Volume Models for Volumetric Data

Vishwa Ranjan and Alain Fournier
Department of Computer Science
The University of British Columbia
2366 Main Mall
Vancouver, B.C., Canada V6T 1Z4

Tel: (604) 822-3061
Fax: (604) 822-5485
E-mail: {ranjan|fournier@cs.ubc.ca}

Abstract

In order to display, transform, and compare volumetric data, it is often convenient or necessary to use different representations derived from the original discrete voxel values; in particular, several methods have been proposed to compute and display an iso-surface defined by some threshold value. In this paper we describe a method to represent the volume enclosed by an iso-surface as the union of simple volume primitives. The needed properties (displayed image, volume, surface, etc.) are derived from this representation.

A survey of properties that might be needed or useful for such representations shows that some important ones are lacking in the representations used so far. Basic properties include efficiency of computation, storage, and display. Some other properties of interest include *stability* (the fact that the representation changes little for a small change in the data, such as noise or small distortions), the ability to determine the similarities between two data sets, and the computation of simplified models.

We illustrate the concept with two closely related representations, one based on the union of tetrahedra, and another based on overlapping spheres, both derived from a Delaunay tetrahedralization of boundary points. The former is simple and efficient in most respects, but is not stable, while the latter needs heuristics to be simplified, but can be made stable and useful for the computation of intrinsic properties of the object, and the determination of similarities between objects.

This approach also helps to develop metrics indispensable to study and compare such representations.

Keywords: `possible changes` visualization, Delaunay triangulation, volume primitives, stability.

1 Problem Statement

A concise statement of the problem we are addressing here is:

> Given a set of points at the boundary of an object derived from *volumetric data*, how can we *represent* the object and in particular *visualize* it from these points?

Volumetric data is used in the sense most authors use it in this issue, but an important feature for our purpose is that an inside-outside test for any point within the volume is available. Our problem is therefore a restricted form of the general problem of visualizing an arbitrary cloud of points [1]. Data in the form of boundary points is now very common due to the advancement of technology in 3D imaging. The most common case is when data points are defined to lie on an iso-surface obtained by thresholding 3D volumetric data (Figures 1a and 1b). To *represent* and *visualize* can be vague concepts. Their meanings depend on the application and the context. A number of properties are considered to be desirable in the representations and primitives used in visualization. A few of these include ease in generation, manipulation, storage and display. Other desirable properties are that the representation captures as much as possible the *intrinsic* characteristics of the object, that is properties independent of the resolution of the data, of the imaging geometry, or even of the imaging method itself. Such properties would help in comparing objects, establishing whether two data sets are from the same object and determining the transformations between them if they do.

To give an intuitive example of the kind of representation we are looking for, assume that we have data somehow representing a human head. The head can in first approximation be represented by a sphere. The surface area and the volume of the sphere will give us rough but useful estimates of the corresponding properties for the head. At the same time the position and radius of the sphere will give us an idea of the translation and scaling to apply to get the head in some canonical position. If we fit an ellipsoid instead, then the additional degrees of freedom might allow us to obtain the parameters of the rotations to apply. Of course we cannot obtain independently estimates for the scaling and the volume or area. Which one is obtainable depends on the context. It is clear that human perception meets these goals very well. We will not seek such power, but be content with some progress in that direction.

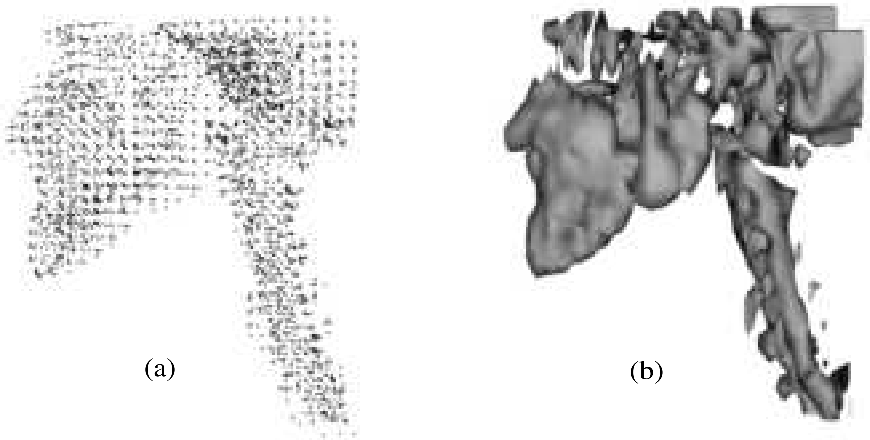


Figure 1: Example of volumetric point data. (a) Points on the surface of a skull in computed tomography head data, and (b) the corresponding iso-surface.

One important property that a representation should have to be useful for our purpose is *stability.* Stability simply means that non-intrinsic changes in the data should not result in significant changes in the model. Non-intrinsic changes include noise and distortion in the data, changes in the resolution in the case of discrete data, and basic geometric transformations, such as scaling, translation and rotations. A semi-formal notation for this property is:

$$R(T_\Delta(\Delta)) = T_R(R(\Delta))$$

where Δ is from the volumetric data (in our case the boundary points extracted form the voxel values), R is the operator applied to the data which generates the representation, T_Δ is a transformation applied to the data and T_R a transformation applied to the representation. Since in general Δ and $R(\Delta)$ will be of different types (in our example Δ is made of boundary points, and $R(\Delta)$ is a union of spheres), the transformations will not be the same. We want, however, to be able to map T_R to T_Δ, so that if we retrieve the transformation from our representation we can deduce the transformation on the data. This is this property that will allow us to obtain a model as independent of the imaging technique as possible.

Another property that is useful is the ability to simplify our representation, while keeping as much of its characteristics as possible. This helps speed up display, interactive manipulation, and the extraction of models independent of the original level of details available in the data.

In section 2, we will survey existing methods to represent and visualize objects from discrete boundary points. Section 3 introduces the general idea of using volume primitives to represent volumetric data. In section 3.3, we will describe two closely related representations based on a Delaunay tetrahedralization (see glossary) of boundary points. Some theoretical and combinatorial results relevant to the representations are also presented. In section 4 we will present experimental results and evaluate this approach and in section 5 we will summarize the results and discuss the current and future extensions.

2 A Sample of Current Methods

We assume that a boundary has been defined for scalar volumetric data set by giving a *threshold value* and defining as *inside* any point above the threshold, *outside* any point below the threshold, and as a *boundary* point any point whose value (generally interpolated) is equal to the threshold. Clearly without loss of generality the role of the outside and the inside could be switched. The problem is then to represent and visualize an object whose boundary is so defined.

The methods used can be divided into three categories: methods which do not represent the surface explicitly, but rely on direct volume rendering and (human) visualization to "see" the boundary, methods which construct the surface explicitly, and methods which use volume primitives whose surfaces (or subset of same) define the object's surface. The first category of methods will not be further discussed here, since by not computing the surface directly they do not meet many of our criteria (but of course if visualization is the only goal they can be quite effective and powerful). We will then concentrate on representatives of the *surface based* and *volume based* methods.

2.1 Surface Based Methods

Surface methods represent the boundary of the object by fitting surface primitives to the point data. Usually the primary concern is fast surface display from different viewpoints. The complexity of the method depends on the surface primitives used. In medical imaging the most commonly used technique of this type is the so called *marching cubes* technique which fits polygons to the boundary points in every voxel [2]. The advantages of such representations are that they are quick to compute and are easy to display since many graphics workstations support polygon display in hardware. Their success is due to the fact that interactivity (that is the ability to compute a new image at or near video rates) plays a crucial role in visualization. The disadvantages are that such surface representations are *piecewise*, lacking information about the global properties of the object (total volume, connectivity, inclusion), there may be a large number of primitives and that their number and distribution are highly resolution and sampling dependent. All of these mean that the primitives cannot be used to characterize the object or compare it to other objects. These methods are nevertheless very useful and important, and *marching cubes* in particular will be used as a standard for comparisons in our own investigation.

A similar approach, based on different surface primitives, consists in fitting a surface to the set of boundary points considered as an arbitrary clouds of points, ignoring whatever neighborhood information could be kept from the original volumetric data. An effective algorithm to compute such surfaces has been developed by Hoppe *et al* [3]. The fact they do not assume any structure in the set of points is both a strength and a weakness of the methods for our purposes (since useful information about inside/outside classification has been lost). The method also shares with the above method the problem of possibly generating a large number of triangles, and in fact they use marching cubes as one step in the overall method. Methods to simplify this kind of mesh exist ([4]), but they can add problems of their own (see about stability below).

2.2 Voxel-based methods

Octrees are hierarchical data-structures to represent volumetric objects [5], assuming a binary classification of each voxel. A bounding box around the whole data is the root node for the tree, and each node is then recursively sub-divided into eight subnodes (octants) if they are mixed (not entirely inside or entirely outside). The advantages of octree representation are that they are generally more compact than the voxel representation, they are hierarchical, and elegant algorithms exist to generate them and apply some operations (e.g. boolean operations) on the representation. For display, voxels can be displayed directly (using raytracing), or triangles can be generated within all boundary voxels (using a method similar to marching cubes). One major disadvantage with the octree representation is that it is highly dependent on the coordinate system. A small change in value, especially near the boundary between nodes high in the hierarchy, will result in very different trees, and this hampers the use of octrees in problems such as characterization of shapes and comparison of data sets.

2.3 Shell Representation

An interesting model, with characteristics of voxel fuzzy classification and boundary representation, is using *shells* [6]. A shell is a set of voxels in the vicinity of the boundary, with "fuzzy" membership values associated with them. Shells can be used directly for rendering, but the authors show that they can also be used for various operations such as volume computations and measurements on surfaces. As useful as they seem to be, especially because of the speed of rendering obtained, shells do not meet some of our criteria, mostly because they are the result of a classification scheme of voxels, and as such very dependent of the resolution and geometry of the original imaging system.

2.4 Fitting volume primitives

The problem of deriving a geometric model of an object from discrete boundary data for classification or comparison is a well-studied problem in computer vision. A common technique in vision is to fit a single volume primitive (such as a deformable superquadric [7] or others [8]) to the boundary points by posing the problem as an extremization problem. The advantages are that a fairly complex shape can be defined with a relatively few parameters, and the hope is that the method is stable making it useful for shape comparison. Since, inside-outside property is defined for these primitives, they can be conveniently ray-casted or ray-traced for display. This approach is attractive for use in visualization but its main drawback in our context is that a single primitive cannot represent effectively most objects of interest with volumetric data. Some *a priori* decision has to be made about the expected complexity of the object(s) to be represented. We will see that our approach can be seen as a variation with simpler, but multiple primitives, whose number is not pre-determined.

2.5 Assemblage of Volume Primitives

Various methods have been used to represent an object with structures of volume primitives. We will mention three widely different ones.

2.5.1 BSP Trees

Binary Space Partitioning (BSP) trees are structures made of the union and intersection of half-spaces [9]. They are widely used not only as data structures for efficient spatial query and navigation, but also as modeling tools [10]. For volumetric data, the strategy is to generate the BSP trees by fitting planes to the boundary points. This needs good heuristics to be effective. Once this is done, the structure produced has the advantages of BSP trees (quick Boolean operations, good visualization). Also the simplification of the model is relatively easy, though stability is not guaranteed, being mainly dependent on the methods used to produce the original planes, and the characteristics of the voxels.

2.5.2 Wavelet Representations

Wavelets have developed in the last few years as powerful analytic and numeric tools for a variety of tasks [11]. They essentially allow the representation of signals in a hierarchical multiscale basis, sometimes with a high level of compression. They have been used in the context of volume data representation by Muraki [12]. Even though the intrinsically hierarchical nature of wavelet transforms allows several level of representation, it is not clear that each level is in fact useful either visually or for the purpose of the analysis of global property. In particular the fact that the wavelets transforms used are not invariant under translation, and sensitive to scaling, makes their use for the comparisons of objects and the extraction of transformations problematic.

2.5.3 Alpha-shapes

One innovative computational geometry technique to investigate "shapes" from an arbitrary cloud of points is the use of *alpha-shapes* [1]. The idea is to assume a spherical eraser of radius α and take it everywhere in 3-space where it can go without including any of the original points. The part of 3-space that has not been erased is the alpha-shape, and represents the object at the scale α. At one extreme, for an infinite α, the result is the convex hull of all the points. At the other extreme, for an α of 0, the result is the set of points itself. In-between, the shape is normally bounded by arcs of circles and spherical segments, but they are normally replaced by straight line and plane segments. In the algorithms designed and implemented by [1], the representation is derived efficiently using Delaunay tetrahedralization. The representation obtained has the great advantage of being a various and controllable scale. One disadvantage is that the same scale applies to all features. This shortcoming was removed by the definition of weighted alpha-shapes [13] but the problem of automatically selecting different weights for different features still remains. The most serious drawback for our purpose is the fact that whatever *shape* is, it is not captured by this representation, and the volume primitive used, the tetrahedra, are sufficiently stable to be *directly* usable. Further the number of tetrahedra can be as large as $O(N^2)$ (where N is the number of boundary points), hence the representation may not be compact to store or easy to manipulate.

3 A Volume Representation for Visualization

3.1 Using volume primitives

Under the right circumstances many properties, such as surface area, volume, even topology or connectivity can be obtained from primitives which are not intrinsically related to the object they represent. For instance surface area can be computed from a triangulation of the surface even if no information is available on the adjacency between triangles. If we want the representation to be *complete*, in the sense that both the surface and the inside of the object are represented, accurate (the measurable properties and the "look" of the representation should be closed to the object -inasmuch as we know what it is), stable (as defined above), scalable (in terms of details), displayable and manipulable, then the choices are relatively few. We need volumes to satisfy completeness, simple volumes to reduce the search space when fitting to the data and facilitate visualization, and more than one volume to hope for accuracy. We will specifically here present the study of the representation of the object as a union of simple volume primitives.

We will describe the algorithms used to extract the representation as a union of spheres, to simplify and display it, and to retrieve transformations between related volumetric data sets using our representation.

3.2 Flow-chart of the method

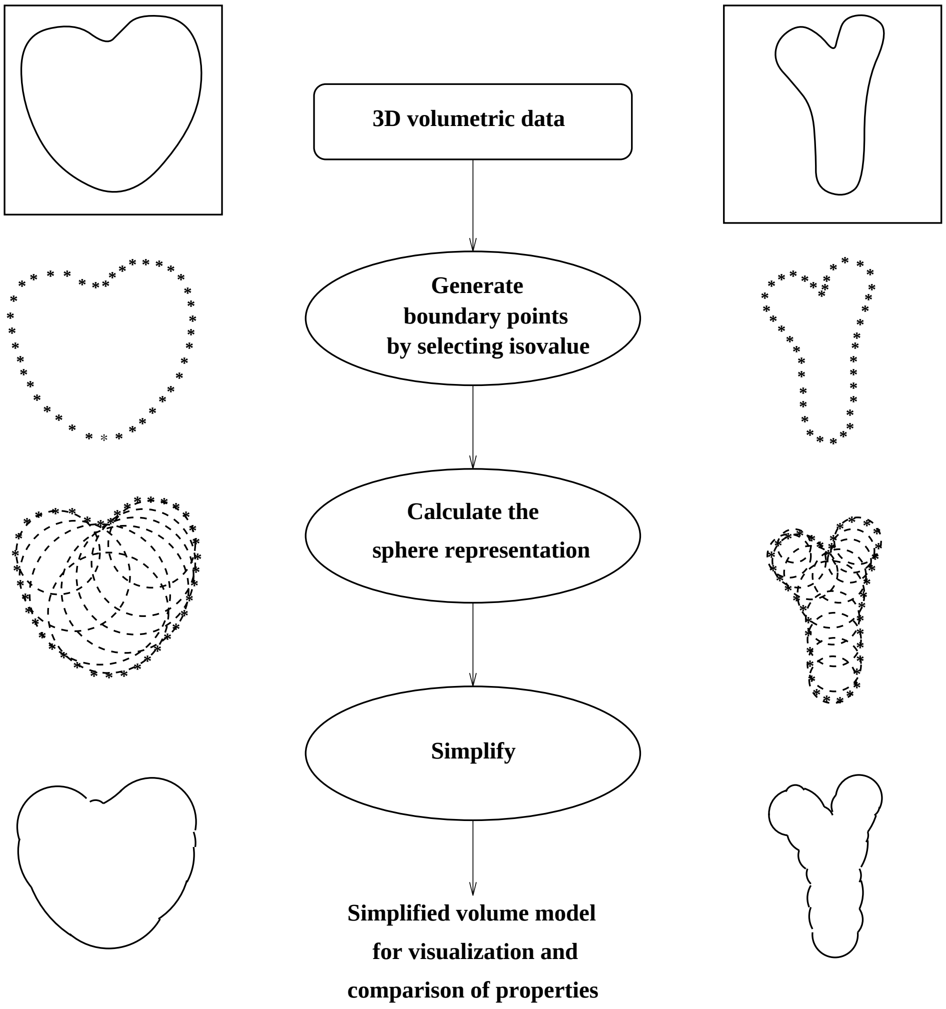


Figure 2: Flow chart for volume representation of volumetric objects.

Figure 2 gives a flow chart of our approach. The steps in the method in the case of a single object are:

1. Generate the boundary points on the surface of the object from the volumetric data.

2. Generate the sphere representation from these boundary points.

3. If needed simplify the sphere representation using clustering or other techniques.

4. Compute needed quantities such as volume or areas from the sphere representation.

Object definition: The boundary points are defined by choosing a threshold value, or a range of values. Again we assume the volumetric data is scalar. Quite obviously the position of the boundary points will depend on the method used to interpolate the original data, and that is another reason to seek a stable method. In our case we find the boundary points by linear interpolation along edges between voxels whose values straddle the threshold. This is an $O(N)$ process, if N is the number of voxel values. The number of boundary points so created if also $O(N)$, with 0 being an obvious minimum, and $2N$ a maximum in 2D, $3N$ the maximum in 3D.

3.3 Generating the sphere representation

There are a number of ways a sphere representation can be derived from the boundary points. For instance one can use image analysis techniques like distance transform `reference here` [14] or thinning to obtain a skeleton with (approximate) distance to the nearest exterior voxel. The use of spheres for representing objects similar to ours has been proposed by Knowlton [15] and further investigated Badler and Bajcsy [16]. In 1979, O'Rourke and Badler presented an algorithm to derive a sphere representation from discrete contour data [17]. Their algorithm is very unstable and there is no guarantee that all of the object volume will be included in the representation. The union of spheres or ellipsoids representation has also been used in graphics for modeling [18] and rendering (e.g. blobbies [19], [20]). Fast display techniques for such a representation have also been discussed. In earlier work, O'Rourke and Badler describe a technique of decomposing an object into spheres from contour data [17]. For all boundary points, their algorithm essentially fits the largest sphere not containing any other boundary point.

Our main criterion is to obtain a set of spheres such that all inside voxels are included within at least one sphere, and no outside voxel is within a sphere. In addition we require that no boundary point is inside any sphere, and a subsidiary criterion is that all boundary points are on the surface of at least one sphere.

3.3.1 The Algorithm

The algorithm to generate the sphere representation is based on a Delaunay tetrahedralization triangulation in 2D) of the boundary points. We use Delaunay tetrahedralization because of its empty sphere property, which guarantees that we will fit the largest spheres possible defined by four boundary points and not including other boundary points. For sake of clarity, the algorithm will be described in two dimensions. For 3D, read “triangle” as “tetrahedron”, and “circle” as “sphere”. Figure 3a shows a computed tomography (CT) slice through a

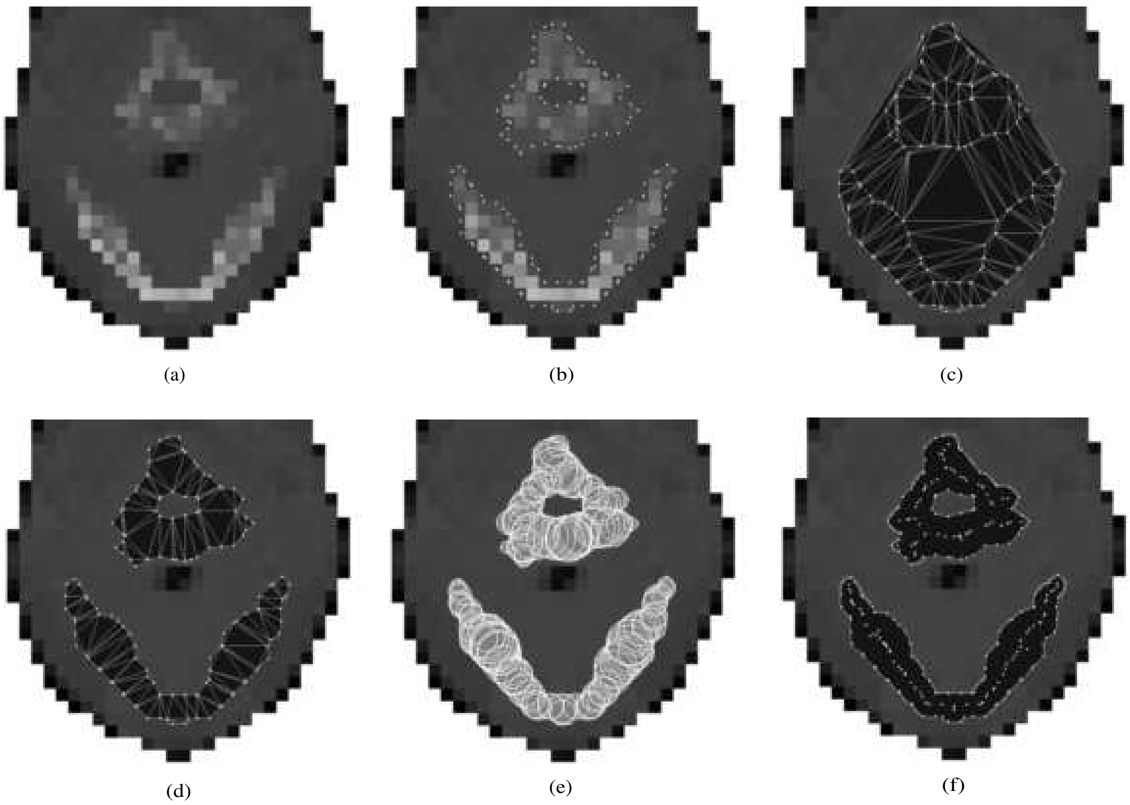


Figure 3: Algorithm to generate the triangles (circles) representation. (a) A (32×32) size CT slice of human head (range 0–255). (b) Boundary points for a threshold of 70.3 (c) Delaunay triangulation of the boundary points. (d) Triangles representation. (e) Circles representation. (f) Circles and triangles superimposed, and the centers of the circles.

human head at a 32×32 resolution. The steps in the representation generation algorithm are:

1. Compute the boundary points (Figure 3b). `give range of values and threshold value used` In the figure, the selected regions correspond to a vertebra and the lower jaw.

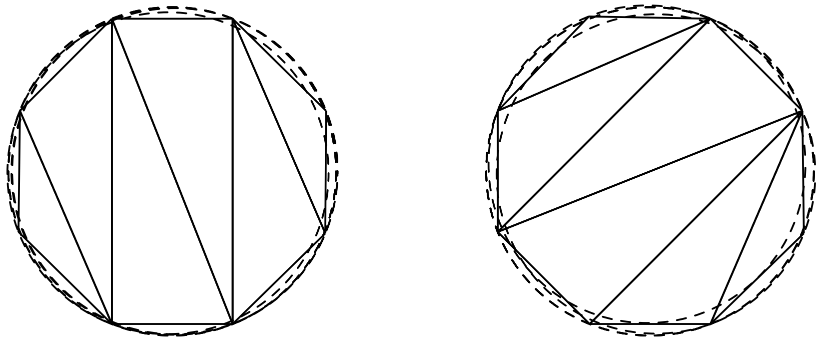
2. Compute the Delaunay triangulation of these boundary points (Figure 3c). We assume here that the generality condition (see glossary) on points holds. This is not a limitation, and there are reliable ways to ensure this [1].

3. Compute the circumscribing circle of each triangle. By the empty circle property of the Delaunay triangulation, this circle does not contain any other boundary point. Therefore, if any point within the circle is known to be inside (or outside), the whole circle can be defined to be inside (or outside).

4. Use the original volumetric data to verify which circles are inside. This is done by checking the value at the grid point if any grid point is inside the circle. If there is no grid point within the circle, we use interpolation to see if the center of the circle is inside (or outside). Instead of working with grid points, we can directly check if the median points of triangles, or the centers of circumscribing circles are inside or outside. By using values at grid points we avoid unnecessary interpolation as much as possible. It is important to note that inside and outside is here a classification of the whole primitive (triangle or circle), but not of all the points inside. In particular an outside circle can intersect an inside one (but it cannot happen with the triangles. For consistency, we use the same interpolation scheme for the inside/outside test of the non-grid point as was used to derive the boundary points.

The inside circles define our representation (Figures 3d and 3e). As shown in the next section, the differences in the properties of the representations as triangles or as circles depend on the boundary point density δ on the surface of the object. If δ is small compared to the "size" of the object features, the two representations are close approximations of each other.

The triangle representation has certain advantages. In 3D, any object with polygonal faces can be decomposed exactly as a union of tetrahedra. Both surface and volume information are readily available. It is easy to compute mass properties such as surface area, volume etc., and topological properties as connected components, holes etc. using Euler's formula for polyhedra. Since display is polygon-based, it is fast on many machines. One disadvantage



(a)

(b)

Figure 4: Figure showing the stability of the circles representation over the triangles representation. (a) Delaunay triangles and the corresponding circumcircles for a set of eight points. (b) Delaunay triangles and the corresponding circumcircles when the points were randomly perturbed by very small amounts.

are that there can be a large number of tetrahedra in the representation. $O(N^2)$ is the worst case in 3D, but $O(N)$ is more typical when dealing with boundary points from volumetric data. The representation is not very stable i.e. perturbation of a few boundary points may result in a very different triangulation (see Figure 4) and therefore comparison of two data sets using individual primitives is not possible.

Derivation of simplified models from this representation is not easy though possible.

The circles, by contrast are stable, even though the triangles that generated them are not. See Figure 4 for an illustration. The disadvantages of the circle representation are that a circle has very few degrees of freedom and hence the kind of shapes that can be accurately represented by a union of small number of circles is fairly limited. Further, many properties (e.g. mass, topological) seem to be difficult to compute as the primitives are overlapping. We will see, however, that efficient techniques based on powerful data structures exist.

Notice finally that while it is straightforward to derive the spheres representation from the tetrahedra representation, it is not reversible.

3.4 Clustering and simplification

There are two main motivations to reduce the number of primitives. The first is to simplify and speed up manipulation and display of the object. The second is to facilitate the comparison of two objects by reducing them to the same number of primitives. The problem here is to reduce the number of primitives without compromising much on the object properties (volume, surface area, topology etc.) The problem is two-fold: (i) deciding which spheres can be clustered or simplified and (ii) deciding which primitive can replace the selected spheres. In our context, replacing the cluster with a sphere is the best choice. Since spheres are used as terminals in the structure, we can have at any time a fully formed data structure. A single primitive can stand exactly for itself, and we can develop easily a meaningful measure of fit between the representative and the clustered spheres. We define as *sphericity* of a cluster of spheres the ratio between the radius of the smallest sphere including all the spheres in the cluster to the radius of the largest sphere in the cluster. Obviously sphericity is a real number between 0 (the limit for vanishingly small spheres such that their maximum interdistance is finite) and 1 (where the set of sphere is represented exactly by a single sphere. `is a figure useful here?`. We could have used the ratio of volumes of the spheres, or surface areas, but they all have the samne range and are monotonically increasing with respect to each other, so they are equivalent in practice. Another choice would be the volume of the union of spheres, instead of the largest sphere, but besides the fact that it is harder to compute, in our case since our spheres have been generated from a Delaunay triangulation, the largest sphere is a better indication of the potential of the set of sphere to be broken down in large spheres than the total volume.

We can now sketch out the clustering algorithm based on the sphericity measure. It is a top-down cluster, working by subdivision, but exploiting the fact the the elements of the clusters are spheres, not points. **Input**: a set of spheres produced from the volumetric data as described above.

Output: a list of Spheres representing the same data, which each Sphere having a sphericity $>= T_S$, where T_S is a chosen sphericity threshold. Each Sphere points to the list of spheres it contains.

1. Initialize the list of Sphere to `NULL`.

2. Compute the smallest enclosing Sphere for the whole set. Compute its sphericity. Add to the list.
3. List processing: If number of Spheres in list equal or greater than limit, quit.
4. Find the Sphere in the list with the smallest sphericity. If above threshold, quit.
5. Find within that Sphere the Sphere with the largest radius whose sphericity is above the threshold. Remove the current Sphere from the list. Add the new Sphere to the list.
6. Once found, identify all the connected components for the union of spheres not inside the Sphere found.
7. For each of these connected components, compute smallest enclosing Sphere and its sphericity, and add to the list.
8. Continue list processing.

We will state without proofs a few facts about the algorithm. The algorithm will terminate, since at each step we put at least one sphere in a Sphere that will not be split again (because it is above the threshold). We are sure to find a Sphere below the threshold in step ?? since each sphere can represent itself with a sphericity of 1. The complexity of the whole algorithm is very high, in fact at least NP-complete `check this; note that clustering is`. The algorithm produces in general a stable representation, which becomes unstable locally in unavoidable cases (long cylinders, nearly parallel boundary planes).

`Describe the median/average based clustering actually used for the pictures.`

3.5 Computing Mass properties and Topology

A specialized data structure, the *power diagram*, or *Laguerre-Voronoi* diagram, is very useful for this and further operations. The power diagram is similar to the Voronoi diagram, but the distance used to define the polyhedral cells is $D^2 - R^2$, where D is the distance from the test point to the center of the sphere, and R is the radius of the sphere. Figure 5 shows the power diagram for 9 circles. The essential property of power diagrams for our purpose is that [21] the union of the sphere is the sum of the intersection of each sphere with its power

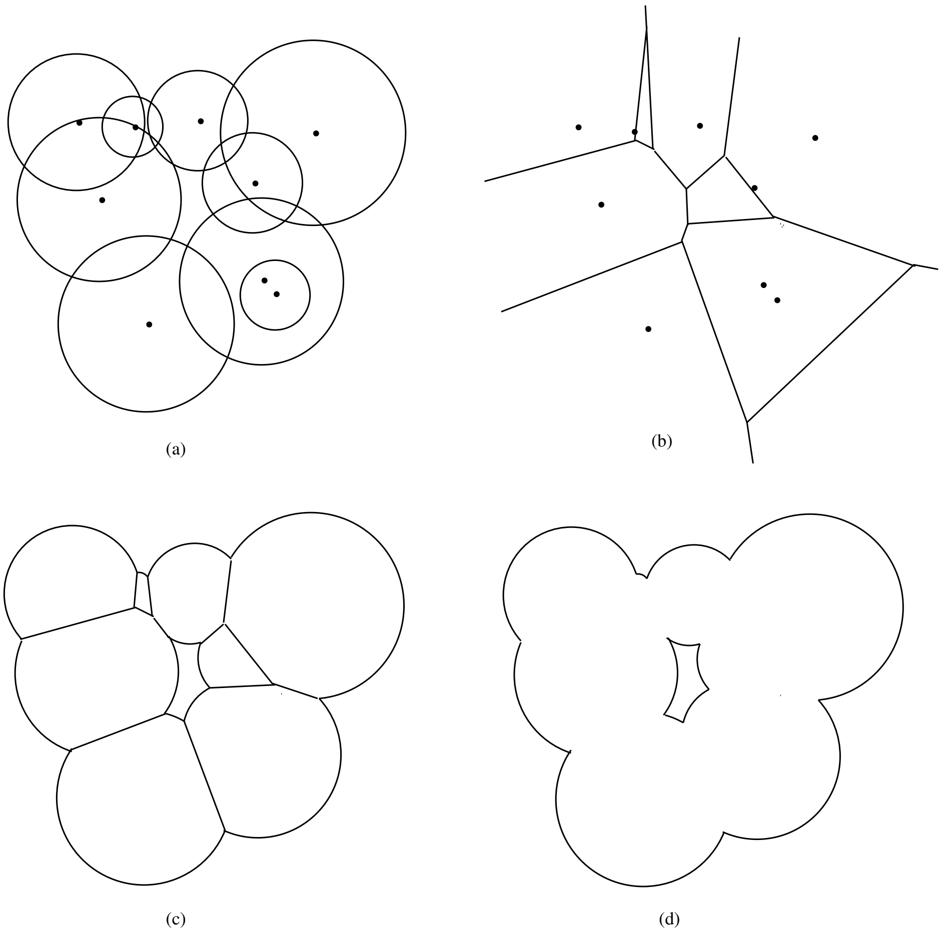


Figure 5: (a) Set of nine circles. (b) Their power diagram. (c) Decomposition of circles using their power diagram. (d) Union of circles.

cell (the intersection can be empty). We can them quantify the contribution of a sphere to the total object, and rank the spheres by order of "contribution" to the total. Edelsbrunner has showed that the power diagram can be used efficiently to compute the volume, surface area and topological properties in $O(N^2)$ time [22]. `more here`.

3.6 Display of Union of Spheres

The direct display of the object as a union of spheres is generally not fast on current graphics workstations, because the hardware and display software is strongly polygon-biased. Research architecture such as PixelFlow [23] allows the rendering of spheres at rates of ???/s. Raytracing software with space subdivision performs well on spheres, and our own raytracer, *optik*, can render 10,000 spheres in ?? s. on a 50 MIPS machine.

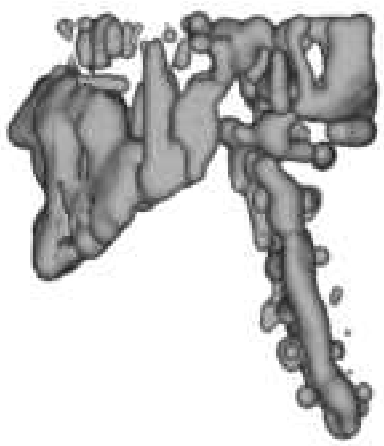


Figure 6: Smooth display of the spheres representation of the skull data using blending functions.

As seen in figure ??, because of the presence of small spheres created by the discrete positions of the voxels, the surface does not appear smooth. One effective way to smooth it in rendering (without affecting the representation itself) is to use blended sphere (also known as *blobbies* in rendering/modeling circles. Our raytracer includes this, and Figure ?? show the result for the same spheres as in Figure ??.

3.7 Extracting Transformations from the Model

`More details here..`

3.8 Some Theoretical Observations

This section presents without proofs theoretical and combinatorial results relevant to the representations described above. These results are in three-dimensions. N is the number of boundary points computed initially.

Observation 1 *The number of boundary points is at most 3 times the number of voxels.*

Observation 2 *The number of tetrahedra (hence spheres) in our representation in the worst case is $O(N^2)$.*

Again the average number is $O(N)$.

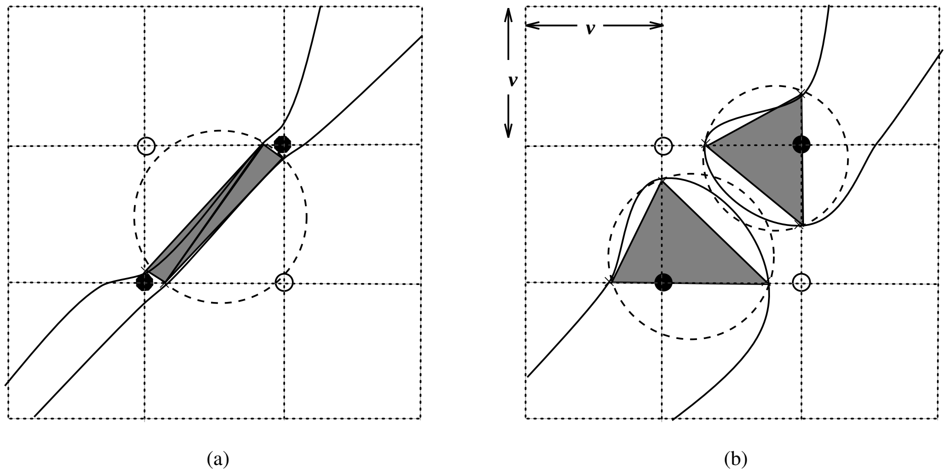


Figure 7: (a) A circular cap of volume $V_\delta/2$ on one of the boundary faces. (b) A configuration of boundary points so that the circles touch each other but the corresponding triangles do not.

Observation 3 *Let V_T denote the volume of the union of tetrahedra and V_S the volume of the union of spheres. Let δ be the boundary point density (see glossary). Let V_δ denote the volume of a sphere with diameter δ. Then, $V_T < V_S < V_T + (N-2)V_\delta$*

This bound is very pessimistic. The boundary point density δ is related to the size of a voxel. To be more specific, $\delta \leq \sqrt{3}v$, where v is the linear dimension of a voxel.

Observation 4 *In general, the topology of union of tetrahedra representation and the topology of union of spheres representation may differ.*

The topology of the tetrahedra representation and that of the sphere representation differ when two spheres touch each other but the corresponding tetrahedra do not (Figure 5b). When this happens, it is indicative of the fact that sharp features exist in the object. In these places, either topology (spheres or tetrahedra) can be correct; the correct topology should be determined from the volumetric data.

Observation 5 *Two components (or branches of the same component) separated by more than δ (as defined above) cannot join.*

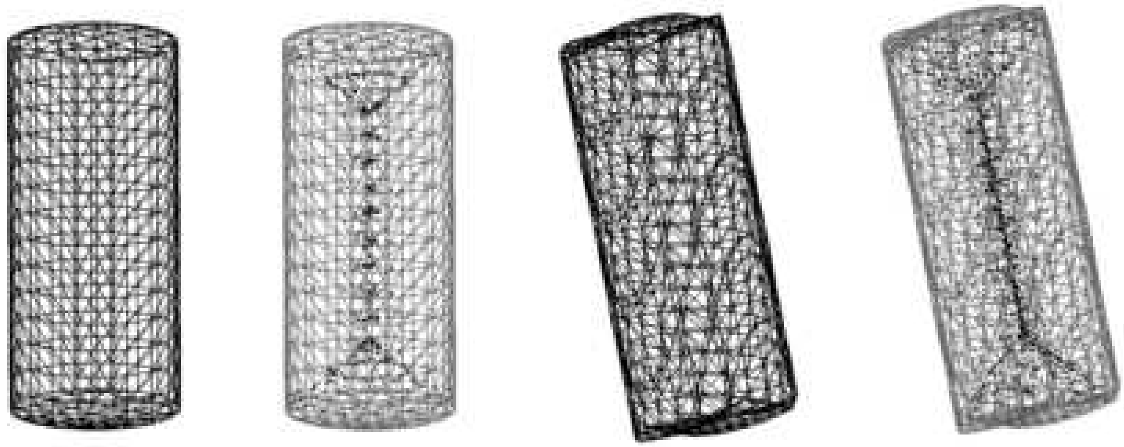
This is because for a boundary point density of δ, the maximum a sphere can protrude is $\delta/2$ (see Figures 5a and 5b).

4 Experimental Results

`Modify according to new figures and results.`

The program is implemented in C language on Silicon Graphics IRIS machines. For our experiments we are working with both synthetic and real volumetric data. Examples of synthetic data include (i) a ball and (ii) a cylinder. The density for both decreases uniformly from the center (255.0 in the center and 0.0 at the radius of about 11.5 voxel units). The radius of the ball (or cylinder) is decided by the chosen threshold value. For example, a threshold of 120.0 defines a ball of radius of about 6.0 voxel units. The real data consists of computed tomography head data which was resampled to a small size ($32 \times 32 \times 32$) for experimental purposes.

Figures 6 and 7 show the results for the ball and cylinder data for a threshold value of 30.5 and 100.5 respectively. Figures 6a and 7a show the points on the boundary of the volumetric object. Linear interpolation between neighboring grid-points was used to compute these boundary points. Figures 6b and 7b show the tetrahedral decomposition of the object. Figures 6c and 7c show the spherical decomposition of the object. When necessary, trilinear interpolation was used to decide if the center of a sphere was inside/outside. The distributions of radii of spheres is shown in the histogram of Figures 6d and 7d. There are 40 buckets in



(a) (b) (c) (d)

Figure 8: Results on artificial cylinder data indicating that stability can be achieved with the proposed representation using appropriate heuristics for simplification. (a) Surface mesh. (b) Centers of spheres in the spheres representation. (c) Surface mesh for the transformed cylinder. (d) Centers of spheres in the spheres representation. Note that the spheres seem to transform "according" to the object.

each histogram. Figures 6e and 7e show the location of the centers of the spheres. Figures 6f and 7f show the objects when small spheres are removed. The possibility of multi-resolution display of objects using spheres is demonstrated in the Figures 7f and 7g. For the cylinder data, if we display only the spheres of the bucket containing the largest spheres, we get the object shown in Figure 7f. Figures 7g shows the cylinder at an "intermediate" resolution. As expected, many spheres are centered close to the medial axes. Spheres whose centers stray away from the medial axes capture the details in the surface. O'Rourke and Badler reported similar results in an earlier paper [17].

The number of boundary points in the sphere data was 1704 and it took about 4 minutes on an IRIS Crimson to derive the representations. There were 5120 tetrahedra (spheres) in the representation. There were 736 boundary points for the cylinder data and it took about 35 seconds of processing time on the same machine. There were 2381 tetrahedra (spheres) in the representation. The maximum number of tetrahedra (or spheres) observed so far is about $4n$ (n being the number of boundary points).

Figures 8a-e show the results when the cylinder is transformed in the volumetric data

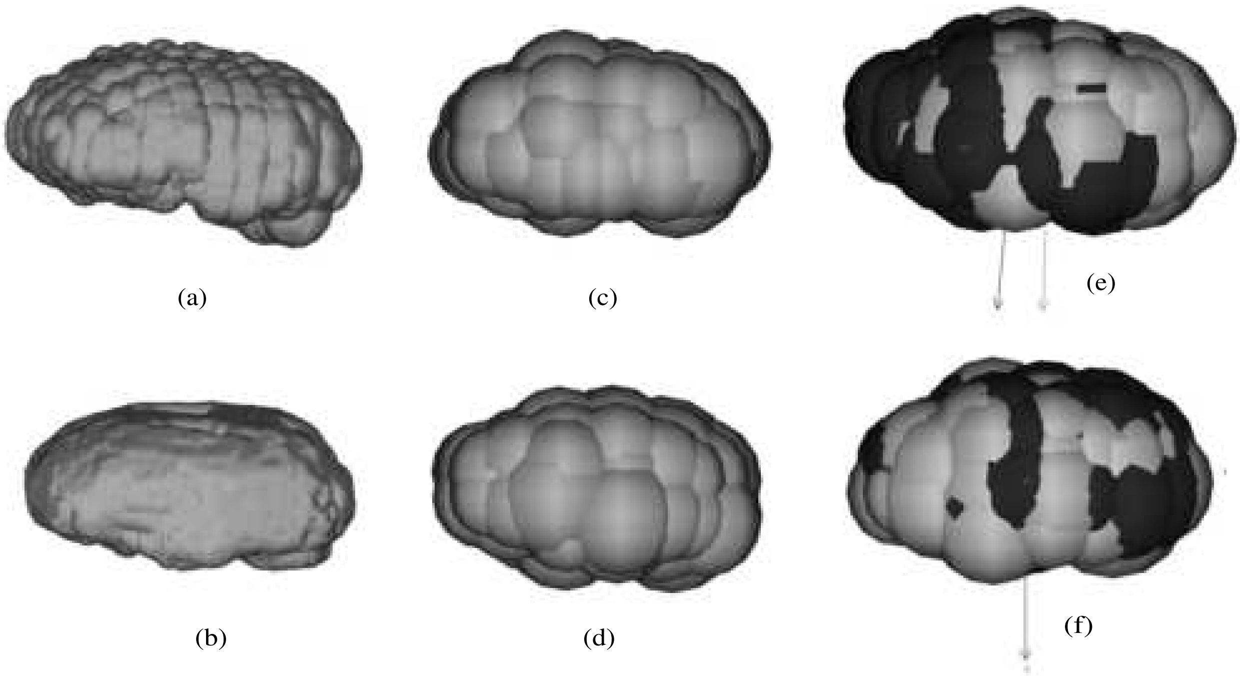


Figure 9: Results on the real PET brain data. (a) Brain at $(16 \times 16 \times 16)$ resolution. (b) Transformed brain at $(64 \times 64 \times 64)$ resolution. (c) Brain in (a) using 64 spheres after simplification. (d) Brain in (b) using 64 spheres after simplification. (e) Data sets (c) and (d) before registration. (f) Data sets (c) and (d) after registration (registration was performed by deriving the transformation between the primitives after matching).

(translation of about 2.5 voxel units in the positive Z direction, rotation of 15 degrees clockwise about the negative X-axis, and dilation of 1.2 in each direction). The Y and Z axes are in the plane of the screen (Z pointing upwards), X-axis comes out of the screen. As can be seen again, many spheres are centered close to the medial axes. This convinces us that proper clustering and simplification of the sphere representation will give us a representation stable enough to develop metrics for shape matching and comparisons. It will then be possible to register different volumetric data sets of the same object by deriving the transformation between the representations of the same object in the two data sets. As a simple example, if there is only uniform scaling between two data sets (e.g. an object is imaged at different resolutions), we can use the radius of the largest sphere in the representations of the same object in the two data sets to determine the scaling between the two data sets.

Figure 9 shows an example of a method to display spheres with their boundary smoothened using blending functions [19]. The image was generated for the resampled head data when blending functions similar to blobbies (or metaballs) were used for smooth display. This result is for the same iso-surface as shown in Figure 1b (which was generated using marching cubes [2]). It took about 12 minutes to raytrace this image with 1204 spheres on an IRIS Crimson machine.

5 Conclusion

`to be modified..`

We propose the use of volume primitives for representation and visualization of volumetric data with . This was motivated by the fact that the current techniques lack some important properties and the use of volume primitives shows some promise to achieve these properties. The method first derives volume models with as many simple volume primitives (spheres) as needed followed by clustering and simplification. The results so far indicate that the representation can be stabilized. This is because the sphere representation is derived using Delaunay tetrahedralization of boundary points which in some sense gives the "largest-possible" spheres inside the object. Intuitively, these largest-possible spheres within the object should not change much with little perturbation in the data even though the generated tetrahedra can be quite different. The representation will be useful in characterization of shapes, and comparison of objects. It will be possible to detect transformations between two data sets using the representation and hence use it for data registration.

The major results so far are:

1. A pair of representations (tetrahedra and spheres) which together have many desirable properties for visualization. Theoretically, the approach works for "volumetric" data in any dimension.

2. A framework which shows some promise to yield simple and stable volume models for volumetric data. So far, we have mostly experimented with small-sized data. The current research emphasis is on methods to simplify the sphere representation.

In general, we are exploring the use of volume primitives in visualization. In this paper we assumed that the points on the surface of an object can be defined by a threshold which is given to us. How to choose a threshold which defines a meaningful object in the data set is another important problem. In our other experiments, volume primitives seem to show some promise in this regard also.

Some problems can benefit from the idea that if volumetric data is available, it can be referred to if necessary. For example, it might help resolve inter-slice connectivity ambiguities in the contour-connecting approaches to generate surface representations from medical data [24]. The idea here will be to see that a line joining a contour on one layer to a contour on another layer is completely inside the object.

References

[1] H. Edelsbrunner and E. P. Mücke, "Three-dimensional Alpha Shapes", Technical Report Rept. UIUCDCS-R-92-1734, Comput. Sci. Dept., Univ., Illinois, Urbana, Illinois, 1992.

[2] William E. Lorensen and Harvey E. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm", *Computer Graphics (SIGGRAPH '87 Proceedings)*, Vol. 21, No. 4, July 1987, pp. 163–169.

[3] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle, "Surface Reconstruction from Unorganized Points", *Computer Graphics (SIGGRAPH '92 Proceedings)*, Vol. 26, No. 2, July 1992, pp. 71–78.

[4] William J. Schroeder, Jonathan A. Zarge, and William E. Lorensen, "Decimation of Triangle Meshes", *Computer Graphics (SIGGRAPH '92 Proceedings)*, Vol. 26, No. 2, July 1992, pp. 65–70.

[5] D. Meagher. "Geometric Modeling using Octree Encoding", *Computer Graphics and Image Processing*, Vol. 19, June 1982, pp. 129–147.

[6] Jayaram K. Udupa and Dewey Odhner, "Shell Rendering". *IEEE Computer Graphics and Applications*, Vol. 13, No. 6, November 1993, pp. 58–67.

[7] D. Terzopoulos and D. Metaxas, "Dynamic 3D Models with Local and Global Deformations: Deformable Superquadrics", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 13, No. 7, July 1991, pp. 703–714.

[8] Xin Li and J. Michael Moshell, "MOdeling Soil: Realtime Dynamic Models for Soil Slippage and Manipulation", *Computer Graphics (SIGGRAPH '93 Proceedings)*, August 1993, pp. 361–368.

[9] Bruce F. Naylor, "Partitioning Tree Image Representation and Generation from 3D Geometric Models", *Proceedings of Graphics Interface '92*, May 1992, pp. 201–212.

[10] Bruce F. Naylor, "Interactive Solid Geometry via Partitioning Trees", *Proceedings of Graphics Interface '92*, May 1992, pp. 11–18.

[11] Charles K. Chui, *An Introduction to Wavelets*, Academic Press, Boston, 1992.

[12] Shigeru Muraki, "Volume Data and Wavelet Transform". *IEEE Computer Graphics and Applications*, Vol. 13, No. 4, July 1993, pp. 50–56.

[13] H. Edelsbrunner, "Weighted Alpha Shapes", Technical Report Rept. UIUCDCS-R-92-1760, Comput. Sci. Dept., Univ. Illinois, Urbana, Illinois, 1992.

[14] G. Borgefors, "Distance Transformations in Arbitrary Dimensions", *Computer Vision, Graphics, and Image Processing*, Vol. 27, 1984, pp. 321–345.

[15] K. Knowlton, "Computer-aided Definition, Manipulation and Depiction of Objects Composed of Spheres", *Computer Graphics*, Vol. 15, No. 4, December 1981, pp. 352–375.

[16] N. Badler and R. Bajcsy, "Three-dimensional Representations for Computer Graphics and Computer Vision", *Computer Graphics (SIGGRAPH '78 Proceedings)*, Vol. 12, No. 3, August 1978, pp. 153–160.

[17] J. O'Rourke and N. Badler, "Decomposition of Three-dimensional Objects into Spheres", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 1, No. 3, July 1979, pp. 295–305.

[18] D. Herbison-Evans, "Real-time Animation of Human Figure Drawings with Hidden Lines Omitted", *IEEE Computer Graphics and Applications*, Vol. 2, No. 9, November 1982, pp. 27–33.

[19] James F. Blinn, "A Generalization of Algebraic Surface Drawing", *ACM Transactions on Graphics*, Vol. 1, No. 3, July 1982, pp. 235–256.

[20] Shigeru Muraki, "Volumetric Shape Description of Range Data using "Blobby Model"", *Computer Graphics (SIGGRAPH '91 Proceedings)*, Vol. 25, No. 4, July 1991, pp. 227–235.

[21] F. Aurenhammer, "Power diagrams: Properties, Algorithms and Applications", *SIAM Journal of Computing*, Vol. 16, 1987, pp. 78–96.

[22] H. Edelsbrunner, "The Union of Balls and its Dual Shape", *Proc. 9th Ann. Sympos. Comput. Geom.*, 1993, pp. 218–229.

[23] Steven Molnar, John Eyles, and John Poulton, "PixelFlow: High-speed Rendering using Image Composition", *Computer Graphics (SIGGRAPH '92 Proceedings)*, Vol. 26, No. 2, July 1992, pp. 231–240.

[24] David Meyers, Shelly Skinner, and Kenneth Sloan, "Surfaces From Contours: The Correspondence and Branching Problems", *Proceedings of Graphics Interface '91*, June 1991, pp. 246–254.

Glossary

boundary point density — A measure of the closeness of *boundary points*. A boundary point density of δ means that any sphere with a boundary point as its center and radius greater than δ necessarily contains some other boundary point.

Delaunay triangulation — Triangulation which is a dual of the *voronoi diagram* (*qv*).

empty circle property (ECP) — The property of *Delaunay triangulations* which says that the circumscribing circle (sphere) of any triangle (tetrahedron) cannot contain any other vertex.

generality condition — In 2D, the condition that no 3 vertices are colinear, and no 4 vertices lie on a circle. In 3D, the condition that no 4 vertices lie in a plane, and no 5 vertices lie on a sphere.

medial axes — The locus of centers of all maximal spheres inside an object; the skeleton or symmetric axes (surfaces in 3D). **power diagram** — Given a set of vertices, blah...

stability — The property of a representation such as changes in the data induce commensurate and predictable changes in the representation.

Voronoi diagram — Given a set of vertices, their Voronoi diagram is a set of cells so that every point within a cell is closer to one vertex than to any other vertex.