure shows the result of subdividing the shape by Algorithms 1 and 2. If one were to stop very early in the subdivision, foldovers would exist in the obtained triangular patches. However, if subdivision is continued, the subdivision will propagate to the shape branches after a number of iterations and the obtained triangles will get close enough to the patches to eliminate the

foldovers. Also note that because parametrization of shape points needs to be determined only after triangulation of a shape is complete, if the process is not stopped very early, the triangular patches obtained by the subdivision will be close to the mesh and the patch points can be parametrized by projecting them to the associating triangles and assigning parameters at triangle points to corresponding patch points.

The triangular patch associated with a triangle is delimited by three edge contours. When a triangle is subdivided at an edge contour, the farthest point on the contour to the associating edge is used to break the contour. Therefore, each subdivision step reduces approximation error between the edge contour and the associating edge. If the distance between all edge contours and the corresponding edges in a triangle are below the required tolerance, but the distance of the triangle to its patch is larger than the tolerance, the patch is subdivided into three smaller patches. The subdivision process again takes the point on the patch that is farthest from the triangle to subdivide the patch. In each subdivision, the newly obtained triangles get closer to the patch than the original triangle. The process stops when distances between the patches and the associating triangles reach the required tolerance.

Due to the digital nature of the given shapes, if the chosen error tolerance is less than two voxels, all foldovers are guaranteed to disappear from the obtained edge contour and triangular patches. This is because to obtain a foldover, a contour or a patch has to extend in one direction and come back, and because the thickness of the digital contour or patch is one voxel, a minimum of 2 voxels is needed to produce a foldover. Figure 6a shows an edge contour and its associating edge. This represents a clear foldover because when the contour points are projected to the edge, it is not possible to assign monotonically increasing or monotonically decreasing parameters to the contour points. For instance, points **A**, **B**, and **C** project to the same edge point. If the parameter at **Q** is smaller than that at **P**, in order for points **A**, **B**, and **C** to have the same parameter, it is required that as one travels along the contour from **Q** to **P**, the parameter first increase, then decrease, and then increase

again. This means that monotonically increasing parameters cannot be assigned to points in contour **PQ** and in order to assign monotonically increasing parameters to points in this contour, it is required to subdivide the contour further until all foldovers are eliminated. The distance of the edge contour in Fig. 6a to its edge is 5 voxels.

Figure 6b shows another edge contour containing a foldover whose distance to its edge is 2 voxels. A foldover exists because point **B** projects to point **Q**, picking the same parameter value that point **Q** has. This means that the parameter values along the contour should increase from **Q** to **A** and then they should decrease from **A** to **B**. Since it is required that parameters from one end of an edge contour to the other monotonically increase or decrease, this contour needs to be subdivided further before it can be monotonically parametrized.

Because of the digital nature of voxels, it is impossible for a contour with a distance of less than 2 voxels to its edge to contain a foldover, as demonstrated in two contours **PQ** and **RS** in Fig. 6c. Voxels in such a contour either lie on the edge or at distances smaller than 2 voxels from the edge. Note that a contour like the one shown in Fig. 6d is not allowed because it violates the 1-voxel thickness assumption of digital shapes. Voxel **A** in Fig. 6d causes the shape to have a thickness of 2 voxels at **A**. Therefore, to enable monotonic parametrization of points in an edge contour, the error tolerance used in the subdivision should be less than 2 voxels. The same argument holds for a triangular patch and its associating triangle if we treat the voxels in the patch falling in a plane normal to the triangle as the contour and the intersection of the plane with the triangle as the associating edge.

If foldovers do not exist in a patch, points in the patch can be parametrized by projecting them to the associating triangle and assigning the parameter coordinates at triangle points to the voxels projecting to them. Parameter coordinates inside a triangle are determined from barycentric coordinates of parameters at vertices of the triangle.

The proposed method assigns monotonic parameter coordinates to voxels in a digital shape for surface fitting, but the parameters may not be smooth. To obtain a smooth parametrization, the parameters obtained here should be used as initial values in one of

the existing smooth parametrization methods. Barhak and Fischer [2] describe methods based on neural networks and partial differential equations that smoothly parametrize a set of points starting from an initial parametrization. A method described by Kobbelt *et al.* [29] first grows the given shape towards its convex hull. It then projects the convex hull to its bounding sphere. Finally, it assigns parameter coordinates of points in the sphere to corresponding points in the shape to obtain smooth parameter coordinates. This resembles a shrink-wrapping process in which a sphere is shrunk to a given shape. An energy minimizing model is used to minimize local distortions and generate a mesh with subdivision connectivity. In a method described by Floater and Reimers [14, 15] the shape points are mapped to a planar parameter domain by solving a linear system of equations obtained by requiring that the parameter coordinates at each point be a convex combination of the parameter coordinates of points in its neighborhood. Another parametrization method worthy of mention is the constraint optimization of Brechbühler *et al.* [6], which obtains smooth parameter coordinates from a heat conduction model.

To obtain a parametric surface that approximates a digital shape it is required that the parameter coordinates assigned to shape points monotonically increase or decrease as one moves from one end of the shape to another. If a texture is to be mapped to the shape, the assigned parameters should also be smooth. If a smooth parametrization is required, the parameters obtained by the proposed method should be used as initial values to one of the parametrization methods mentioned above to obtain a monotonic and smooth mapping between the shape voxels and the parameter (texture) coordinates.

6 Parametric Representation of a Digital Shape

Knowing the positions and parameter coordinates of points in a digital shape, we approximate the shape by a smooth parametric surface. If the shape to be constructed is inherently smooth, a parametric surface is more suitable to represent it than a triangular mesh. Various parametric surface methods have been developed for representing shapes created by meshes.

They include: subdivision methods [4, 9], triangular patches [13, 20, 41], generalized B-splines [32], triangular NURBS [43], and wavelets [35]. Among these methods, wavelets provide an efficient means for reconstructing a shape at multiresolution. Such representations are becoming increasingly important because of the need for transmission of geometric data over the Internet. In the following, a multiresolution approximation to a digital shape using a RaG surface [18, 19] is described.

If $\{\mathbf{p}_i : i = 1, \dots, n\}$ are the control points and $\{(u_i, v_i) : i = 1, \dots, n\}$ are the parameter coordinates associated with the points, a RaG surface that approximates the points is given by [18, 19],

$$\mathbf{p}(u, v) = \sum_{i=1}^n \mathbf{p}_i g_i(u, v), \quad (1)$$

where $g_i(u, v)$ is the i th basis function of the surface defined by

$$g_i(u, v) = \frac{G_i(u, v)}{\sum_{j=1}^N G_j(u, v)}, \quad (2)$$

and

$$G_i(u, v) = \exp\{ -[(u - u_i)^2 + (v - v_i)^2] / 2\sigma^2 \} \quad (3)$$

is a 2-D Gaussian of height 1 centered at (u_i, v_i) in the parameter space. Formulas (1)–(3) are for an open surface. If the surface is required to close from one side, like a generalized cylinder, $G_i(u, v)$ should be replaced with

$$G_i(u, v) = \sum_{k=-\infty}^{\infty} \exp\{ -[(u - u_i)^2 + (v - v_i + k)^2] / 2\sigma^2 \}. \quad (4)$$

If the opening at each end of the cylinder is represented by a single point, the cylinder will close and will represent a closed shape. The surface will be smooth everywhere except at the two ends. At the two end points the surface will be only continuous. The ∞ in formula (4) is because of the fact that a 2-D Gaussian wraps around the closed side of the surface infinitely. In practice, however, since a Gaussian approaches zero exponentially, its effect vanishes after one or two cycles. Therefore, in practice, the ∞ in formula (4) is replaced by 1 or 2 [18].

The mesh vertices obtained by the triangulation process can be used as the control points. Therefore, by knowing the parameter coordinates at the mesh vertices, a RaG surface can be approximated to the points. This, however, will not provide the required accuracy in approximation. To achieve the required accuracy, we determine the control points of the RaG surface according to the algorithm below. Note that this algorithm does not solve a system of equations in order to obtain the control points; rather, it selects the shape points as the control points in a progressive manner. This representation facilitates shape editing because the control points of the surface are very close to the digital shape being approximated. To determine the distance between shape point \mathbf{p}_i and the approximating RaG surface, the point on the surface with parameter coordinates (u_i, v_i) , $\mathbf{p}(u_i, v_i)$, is computed and its distance to the shape is determined: $\|\mathbf{p}(u_i, v_i) - \mathbf{p}_i\|$. Then, the maximum distance between corresponding points in the shape and the surface is used as the approximation error: $D_p = \max_i \|\mathbf{p}(u_i, v_i) - \mathbf{p}_i\|$. This error is used as the stopping criterion in the following coarse-to-fine algorithm.

Algorithm 3: Approximation of a digital shape by a RaG surface

This algorithm approximates a digital shape of spherical topology by a RaG surface with error tolerance ε_p .

1. **Initialization:** Let the control points of the RaG surface be the vertices of the octahedron that approximates the shape. Recall that these vertices are a subset of the shape points.
2. **Surface approximation:** Knowing the parameter coordinates associated with the control points, find the RaG surface that approximates the points.
3. **Stopping criterion:** Determine the maximum distance between the shape points and the RaG surface: D_p . If $D_p < \varepsilon_p$, stop. Otherwise, find points with locally maximum error that are larger than ε_p and add those points to the set of control points of the

surface and go back to Step 2.

Shape points with locally maximum errors are selected in each step of the algorithm to ensure that maximum error is reduced in each iteration. A control point selected in this manner will reduce the error not only at that point but also at points in its vicinity. At each iteration, more than one point may be selected. Therefore, the process converges in a few to several iterations. Figure 7 shows intermediate results of this algorithm when approximating the left ventricular cavity by a RaG surface. Note that processing is from coarse to fine. By each iteration of the algorithm, a finer surface with a smaller error is obtained that approximates the shape.

The standard deviation of Gaussians used in a RaG surface determines the level of details reproduced in the surface. Smaller standard deviations reproduce more details, while larger standard deviations smooth more details. The standard deviation of Gaussians selected should relate to the density of the control points. If control points are sparsely spaced, the standard deviation of Gaussians should be large, while if a dense set of control points is given, the standard deviation should be small. Experimental results show that a standard deviation that is equal to the average parameter distance between adjacent points produces surfaces that resemble cubic B-splines [40]. Very small standard deviations produce sharp corners and edges, while very large standard deviations unnecessarily smooth the shape details. Two examples showing the effect of standard deviation of Gaussians in generated shapes are given in Fig. 8.

7 Results

The digital shapes depicted in Fig. 9 show a human liver obtained by segmenting a CT chest image, a brain tumor obtained by segmenting an MR brain image, a spleen and a kidney also obtained by segmenting a CT chest image, and a left ventricular cavity obtained by segmenting a cardiac MR image. RaG surfaces approximating these data sets using Algorithm 3 with error tolerance of 1 voxel are shown in Figs. 10a–e. By error tolerance of

1 voxel, it is meant that the maximum distance between points in the digital shape and the approximating RaG surface is equal to the length of a side of a voxel.

Figure 11a shows the relation between error in surface approximation and the number of iterations. For the shapes shown in Fig. 9, from 5 to 7 iterations were sufficient to obtain an approximation error of 1 voxel. In the first few iterations, approximation error decreases sharply, but when approximation error reaches voxel level, it reduces only slightly in each iteration. This can be attributed to the fact that after four or five iterations, the approximating surface gets very close to the digital shape and the difference between the two reflects the digital nature of the shape. Further iterations will only cause the surface to reproduce digital details in the shape and is not recommended.

Figure 11b shows the compression rate (the number of voxels in a shape divided by the number of control points in the approximating surface) as a function of the number of iterations. The liver data set has produced the highest compression rate because it has large, flat segments that can be reproduced with a small number of control points. Only 1/8th of the total 23,957 voxels were sufficient to create a surface approximating the liver with 1 voxel accuracy.

The approximation error as a function of compression rate is depicted in Fig. 12a for the shapes in Fig. 9. As can be observed, approximation error is proportional to compression rate. To reduce approximation error, an increasingly large number of control points is needed. As error in approximation increases, compression rate increases almost exponentially. The approximation error as a function of standard deviation of Gaussians is depicted in Fig. 12b after 6 iterations in all cases. The number of control points obtained in an approximation is not a mere function of the error tolerance but is a function of the required smoothness of the surface, as well. We see that as the standard deviation of Gaussians increases from a very small value, the error initially decreases up to a point and then it increases. This is because when the standard deviation of Gaussians is very small, the approximating surface passes very close to the triangular mesh producing flat areas as well as sharp edges and corners. As

the standard deviation of Gaussians increases, the obtained surface becomes smoother and more closely resembles the given shape. When standard deviation of Gaussians increases further, details in a reconstructed surface start to disappear and it becomes less similar to the given shape, thus, increasing the error again.

The computational complexity of the subdivision algorithm is initially proportional to the number of triangles in the mesh. After a few iterations, triangles in smooth areas of a shape stop being subdivided and only triangles in detailed areas are subdivided. The process stops when error between the triangles and the associating patches reach the given tolerance. During the first few iterations, the computational complexity of the subdivision algorithm increases sharply, then it flattens, and after a few iterations, it decreases sharply. The computational complexity of the surface approximation algorithm in an iteration is proportional to the number of new control points obtained. Initially, the number of newly obtained control points increases sharply and after a few iterations new control points are obtained only at detailed areas in a shape. Finally, the number of new control points found in an iteration decreases sharply before the algorithm converges. For the shapes shown in Fig. 9, from 10 to 30 seconds were needed to obtain the triangular meshes with 1 voxel error tolerance, and from 40 to 60 seconds were needed to obtain RaG surfaces also with 1 voxel error tolerance on an SGI Octane computer with one 250 MHz IP30 (CPU: R10000, FPU: R10010) processor and 128 MB RAM.

The optimal standard deviation of Gaussians for approximation of one shape may be different from the optimal standard deviation of Gaussians for approximation of another shape. This optimal standard deviation can be determined by a steepest descent algorithm. Since the standard deviation of Gaussians determines the level of detail in a shape, when a coarse shape is to be reproduced from a small number of control points, a larger standard deviation is needed compared to when a shape in a fine resolution is to be reproduced from a large number of control points. Experimental results on digital shapes in medical images show that standard deviations from the average parameter distance between adjacent points

up to three times that much reproduce a moderate amount of details. To obtain the optimal standard deviation for a shape, starting from an initial guess, a steepest descent algorithm may be used to locate the minimum error by iteratively increasing or decreasing the standard deviation. Since our ultimate goal in surface approximation is surface editing, at the time of editing we interactively choose the standard deviation of Gaussians as needed to reproduce a desired amount of details in the constructed shape.

8 Application: Shape Editing

The proposed surface representation lends itself to easy editing of recovered shapes. The process is demonstrated in Fig. 13. Suppose the brain tumor shown in Fig. 9b has been obtained by an automatic segmentation method, which contains some errors. To correct the errors, the tumor in digital form is first subdivided into a triangular mesh and the shape points are parametrized. Then, a RaG surface is approximated to the points with 1 voxel accuracy. This surface approximation is shown in Fig. 10b. Displaying the obtained surface within the original volumetric image, we obtain Fig. 13a, shown in four windows. The upper left window shows the surface representation of the tumor in 3-D. The other three windows show the axial, sagittal, and coronal cross-sections of the volumetric image containing the surface. The small dots show the control points of the approximating surface.

To revise the surface representing the tumor, a small sphere attached to the cursor is used to select and displace some of the control points (see Fig. 13b). The radius of the sphere can be interactively changed to select a desired number of control points. The control points falling in the sphere are displaced with the motion of the mouse. The center of the sphere is kept in the selected axial, sagittal, or coronal slice. As the mouse is moved, the sphere center is displaced in the selected slice. When revising the surface by moving the mouse in a slice, the effect of the revision is displayed in all four windows. The user, by interacting with any of the four windows, as needed, will pull or push the surface locally in any one of the three orthogonal directions to revise the surface.

When the center of the sphere is moved in one of the orthogonal slices, not all of the control points selected by the sphere are moved by the same amount. Also, not all of the control points are moved in the same direction. Rather, the points are moved in the direction obtained by connecting them to the center of the sphere and by the amount proportional to the cosine of the angle between that direction and the direction of motion. Control points falling in one hemisphere produce a positive motion while control points falling in the opposing hemisphere, if any, produce a negative motion due to the negative sign of the cosine. Only control points with positive motion are displaced. While viewing the revised surface overlaid with the volumetric image in real time, the user modifies the surface until the desired shape is sculpted.

9 Concluding Remarks

Increased use of volumetric images in medical applications has prompted a critical need for methods that can represent and manipulate shapes in these images. Since digital shapes contain digital errors as well as scanner noise and segmentation inaccuracies, methods that can smooth the noise and correct the inaccuracies are of special interest. In this paper, a method for approximating digital shapes by RaG surfaces was presented. The method first parametrizes the shape points using a triangulation algorithm. It then finds the control points of an approximating RaG surface through an iterative surface approximation algorithm.

Both triangulation and surface approximation algorithms proceed from coarse to fine. The triangulation uses an octahedron as the base mesh and subdivides it until the error between the mesh and the shape reaches a required tolerance. Surface approximation is achieved from coarse to fine, gradually reducing the error between the given digital shape and the approximating surface. These algorithms subdivide a digital shape to a triangular mesh in coarse to fine and approximate a digital shape by a parametric surface in coarse to fine.

Since a parametric surface representation of a digital shape has far fewer control points

than the points in the digital shape, the parametric representation enables efficient transmission and editing of the shape. Moreover, since approximation is achieved from coarse to fine by gradually creating a more accurate representation, transmission can be achieved very quickly at a coarse resolution, and as more details are demanded, more accurate approximations can be generated and transmitted.

Acknowledgements

This work was supported in part by the Air Force Research Laboratory (AFRL/HEOP) Air Force Material Command, USAF, under cooperative agreement F33615-98-2-6002 and by the Wallace-Kettering Neuroscience Institute. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Data for this work were provided by Martin Satter, Kettering Medical Center, Kettering, Ohio, and by Georg Gotschuli and Erich Sorantin, University of Graz, Austria. These contributions are greatly appreciated. We also would like to thank the anonymous reviewers for their insightful comments and suggestions.

References

- [1] 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.
- [2] J. Barhak and A. Fischer, Parametrization and reconstruction from 3-D points based on neural network and PDE techniques, *IEEE Trans. Visualization and Computer Graphics*, vol. 7, no. 1, 2001, 1–16.
- [3] J. Bezdek, L. Hall, and L. Clarke, Review of MR image segmentation techniques using pattern recognition, *Med. Phys.*, vol. 20, 1993, 233–260.

- [4] W. Boehm and G. Farin, Concerning subdivision of Bézier triangles, *Computer Aided Design*, vol. 15, no. 5, 1983, 260–261.
- [5] M. Bomans, K-H Höhne, U. Tiede, and M. Riemer, 3-D Segmentation of MR images of the head for 3-D display, *IEEE Trans. Medical Imaging*, vol. 9, no. 2, 1990, 177–183.
- [6] 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.
- [7] M. Brejl and M. Sonka, Directional 3-D edge detection in anisotropic data: detector design and performance assessment, *Computer Vision and Image Understanding*, vol. 77, no. 2, 2000, 84–110.
- [8] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans, Reconstruction and representation of 3-D objects with radial basis functions, *Proc. SIGGRAPH*, 2001, 67–76.
- [9] E. Catmull and J. Clark, Recursively generated B-spline surfaces on arbitrary topological meshes, *Computer Aided Design*, vol. 10, 1978, 350–355.
- [10] 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.
- [11] D. Cohen-Or, D. Levin, and O. Remez, Progressive compression of arbitrary triangular meshes, *Proc. IEEE Conf. Visualization*, Oct. 25–29, 1999, 67–72.
- [12] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle, Multiresolution analysis of arbitrary meshes, *Proc. SIGGRAPH*, 1995, 173–182.
- [13] G. Farin, Triangular Bernstein-Bézier patches, *Computer Aided Geometric Design*, vol. 3, no. 2, 1986, 83–128.

- [14] M. S. Floater, Parametrization and smooth approximation of surface triangulations, *Computer Aided Geometric Design*, vol. 14, 1997, 231–250.
- [15] M. S. Floater and M. Reimers, Meshless parametrization and surface reconstruction, *Computer Aided Geometric Design*, vol. 18, 2001, 77–92.
- [16] J. M. Galvez and M. Canton, Normalization and shape recognition of three-dimensional objects by 3-D moments, *Pattern Recognition*, vol. 26, no. 5, 1993, 667–682.
- [17] A. Goshtasby, D. Turner, and L. Ackerman, Matching of tomographic image slices for interpolation, *IEEE Trans. Medical Imaging*, vol. 11, no. 4, 1992, 507–516.
- [18] 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.
- [19] A. Goshtasby, Geometric modeling using rational Gaussian curves and surfaces, *Computer Aided Design*, vol. 27, no. 5, 1995, 363–375.
- [20] J. Gregory and P. Charrot, A C^1 triangular interpolation patch for computer-aided geometric design, *Computer Graphics and Image Processing*, vol. 13, no. 1, 1980, 80–87.
- [21] I. Guskov, K. Vidimce, W. Sweldens, and P. Schroder, Normal meshes, *Proc. SIGGRAPH*, 2000, 95–102.
- [22] A. J. Hanson, Hyperquadrics: Smooth deformable shapes with convex polyhedral bounds, *Computer Vision, Graphics, and Image Processing*, vol. 44, 1988, 191–210.
- [23] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, Surface reconstruction from unorganized points, *Computer Graphics*, vol. 26, no. 2, 1992, 71–78.
- [24] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, Mesh optimization, *Proc. SIGGRAPH*, 1993, 19–26.
- [25] H. Hoppe, Progressive meshes, *Proc. SIGGRAPH*, 1996, 99–108.

- [26] H. Hoppe, Efficient implementation of progressive meshes, *Comput. & Graphics*, vol. 22, no. 1, 1998, 27–36.
- [27] J. Hoschek and D. Lasser, *Computer Aided Geometric Design*, A. K. Peters, 1989, 289–291.
- [28] A. Khodakovsky, P. Schröder, and W. Sweldens, Progressive geometry compression, *Proc. SIGGRAPH*, 2000, 271–278.
- [29] L. P. Kobbelt, J. Vorsatz, U. Labsik, and H-P Seidel, A shrink wrapping approach to remeshing polygonal surfaces, *EUROGRAPHICS*, vol. 18, no. 3, 1999.
- [30] 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.
- [31] A. Lee, H. Moreton, and H. Hoppe, Displaced subdivision surfaces, *Proc. SIGGRAPH*, 2000, 85–94.
- [32] C. Loop and T. DeRose, Generalized B-spline surfaces of arbitrary topology, *Computer Graphics*, vol. 24, no. 4, 1990, 347–356.
- [33] A. M. López, David Lloret, J. Serrat, J. J. Villanueva, Multilevel creases based on the level-set extrinsic curvature, *Computer and Image Understanding*, vol. 77, no. 2, 2000, 111–144.
- [34] W. Lorensen and H. Cline, Marching cubes: A high resolution 3-D surface construction algorithm, *Computer Graphics*, vol. 21, 1987, 163–169.
- [35] M. Lounsbery, T. D. DeRose, and J. Warren, Multiresolution analysis for surfaces of arbitrary topological type, *ACM Trans. Graphics*, vol. 16, no. 1, 1997, 34–73.

- [36] T. McInerney and D. Terzopoulos, A dynamic finite element surface model for segmentation and tracking in multidimensional medical images with application to cardiac 4D image analysis, *Computerized Medical Imaging and Graphics*, vol. 19, no. 1, 1995, 69–83.
- [37] B. S. Morse, T. S. Yoo, P. Rheingans, D. T. Chen, and K. R. Subramanian, *Shape Modeling International*, 2001, 1–12.
- [38] R. Pajarola and J. Rossignac, Compressed progressive meshes, *IEEE Trans. Visualization and Computer Graphics*, vol. 6, no. 1, 2000, 79–93.
- [39] A. Pentland, Automatic extraction of deformable parts models, *Int'l J. Computer Vision*, 1990, 107–126.
- [40] J. Peters, Constructing C^1 surfaces of arbitrary topology using biquadratic and bicubic splines, in *Designing Fair Curves and Surfaces*, N. Sapidis (ed.), 1994, 277–293.
- [41] M. Powell and M. Sabin, Piecewise quadratic approximation on triangles, *ACM Trans. Mathematical Software*, vol. 3, 1997, 316–325.
- [42] S. Punam, J. K. Udupa, and O. Dewey, Scale-based fuzzy connected image segmentation: Theory, algorithms, and validation, *Computer Vision and Image Understanding*, vol. 77, no. 2, 2000, 145–174.
- [43] H. Qin and D. Terzopoulos, Triangular NURBS and their dynamic generalization, *Computer Aided Geometric Design*, vol. 14, 1997, 325–347.
- [44] S. P. Raya, Low-level segmentation of 3-D magnetic resonance brain images—A rule-based system, *IEEE Trans. Medical Imaging*, vol. 9, no. 3, 1990, 327–337.
- [45] 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.

- [46] L. H. Staib and J. S. Duncan, Deformable Fourier models for surface finding in 3-D images, *Proc. SPIE*, vol. 1808, 1992, 90–104.
- [47] H. Tek and B. Kimia, Volumetric segmentation of medical images by three-dimensional bubbles, *Computer Vision and Image Understanding*, vol. 65, no. 2, 1997, 246–258.
- [48] 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.
- [49] G. Turk and J. F. O’Brien, *Variational Implicit Surfaces*, Georgia Tech. Report No. GIT-GVU-99-15, 1999.
- [50] G. Turk and J. F. O’Brien, Shape transformation using variational implicit functions, *Proc. SIGGRAPH*, 1999, 335–342.
- [51] Z. J. Wood, M. Desburn, P. Schröder, and D. Breen, Semi-regular mesh extraction from volumes, *Proc. IEEE Visualization*, 2000, 275–282.
- [52] D. Zorin, P. Schroder, and W. Sweldens, Interactive multiresolution mesh editing, *Proc. SIGGRAPH*, 1997, 259–268.

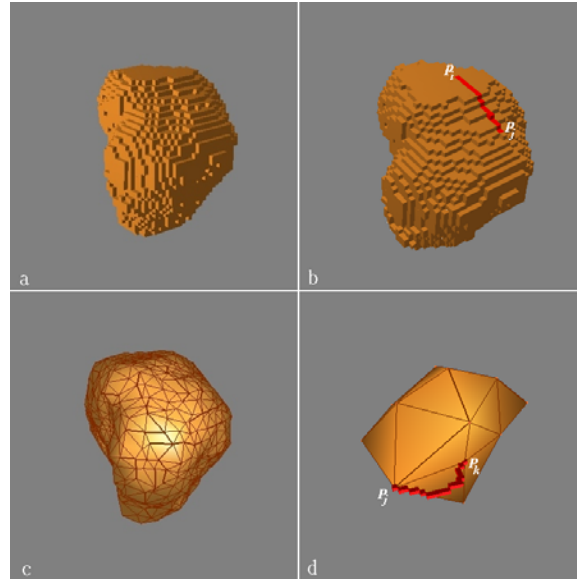


Fig. 1. (a) A digital shape. (b) A path connecting points \mathbf{p}_i and \mathbf{p}_j . (c) A triangular mesh approximation of the shape in (a). (d) An edge contour.

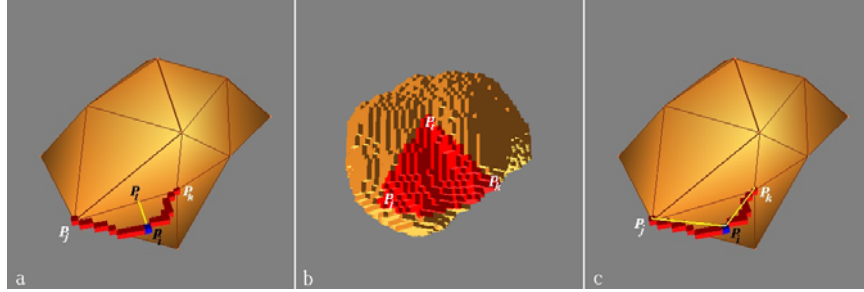


Fig. 2. (a) Distance between point \mathbf{p}_i and edge $\mathbf{P}_j\mathbf{P}_k$. (b) A triangular patch. (c) Subdivision of edge $\mathbf{P}_j\mathbf{P}_k$.

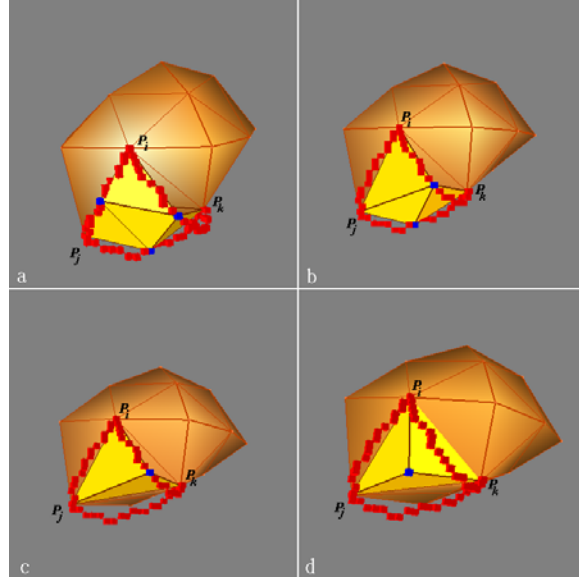


Fig. 3. (a) Four new triangles are obtained by subdividing all three edges of a triangle. (b) Three new triangles are obtained by subdividing only two edges of a triangle. (c) Two new triangles are obtained by subdividing only one of the triangle edges. (d) Three smaller triangles are obtained by connecting the triangle vertices to the point in the patch that is farthest from the triangle.

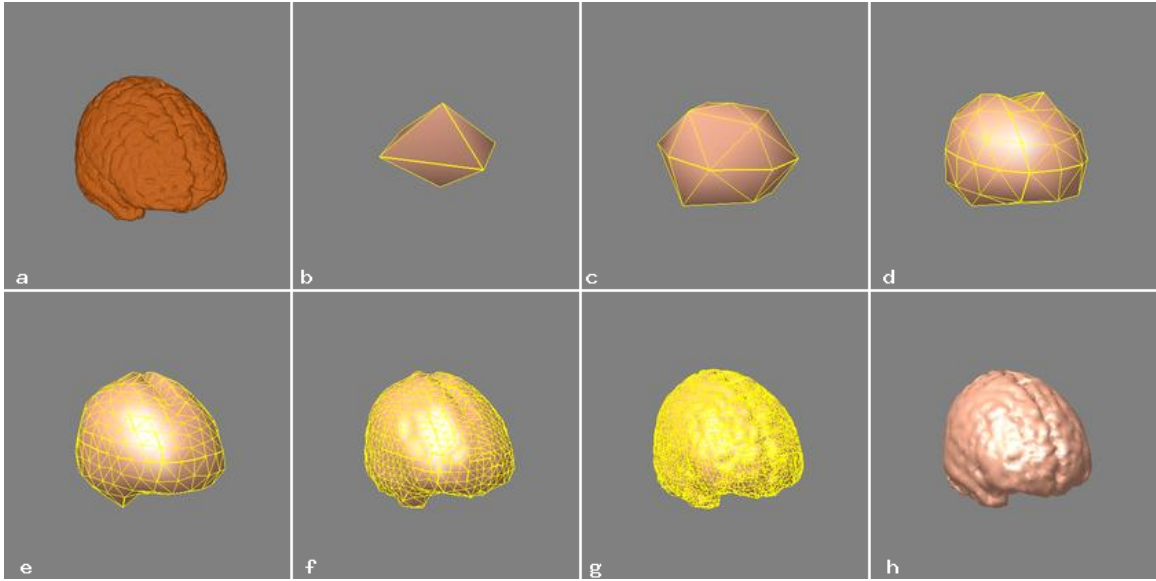


Fig. 4. (a) A digital volumetric brain. (b) Initial approximation of the brain by an octahedron. (c)–(g) Intermediate subdivision steps. (h) Brain shown by the final mesh in shaded form. ε_t is 1 voxel.

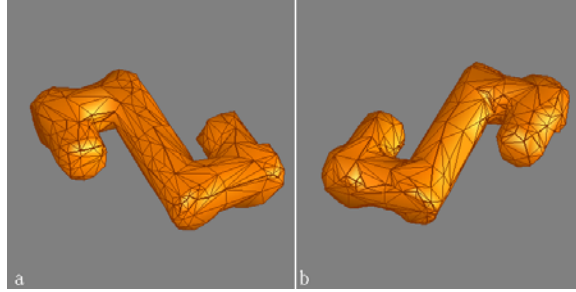


Fig. 5. Two views of the triangulation of an S-like shape produced by the proposed subdivision algorithm starting from an octahedral base mesh.

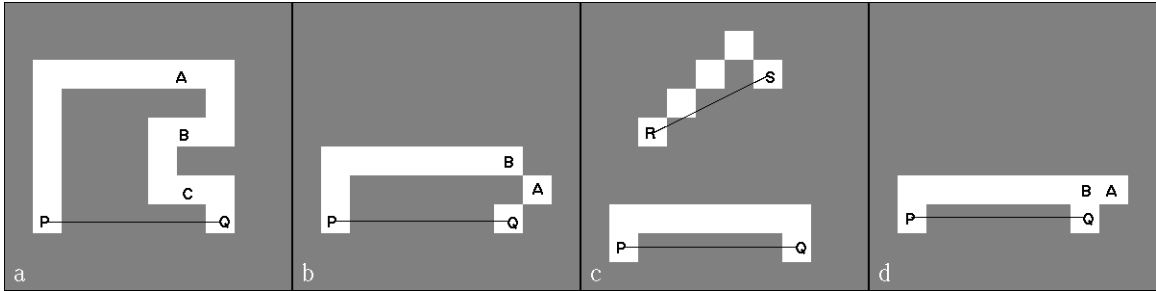


Fig. 6. (a) A contour with distance 5 voxels to the associating edge with a foldover. (b) A contour with distance of 2 voxels to its edge with a foldover. (c) Two contours free of foldovers. Distances of these contours to the associating edges are less than 2 voxels. (d) An illegal edge contour.

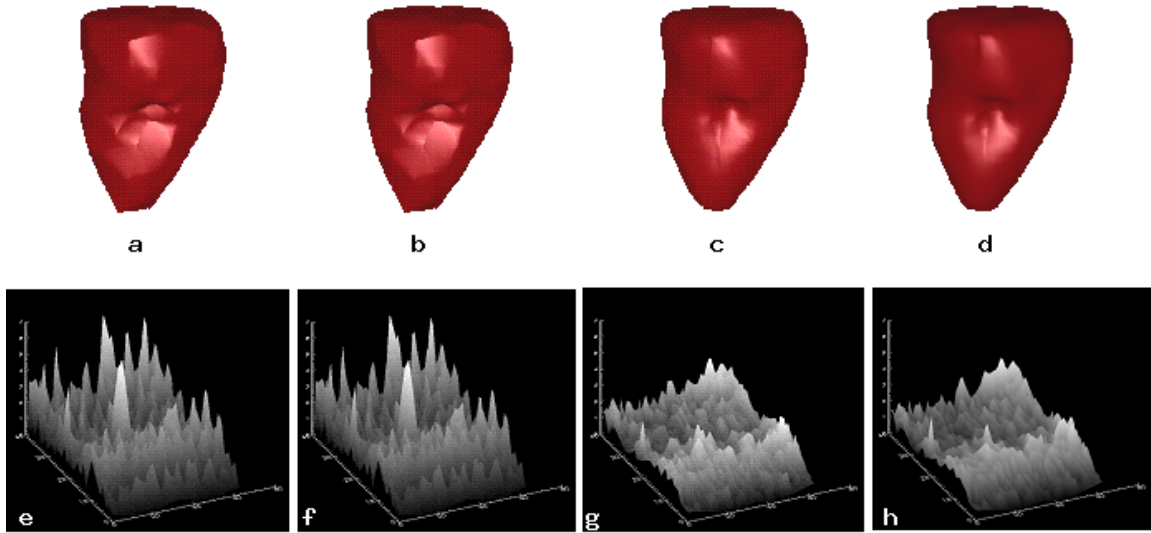


Fig. 7. Approximation of the left ventricular cavity by a RaG surface. (a)–(d) Progressively more accurate approximations to the cavity are obtained by RaG surfaces with 238, 421, 609, and 780 control points. The cavity in digital form contains 4554 voxels. (e)–(h) Errors between the digital shape and the approximating surface plotted in the uv space.

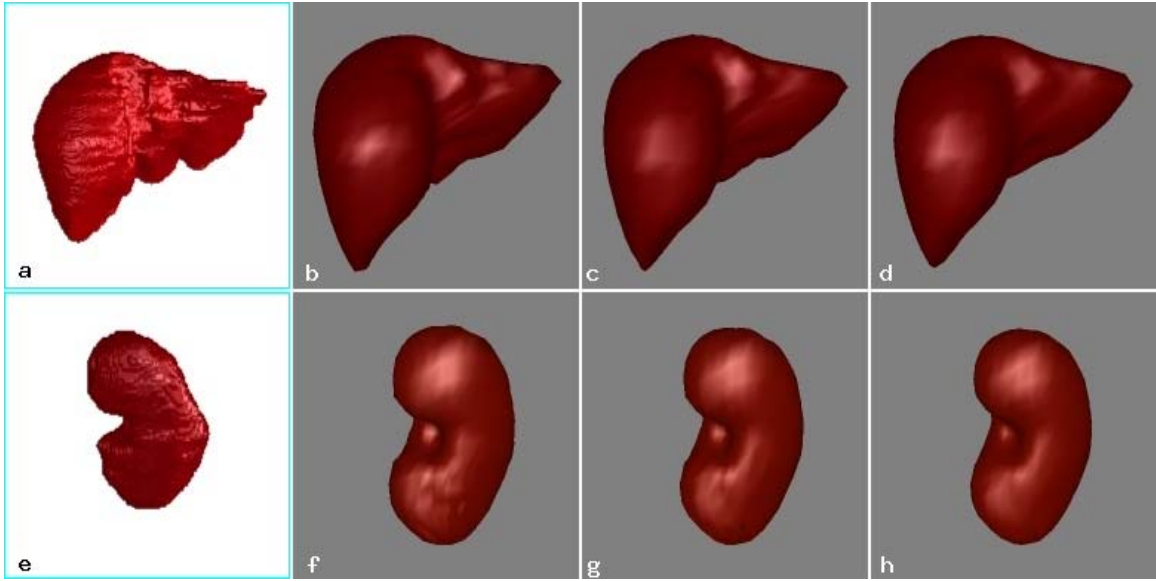


Fig. 8. Rows 1 and 2 show approximations of a liver and a kidney by RaG surfaces, respectively. The first column shows the shapes in digital form. The standard deviation of Gaussians increases from left to right in columns 2–4.

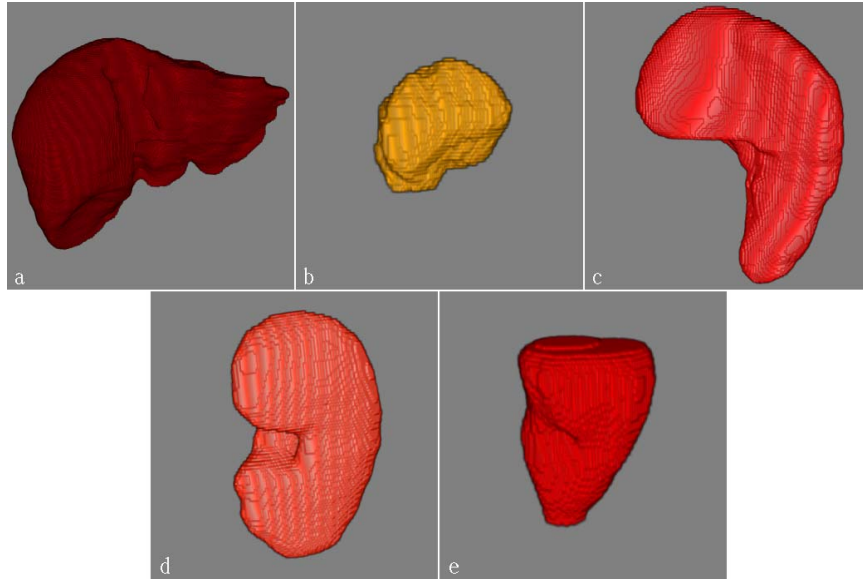


Fig. 9. Digital 3-D shapes. (a) Liver: 53,267 voxels; (b) Brain tumor: 3,012 voxels; (c) Spleen: 23,957 voxels; (d) Kidney: 10,376 voxels; (e) Left ventricular cavity: 4,554 voxels.

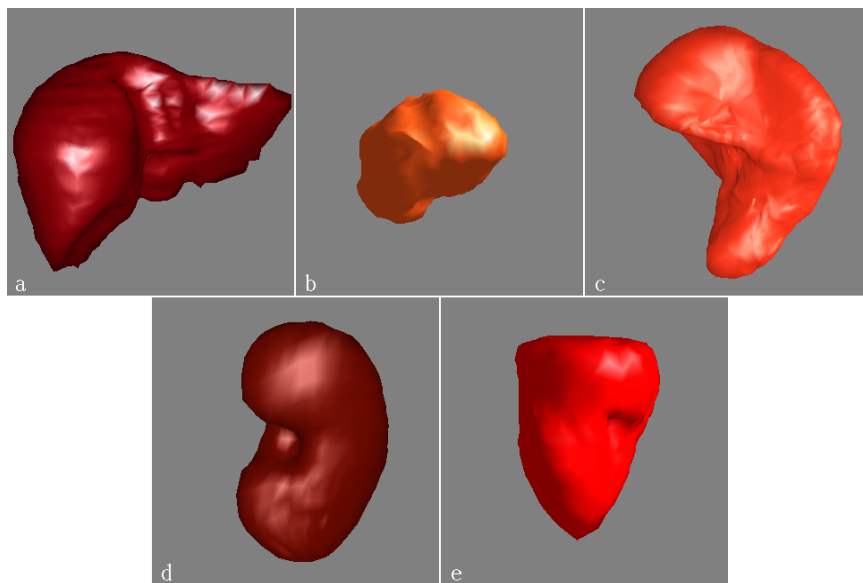
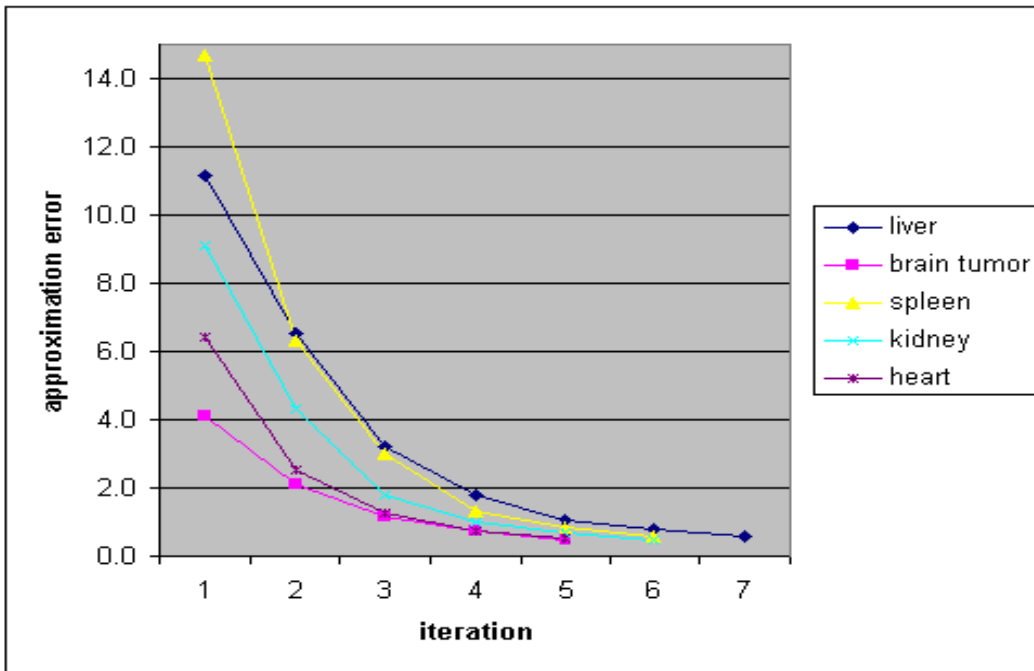
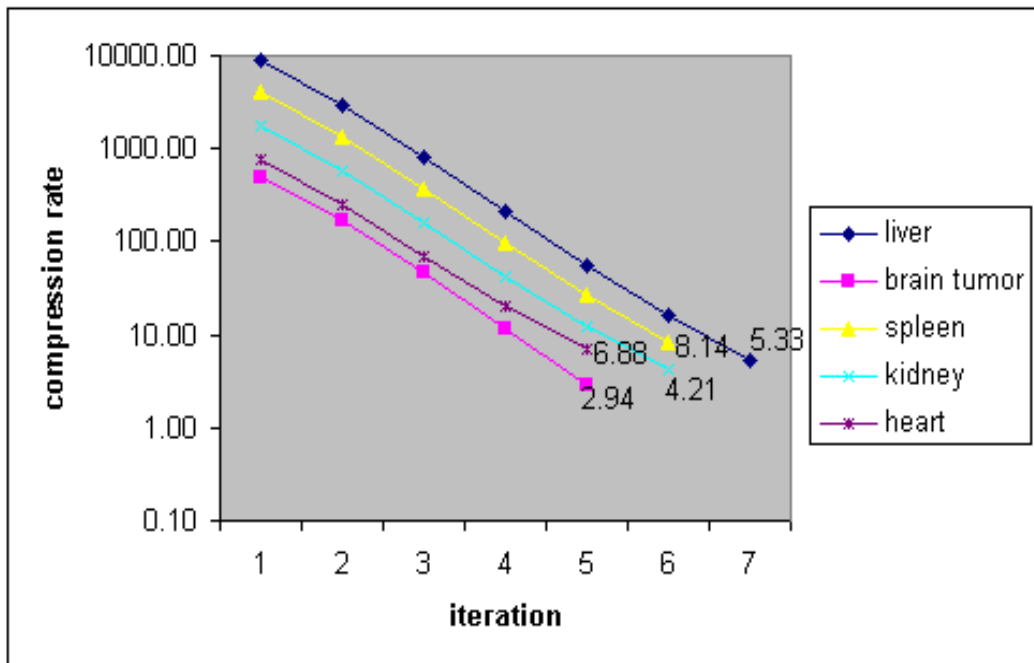


Fig. 10. RaG surfaces approximating the shapes in Fig. 9 with error tolerance of 1 voxel. The number of control points in these surfaces are: (a) Liver: 9,986; (b) Brain tumor: 1,026; (c) Spleen: 2,942; (d) Kidney: 2,463; (e) Left ventricular cavity: 662.

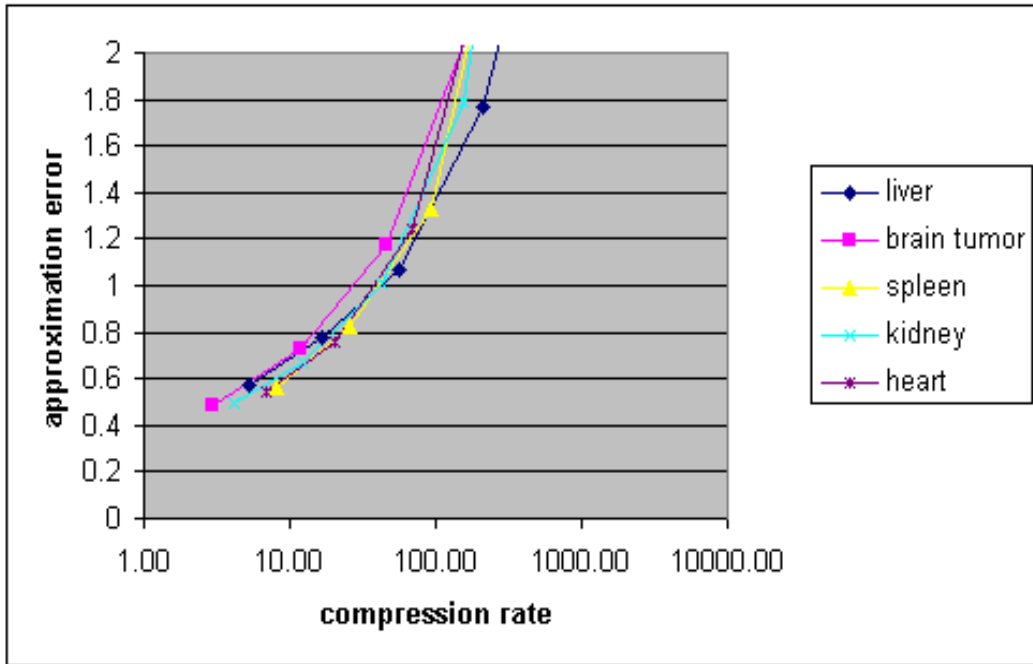


(a)

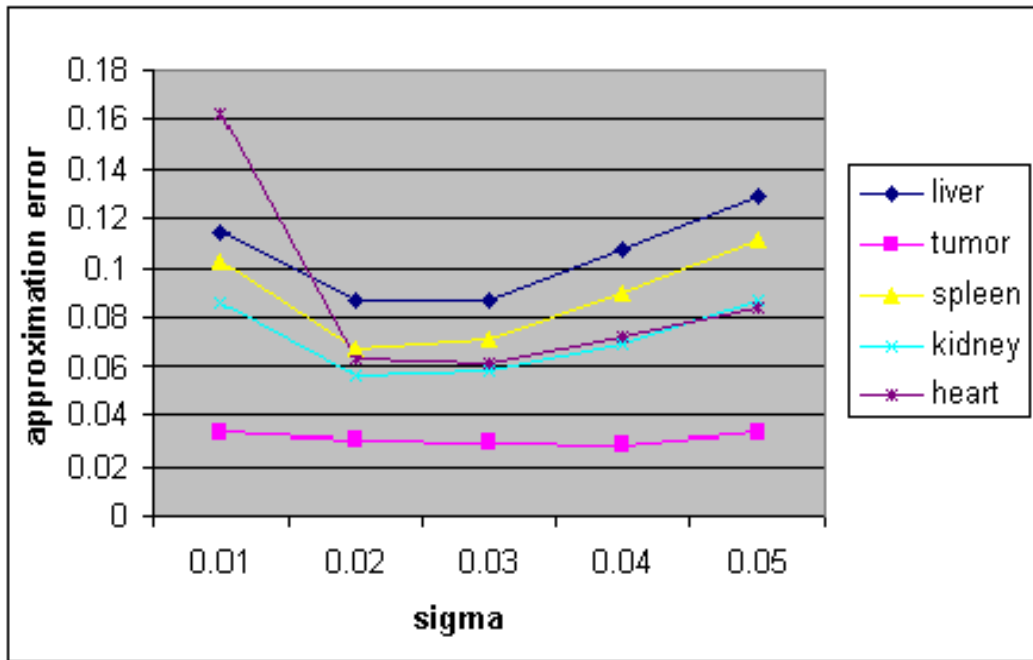


(b)

Fig. 11. (a) Approximation error in surface approximation versus number of iterations for the shapes in Fig. 9. (b) Compression rate versus number of iterations. The scale of the vertical axis in this chart is logarithmic. The unit in measurement of errors is the length of a side of a voxel.



(a)



(b)

Fig. 12. (a) Approximation error versus compression rate for shapes in Fig. 9. The horizontal axis in this figure is logarithmic. (b) Approximation error as a function of the standard deviation of Gaussians. The unit in measurement of errors is the length of a side of a voxel.

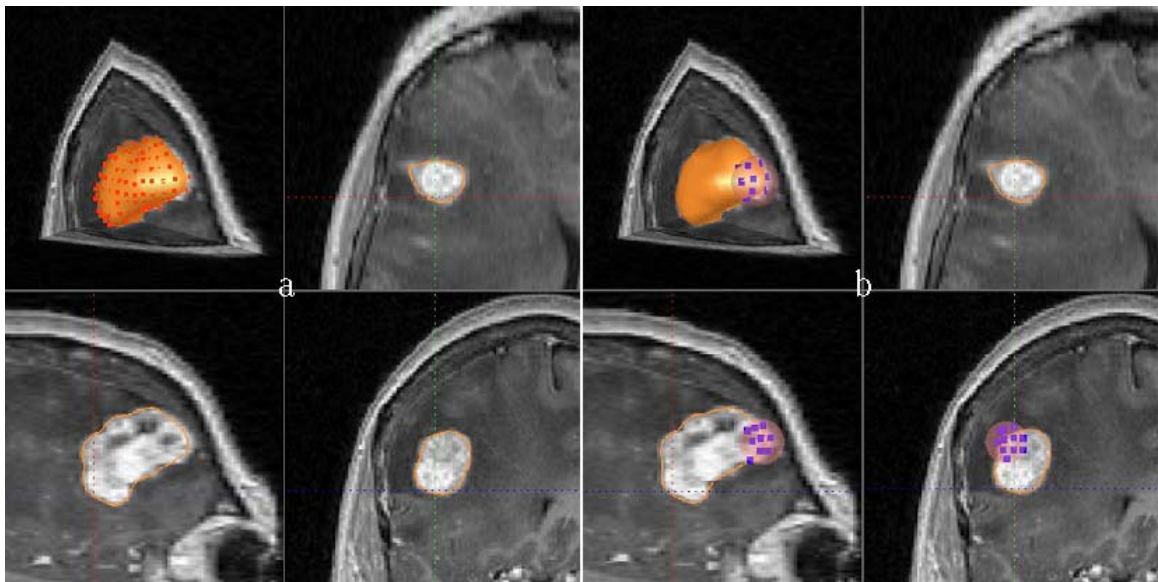


Fig. 13. (a) A brain tumor segmented and approximated by a RaG surface. (b) An intermediate step showing interactive revision of the surface.