CHAPTER 6

Display Methods for Gray-Scale, Voxel-Based Data Sets

Victoria Interrante, William Oliver,[1] Stephen Pizer, and Henry Fuchs

Department of Computer Science
University of North Carolina
Chapel Hill, North Carolina

The dramatic increase in the use of three-dimensional (3D) image acquisition devices has inspired major new developments in the display of volume data sets. In this chapter we present an overview of these diverse

[1] Present address: Digital Image Processing Laboratory, Department of Cellular Pathology, The Armed Forces Institute of Pathology, Washington, D.C. 20306.

display methods and discuss the relative advantages and disadvantages of each of the different approaches. In addition, we touch on some of the major issues involved in creating high-quality images from volume data, including the problems of surface definition and object segmentation.

Due in part to the rapid, almost frantic, pace of developments in methods for rendering images from volume data, there has not yet emerged any widely accepted taxonomy for these methods. Because the human visual system is adapted for environments in which images of surfaces predominate, most algorithms emphasize in one way or another the display of surface-like information, either implicitly or explicitly. For clarity, we will avoid using the terms *surface rendering* and *volume rendering* to describe the various methods, because although prevalent in the literature they have no precise, commonly accepted definitions. Instead, we will differentiate the various rendering methods using the following three characteristics, which are somewhat more precise and, we hope, less misleading: (1) whether the explicit creation of an intermediate surface representation is required (if so, then we will refer to the method as "indirect"), (2) whether the method is designed to operate on binary or on gray-scale data, and (3) for methods that operate on gray-scale data, whether a binary decision must be made about the existence of a surface at any given location in the volume.

I. INDIRECT METHODS

A. Tiling Methods

One of the earliest approaches used to create images from volume data sets involved the construction of three-dimensional, polygonally tiled surfaces from planar contour curves defined on successive two-dimensional slices through a volume (Keppel 75). In the simplest cases, a single closed contour (either hand drawn or computed using an automatic boundary detection algorithm) is used to outline the object of interest separately on each slice and contour points from adjacent slices are then joined to form triangular tiles. Although a number of different methods have been proposed for determining the connections between contour points, it has been shown that optimal results in the form of a minimum area surface can be obtained when this process is recast in terms of a graph-traversal problem (Fuchs *et al.*, 1977). If speed is more important than quality, heuristic methods can be used to approximate these optimal results (Ganepathy and Dennehy, 1982).

Because these tiling methods allow a user to define surface contours explicitly, using judgment based on the slice data, these methods are still

popular in many applications in which surface localization is a highly sensitive and subjective task. The extraordinary amount of user input required by tiling methods is probably also their greatest drawback, although other problems with this approach certainly exist. Many surfaces are not well represented by the relatively large polygons produced from tiling, and inaccuracies in the surface approximation are exacerbated when the distance between successive contours is particularly large, the number of points in each contour particularly small, or the shapes of adjacent contours especially dissimilar. In addition, surfaces can appear to be skewed or warped if the points between two contours are not connected appropriately. Spline interpolation methods have been used to correct some of these problems by more closely approximating the underlying data both within the between slices, and by allowing a more accurate estimation of surface normals (Sunguroff and Greenberg, 1978). Difficulties also arise in the use of tiling methods to represent surfaces with multiple, complex bifurcations. In these cases, a single slice may contain a number of closed contours that each need to be joined to one or more contours on a neighboring slice, and it is often difficult to determine, without a priori information about the three-dimensional structure of the object being imaged, how the various contours should be connected to represent the surface most accurately. Although some heuristic methods have been developed to handle bifurcations (Christiansen and Sederberg, 1978), user intervention is often required to ensure that the branches have been constructed correctly (Pizer *et al.*, 1986).

B. Variations on Tiling Methods

The problems associated with the conventional tiling approach stimulated research into alternate methods of displaying polygonal surfaces from volume data sets. One variation on these methods, which is particularly well suited for modeling thin, highly complex twisting and branching structures, uses connected segments made of truncated cones a structural primitives (Barillot *et al.*, 1985). Data points are manually selected from the volume in sequence along each structure to be modeled, and a radius value associated with each point is used to generate a circle in the plane defined by the sum of the vectors between the previous and next sample locations along the structure. Truncated cones are then constructed between adjacent samples by tiling between points defined on the circles around each sample, using quadrilateral polygons. Subbranches are created individually by selecting a data point along the axis of an existing structure as the start of a new segment.

A "volume" tiling method was also developed, which for the first time

correctly handled objects containing bifurcations, and objects defined by multiple contours in a single slice plane, in an automatic way (Boissonnat, 1988). In this method, solid objects are represented by selected tetrahedra derived from the three-dimensional Delaunay triangulation of the contour points on adjacent slices through the data. Beginning with a set of contours defined on individual slices, the first step in this method is to ensure that each contour is completely contained in the two-dimensional (2D) Delaunay triangulation of its vertices, by splitting contour segments and adding extra points where necessary. After each of the 2D Delaunay triangulations has been defined, a set of three-dimensional Delaunay triangulations of the combined vertices of two adjacent contours is computed. External and unsolidly connected tetrahedra are then deleted from the representation, and a correction step is applied where necessary to resolve inconsistencies.

C. Indirect, Voxel-Based Methods

In addition to the tiling methods described above, a number of voxel-based methods have been introduced that compute and display surfaces from volume data. Most of these methods assume, for algorithmic simplicity, that all data points in the volume are equidistant. If the data have been sampled at unequal rates in the different dimensions (e.g., if the distance between data points within a slice is less than the distance between data points in adjacent slices), then some kind of interpolation is generally used to obtain the necessary intermediate values. In most gray-scale data sets, the value at a voxel represents a point sample of a continuous three-dimensional distribution, which we will refer to as "intensity" (not to be confused with pixel intensity in a 2D-rendered image). The quality of the rendered images produced by the various voxel-based algorithms will depend to a great extent on the methods used to reconstruct this underlying distribution.

In the cuberille method (Herman and Liu, 1977), the volume occupied by each voxel is modeled as a cube of constant intensity consisting of six polygonal faces, and object surfaces are constructed from appropriately selected connected subsets of these cube faces. Despite the large number of primitives, images can be displayed fairly quickly once the set of cube faces comprising the surface has been found, because the shading and hidden surface computations can be optimized to take advantage of the regularity of the polygonal representation (Herman and Liu, 1979). The quality of an image produced by this method will be highly dependent on the accuracy of both the surface localization and the estimation of surface

normals, and progress has been made in both of these areas since the method was first introduced.

A variety of algorithms has been developed for choosing the desired set of cube faces. In one approach, closed surfaces are found using a sequential search from an initial boundary element; subsequent boundary elements are selected from the neighbors of the current element that satisfy certain criteria based on voxel density or gradient magnitude (encoded in a continuously updated global decision variable), and backtracking is employed where necessary to reduce the likelihood of error propagation (Liu, 1977). An alternate approach, which requires that the data first be transformed by thresholding or by some other segmentation process into a binary-valued volume, uses a sequential search of a different kind to extract closed surfaces. After the set of cube faces that separates the two voxel types has been isolated, a directed graph can be constructed from this set by taking each cube face as a node and each edge as an arc, with at most two incoming and two outgoing arcs being attached to each node. Beginning from an initial face element, a connected surface can then be defined as the subset of cube faces visited during a traversal of this graph (Artzy *et al.*, 1981).

Because the cuberille method relies on an intermediate representation in which polygons may appear in only one of three possible orientations, the images produced by this method will exhibit sharp discontinuities in surface brightness if shading is based solely on the polygonal approximation. Attempts to mitigate the effects of these blocky artifacts by low-pass filtering the final image, antialiasing using subpixel sampling, or averaging surface normals from adjacent polygons (Chen *et al.*, 1985) have met with little success.

Better results have been achieved with a variation of the cuberille method (Udupa and Hung, 1990), in which the cube faces comprising the surface are not restricted to lie exactly on voxel boundaries and the cube normals used in shading the surface are derived from the underlying voxel data alone, ignoring the cuberille approximation.

Whereas the cuberille methods typically use linear interpolation to produce cubic voxels in the gray-scale data before segmentation, an alternate method performs the segmentation first and then interpolates. Beginning with contours defined on slices through the volume, this approach uses "dynamic elastic interpolation" to generate contour curves on the required number of intermediate slices, taking higher order continuity into consideration and incorporating global information into the interpolation process (Lin *et al.*, 1988; Chen *et al.*, 1990). A solid model of the object is created by first filling in each contour and then stacking all of the contours together; the result is a binary voxel representation that can be displayed

by any number of rendering methods. An advantage of this approach is its ability to create smoothly varying surfaces between widely separated, disparate or branching contours, once the correspondences between the contours have been established.

D. Indirect, Subvoxel-Based Methods

One of the most popular voxel-based methods is the "marching cubes" algorithm (Lorensen and Cline, 1987), which uses subvoxel-sized triangular elements to represent isovalue surfaces in a volume. Unlike other methods, in which each data point, or voxel, is represented as a small cube, the basic unit of operation in this algorithm is the cubic volume defined between eight neighboring data samples. Each such cube in the data set is examined, either sequentially or in parallel, and if the desired isovalue surface is found to be contained within that volume, a triangular tesselation of the enclosed portion of the surface is derived. As a first step, the voxel intensity (or some precomputed object inclusion likelihood value) at each of the eight cube vertices is compared with a threshold value to determine whether that voxel lies inside or outside of the desired surface. The cube is then assigned 1 of 256 possible index values based on the configuration of its vertices. These 256 cube configurations can be mapped by rotation into 15 topologically distinct arrangements, and a triangular tesselation is defined for each of these canonical cases. Vertex coordinates are computed for each triangle in the tesselation by linearly interpolating between the end points of the enclosing cube edge, according to the difference between the threshold value and the values at each of the end points. To achieve the best possible results in shading, normals at the triangle vertices are computed not from the polygonal approximation of the surface, but rather from the intensity gradients of the underlying voxel data; gradient values found at each of the eight cube vertices, using central differences, are linearly interpolated along cube edges to the triangle vertex locations.

To maximize rendering efficiency for data sets in which the number of triangles is large and the screen area covered by each triangle is small, a variation of the marching cubes algorithm has been developed that uses point primitives instead of triangles in the surface representation (Cline *et al.*, 1988). In this "dividing cubes" approach, the cube formed between eight neighboring voxels is subdivided until the area projected by any subcube face fits within a single pixel on the screen. Voxel intensity values are interpolated to the subcube vertex locations, and subcubes are then classified as interior, exterior, or intersecting the isovalue surface,

depending on the intensity values at their vertices. If a subcube is identified as intersecting the surface, a point primitive is defined at its center and a gradient value is interpolated to that point from the gradients at the vertices of the enclosing undivided cube.

An extension of this marching cubes approach has also been developed to represent isovalue surfaces in arbitrarily shaped elements, using bicubic patches (Gallagher and Nagtegaal, 1989).

The marching cubes algorithm has a number of features that recommend it. Because subvoxel elements of arbitrary orientation are used in the polygonal representation, surfaces produced by this method generally fit the underlying data more closely than do surfaces obtained through either a cuberille or tiling approach, and virtual memory requirements are kept to a minimum because the algorithm operates on only two slices of data at a time. A marching cubes surface will usually take longer to render than a cuberille or tiled surface, however, because of the larger number of polygons and also because of the loss of regularity in polygonal orientation that allowed simplified shading and hidden surface computations to be used in the rendering of cuberille data.

The greatest drawback of the marching cubes approach is that surfaces in a volume are generally located on the basis of thresholding alone, which in many cases is not a sufficient criterion for boundary detection and does not guarantee connectivity. Because surfaces are generated in response to all threshold crossings, regardless of context, isolated voxels are independently free to spawn spurious surfaces in an image when the threshold value is set too low and desired surfaces may become fragmented when the threshold value is set too high. These problems can be alleviated somewhat by applying a more sophisticated surface detection algorithm to the data, and using these surface likelihood values rather than the original voxel values in the marching cubes algorithm. An additional problem is that ambiguities exist in the marching cubes tesselation of the surface within cubes in which the threshold value is exceeded only at diagonally opposing vertices across a face. If not handled correctly, these ambiguities can cause tiny holes to appear in the surface where none should exist (Wilhelms and Van Gelder, 1990a).

It has been shown that the efficiency of a marching cubes method can be substantially increased if an octree traversal of the data is used instead of a sequential approach, allowing large areas of the volume that lie inside or outside of the isovalue surface to be bypassed in one step rather than on a cube-by-cube basis (Wilhelms and Van Gelder, 1990b).

Semitransparent polygons have also been used to model volume data, in some of the direct projection and splatting algorithms discussed in Section V. It should be noted, however, that because of the need for compositing, the time required to render a set of semitransparent poly-

gons will always be greater than that required to render an opaque polygonal data set of similar size.

II. DIRECT METHODS FOR DISPLAY OF SURFACES IN BINARY DATA

In the mid-1980s a number of methods for visualizing volume data sets were proposed, which did not require the extraction of surfaces as a first step.

A. Octree Methods

One of the earliest direct methods uses octree encoding (Meagher, 1982) to model objects defined by a binary-valued volume obtained from the gray-scale data through thresho'ding or some other segmentation procedure. Octree methods achieve data compression by storing voxel information in a hierarchical tree structure, which is built in a top-down fashion by recursively subdividing inhomogeneous regions of the volume into eight subregions until each terminal node of the tree corresponds to a region of the volume in which all voxels share the same value. The original goal of the octree approach was to allow more rapid manipulation and display of complex three-dimensional objects than was currently possible using polygonal models. Efficient integer algorithms were developed to implement Boolean operations for cutaway views, and to perform geometric transformations such as rotation, translation, and scaling on octree-encoded data; hidden surface views of octree-encoded objects can also be generated quickly for any observer position, due to the spatially presorted format in which the data is stored. In addition, the hierarchical octree structure is useful in successive refinement for generating images at varying resolutions using data from intermediate levels in the tree. The primary disadvantages of octree-encoding binary data are the time it takes to build the tree and the complexity of the tree-traversal operations. In addition, the octree representation of a data set is highly dependent on the initial position and orientation of the data; two data sets that differ only by a small translation or rotation may have substantially different octree representations.

Images can be generated from an octree data set by establishing a correspondence between pixels in the image and face-connected strings of voxels in the volume, then recursively traversing the octree in a back-to-front order and, whenever a nonempty voxel is encountered, overwriting the corresponding pixel value with the color of the current voxel (Doctor

and Torborg, 1981). Pseudoshading can be implemented by defining top, bottom, left, and right illumination factors that are added to the color of a voxel when its neighbor in the corresponding direction is empty. Front-to-back octree display algorithms have also been developed for octree data sets (Meagher, 1982).

B. Object–Space Methods

Algorithms that create images by projecting volume information onto the image plane are commonly referred to as "object–space" methods. An important feature of these algorithms is that they do not require that the complete data set be resident in memory at all times. One of the earliest object–space approaches renders images from binary-valued voxel data sets by traversing slices of the arbitrarily oriented volume from back to front, beginning within each slice at the most distant corner and progressing along either rows or columns, projecting a value derived from the depth of each voxel perpendicularly onto an individual pixel of the final image (Frieder *et al.*, 1985). This method, although straightforward, is not optimal for a variety of reasons. The resulting images also suffer from the same shading problems that were encountered in their cuberille approach and, because a given pixel may be colored and recolored many times before a final value is assigned, efficiency is sacrificed. In addition, this method is limited to performing parallel planar projections; a perspective view can be achieved only by prewarping the volume data before rendering is begun. Moreover, because of the poor quality of the reconstruction algorithm, small hole artifacts can appear in the rendered image when the volume is viewed at certain angles or magnifications.

If voxel data are projected onto the image plane in order of increasing distance from the viewer, only the first value received by each pixel needs to be retained and all subsequent attempts to write to the same pixel must be disallowed. A front-to-back version of the above method, which produces equivalent images in less time, avoids pixel rewrites through the use of a dynamic data structure that keeps track of unwritten scan line segments (Reynolds *et al.*, 1987). The algorithm first ensures that face-connected voxels in a single row of the volume will project onto adjacent pixels in a single scan line of the two-dimensional image by requiring that all rotations be done in a specific sequence. As each row of voxels is projected onto the image plane, the end points of the newly generated spans can then be compared with the end points of the unwritten portions of the scan line, which eliminates the need to test for overwrites at each individual pixel.

C. Image–Space Methods

In addition to the object–space methods described above, image–space methods, which render the data on a pixel-by-pixel basis, have also been developed to display solid objects defined by binary-valued voxel data sets. The simplest of these methods (Tuy and Tuy, 1984) steps through the volume along rays cast perpendicularly from the center of each pixel until the first nonempty voxel is encountered, shading pixels of the final image with a value based primarily on their distance from this surface voxel. When the viewing direction is not aligned with an axis of the volume, nearest-neighbor interpolation is used to compute the values encountered at discrete intervals along the ray.

One advantage of the ray-casting approach is that it does not require a complete traversal of the volume; processing can be stopped as soon as a surface voxel is encountered along the ray from each pixel. A disadvantage of this approach is that in order to generate images at arbitrary orientations the entire data set must generally be accessible in memory at all times. Techniques have been developed that reduce the time required to render a series of images by exploiting the coherence between consecutive frames of a rotation sequence (Rhodes *et al.*, 1987).

III. DIRECT BINARY METHODS FOR DISPLAY OF SURFACES IN GRAY-SCALE DATA

While most early methods were designed to operate on binary-valued voxel data sets, in which surfaces have been predefined using some type of segmentation procedure, more recent methods have moved toward integrating segmentation into the rendering process. The motivation for doing this is twofold. First of all, retaining the gray-scale information at each voxel allows the use of shading models that more accurately reflect the orientation of underlying object surfaces. Because the values found at each voxel typically represent point samples of a smoothly varying function in three dimensions, shading methods that base their estimate of surface orientation on a zeroth-order reconstruction in which the volume is represented by a set of cubes of uniform intensity are clearly not optimal. A much better approximation to the surface orientation at a boundary voxel is given by the gradient of gray-scale values across the voxels in a 3×3 neighborhood (Hoehne and Bernstein, 1986). In addition, surface position can be estimated with greater precision when the possibilities are not limited to specific locations on a regular grid.

An investigation of shading algorithms for gray-scale data has shown that of the methods currently in use, a technique called "adaptive gray-

level gradient shading" (Pommert *et al.*, 1989) produces the smoothest images with the least amount of error in surface normal estimation (Tiede *et al.*, 1990). This method operates by adaptively selecting the neighborhood over which gradients are computed to more accurately represent surface normals on very thin objects. For example, the partial derivative in the x direction at a voxel (i, j, k) is normally computed at $\rho(i + 1, j, k) - \rho(i - 1, j, k)$. In this method, if $\rho(i, j, k) > \rho(i + 1, j, k)$ and $\rho(i, j, k) > \rho(i - 1, j, k)$, then the voxel is interpreted as being part of a very thin surface and the partial derivative is computed as $\rho(i, j, k) - \max\{\rho(i + 1, j, k), \rho(i - 1, j, k)\}$.

One method that produces high-quality images of surfaces detected by thresholding in a gray-scale volume uses gray-scale gradients to approximate surface normals and linear interpolation to resample the volume data (Hoehne *et al.*, 1987). Rays are projected perpendicularly from the image plane, which is kept aligned with the data volume, until a voxel is encountered whose value exceeds the specified threshold. Rotated or perspective views are generated by a preprocess in which the entire volume is transformed and then resampled before any rays are cast. Because gray-scale information is retained at each voxel, it is possible to generate hybrid images in which raw voxel values are displayed on selected cut planes in combination with a surface representation of the data. In this case, voxel values are sampled along each ray until the cut plane is encountered, and if the gray-scale value at the voxel intersected by the cut plane exceeds the surface threshold, this gray-scale value is assigned to the pixel of the final image. Otherwise, sampling continues along the ray until a surface voxel is hit and the image pixel is assigned a color derived from the distance and orientation of the surface at that point.

An extention of this method renders surfaces detected by arbitrarily complex segmentation procedures through the use of a data structure called the "generalized voxel model" (Hoehne *et al.*, 1988), in which an array of values is stored at each data location in the volume. This technique can also be used to generate combined images using information from multiple, registered data volumes.

One approach in which more elaborate surface detection criteria are used is called "active ray tracing" (Trousset and Schmitt, 1987), an ad hoc technique that uses local information about voxel intensity and gradient magnitude to locate boundaries in a gray-scale volume. Rays are cast through the data in a direction orthogonal to the slice axis (thus ensuring that each ray will be completely contained within a single slice) and, as each ray penetrates the volume, the intensity values of the voxels it passes through are compared with predetermined upper and lower threshold bounds. A connected subset of voxels along the ray whose intensity values fall within the specified range is classified as a candidate boundary segment. If the potential boundary indicated by this segment is deter-

mined not to be part of a surface already detected by previous rays, it is judged to represent either the start of a new surface or a piece of noise, depending on an estimate of the area of the potential surface. Once a candidate segment has been validated, the location of the surface within that segment is chosen. If the gradient magnitude is not sufficiently large, either at an individual voxel or across the connected set of voxels, gradient information in the candidate segment is regarded as unreliable and the boundary is simply placed at the first voxel whose intensity falls within the threshold range. When the gradient information is judged to be reliable, however, then for a new surface the boundary is placed at the voxel of largest gradient magnitude in the candidate segment, or for an existing surface at the voxel of largest gradient magnitude whose gradient direction is compatible with the direction of the surface.

IV. DIRECT BINARY METHODS FOR DISPLAY OF INTENSITY DISTRIBUTIONS IN GRAY-SCALE DATA

Most of the methods presented so far are designed to display surfaces in gray-scale data sets. Although the majority of algorithms that attempt to portray volume intensity distribution information are nonbinary and are discussed in the next section, a direct binary method has been developed that uses color to represent different intensity ranges within a volume. Before three-dimensional surface or volume-rendering methods came into widespread use, volume data were typically viewed on a slice-by-slice basis. A display method that builds on this slice-based approach forms images using stacks of two-dimensional slices (Farrell *et al.*, 1984). Data are extracted from the volume along a user-defined axis to produce the set of slices, and on each slice data points are assigned one of several possible colors based on the density range into which they fall. The slices are then projected onto the screen, one at a time from back to front, with each slice slightly offset from its predecessor to produce an oblique view. Only those points falling within the desired density ranges are displayed on each slice, to prevent the unwanted obscuration of data from preceding slices.

V. DIRECT NONBINARY METHODS FOR DISPLAY OF GRAY-SCALE DATA

A. Multiplanar Reconstruction

Multiplanar reconstruction is a two-dimensional method that has been used to visualize gray-scale volume data. This method allows structural details that originally spanned multiple two-dimensional slices along a

given axis to be portrayed in a single slice image, through a process in which the volume is reconstructed from the original slices and a second set of two-dimensional slices is then extracted along a different, orthogonal axis (Glenn *et al.*, 1975).

B. Reprojection Methods

One of the earliest direct, nonbinary methods used to visualize gray-scale volume data is a technique called reprojection (Harris *et al.*, 1979). In this method, planar orthogonal views are generated by projecting the voxel data along parallel paths through the volume into an image buffer, summing the values of the voxels encountered along each path into individual pixels of the image. A stereo pair can be produced when two such images are generated using appropriate parameters for the position and angle of view, but there is no perspective distortion and the image will be equally in focus at all distances into the volume. Obscuring structures can be partially removed in a reprojected image by selectively decreasing the relative contributions of certain voxels, distinguished either by their spatial location or by their intensity value, during the summation process; obscuration can be also avoided to a certain extent through computing the reprojection from a variety of different angles. Although reprojection allows more of the volume data to be seen than do methods that display only surfaces, a significant drawback to this approach is that all depth information is lost during the summation process, making the images look uniformly flat. Figure 6.1 shows a stereo pair of reprojected images of metaphase chromosomes in a human lymphocyte.

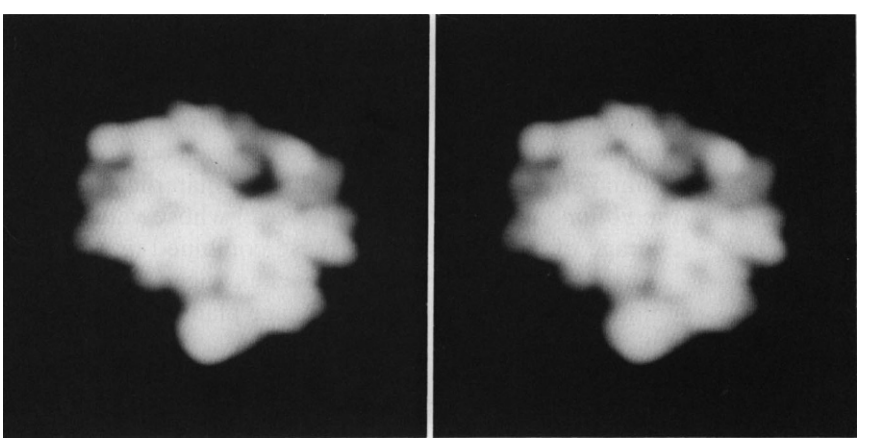

Fig. 6.1. Reprojected images of metaphase chromosomes in a human lymphocyte.

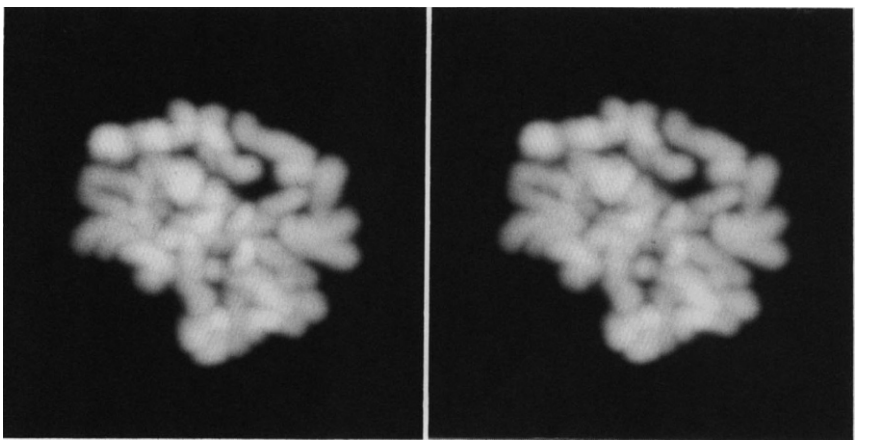

Fig. 6.2. The chromosome data of Fig. 6.1, rendered using the maximum intensity method.

A similar method assigns each pixel of the image a gray-scale value equal to the maximum intensity encountered along the ray emanating from that pixel (van der Voort *et al.*, 1985). Because of its simplicity, this method is frequently used with success for many applications. Images produced by this approach lack depth information, however, and discontinuities may arise when a voxel that contains the maximum value along a ray from one direction is no longer found to be a maximum when the rays are cast from a slightly different direction. Figure 6.2 shows a stereo pair of images generated using the maximum intensity method.

C. Image–Space Methods

Methods have also been developed that display surfaces in gray-scale volumes without requiring a binary decision to be made about the exact surface location. These methods are based on a recognition of the "partal volume effect," in which a mixture of materials may be contained within the volume covered by a single voxel whose value represents a weighted average of the intensities of these combined materials.

One of the earliest such approaches traced perspective rays from an eye point through the pixels of the image plane into the volume data, computing a color, transparency, and filter value for each voxel encountered, using a look-up table indexed by voxel intensity (Schlusselberg and Smith, 1986). The filter value was used to define shading characteristics and to implement masking, and transparency calculations were accumulated along a ray until the first completely opaque voxel was encountered.

Although the approach was promising, image quality was compromised by the use of a nonoptimal nearest-neighbor interpolation technique and by undersampling in the rear portions of the volume due to ray divergence.

High-quality images were produced several years later by two similar, independently developed rendering algorithms. The first of these methods (Levoy, 1988a) uses a multistep approach to display smooth-looking surfaces from gray-scale, voxel-based data. If the data in the original volume are not equally spaced in all three dimensions, the volume is first resampled, using B-spline interpolation to obtain the necessary data resolution. A "surface normal" is then computed for each voxel in the volume, based on the gray-scale gradients at that point. Classification ramps are set up that define opacity as a piecewise linear function of both voxel intensity and gradient magnitude (the latter is used as a surface likelihood indicator), and the results are stored in look-up tables that are used in later steps to assign an opacity value to each voxel in the volume. A color value is also computed for each voxel, using a Phong shading model based on the estimated surface normal, surface distance, and light position. Finally, parallel rays are cast from each pixel in the image through the computed color and opacity volumes, and trilinear interpolation is used to sample these values at equal intervals along each ray. The color value is premultiplied by the opacity at each sample, and both color and opacity are composited from front to back along each ray until either the opacity has accumulated beyond a specified cutoff value or the data have been exhausted. Finally, the color accumulated along each ray is assigned to the corresponding pixel of the image. Although a back-to-front compositing order was first proposed, this front-to-back approach achieves greater efficiency by allowing the adaptive termination of rays (Levoy, 1990b). The stereo images in Fig. 6.3 were produced using the Levoy algorithm.

Isovalue surfaces in gray-scale volume data can also be rendered using a variation of this direct, nonbinary approach (Levoy, 1988b). Whereas a binary approach would merely assign an opacity α_v to voxels having the selected intensity value ρ_v, this method creates a smoother surface by also assigning voxels whose values are close to ρ_v opacities that are close to α_v. To avoid holes while ensuring that the surface does not become too wide, opacity values are defined to fall off with the difference from the target intensity at a rate inversely proportional to the magnitude of the gradient at the surface.

Geometric primitives such as polygons can be integrated into the final images produced by either of these methods through the use of a hybrid ray-casting algorithm (Levoy, 1990a).

An extension of this ray-casting method has been developed that allows the production of high-quality perspective images (Novins *et al.*, 1990). Because perspective is most useful as a depth cue when used on data

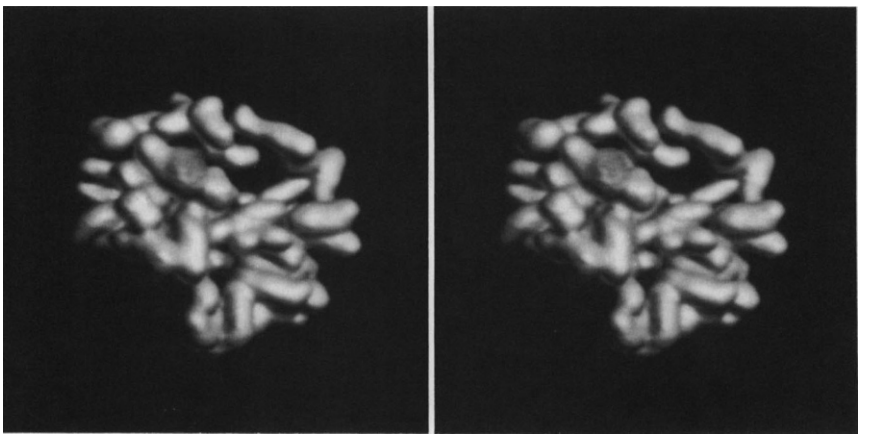

Fig. 6.3. The chromosome data of Fig. 6.1, volume rendered using the Levoy algorithm.

containing recognizable geometry, it was originally not implemented in many alorithms designed for the visualization of medical or biological data. With the introduction of geometric cutaways as a technique for viewing raw data in the interiors of objects and with the increasing speed of many displays, which encourages interactive fly-throughs of complex, three-dimensional objects, the need for perspective rendering has increased. For a ray-casting method to handle perspective correctly, care must be taken to ensure adequate sampling in all parts of the volume. The approach taken by this method is to split each ray into four divergent subrays whenever the sampling rate falls below a specified threshold and then to combine the accumulated results from each subray to yield a composite value for the final pixel.

The high quality of the images produced by the methods of Levoy and Drebin *et al.* (discussed below) is primarily due to the effort they take to avoid any unnecessary introduction of aliasing or quantization artifacts. One example is the preservation of continuity in the underlying volume data through the use of cubic or linear interpolation for resampling rather than a nearest-neighbor method. In addition, by avoiding binary classification these algorithms allow a representation of the uncertainty of data segmentation results. In the nonbinary approach, voxels in the vicinity of a surface boundary will have an opacity proportional to the likelihood of their actually being a part of this surface. One drawback with these algorithms, which applies slightly more to the first method than to the second, is the difficulty of defining appropriate classification and shading functions. This can be a nonintuitive process, requiring several iterations to "get right," and no one has yet been able to measure objectively the adequacy of the parameters selected.

A different nonbinary ray-casting approach is used in a technique referred to as "simulated fluorescence" (van der Voort *et al.*, 1989), a two-step algorithm that models the excitation and emission phases of a simplified fluorescence process. The first step in this method is to compute the amount of "exciting radiation" that is absorbed by each voxel in the data set. A set of parallel rays is cast through the data from the excitation direction and the intensity value at each voxel encountered along a ray is multiplied by the radiation intensity at that point. Radiation is attenuated at each step along the ray by an amount derived from the intensity of the voxel through which it has passed. The second step in the method is to compute the amount of "emitted radiation" reaching each pixel of the final image. A second set of parallel rays is cast through the data from the viewing direction and the excitation values are composited from back to front, using voxel intensity to regulate the attenuation. This method produces three-dimensional-looking images without making any attempt to extract surface information, and the incorporation of shadows into the shading process heightens the perception of depth.

A similar approach, but one that uses gradients of voxel intensity for shading, is described by the Heidelberg ray-tracing model (Meinzer *et al.*, 1991).

D. Object–Space Methods

At the same time that Levoy's image–space method was being developed, a high-quality object–space approach was also proposed (Drebin *et al.*, 1988) that produces similar results. In this approach, which also requires that the voxel data be evenly spaced in all three dimensions, the image plane is kept aligned with the data grid, and rotated or perspective views are generated by resampling the volume. Rotating by resampling the entire data set will in general be more expensive than interpolating values as needed along each ray, because most rays will in general be traced only partway through the volume. However, maintaining alignment between the image plane and the volume allows processing to be confined to a well-defined subset of the data, substantially reducing virtual memory requirements. The first step in generating images with this method is to classify voxels into material types on the basis of their intensity values. Separate volumes are created for each material type, in which the value stored at each voxel represents the percentage of material present in the voxel. This classification is nonbinary, and reflects the fact that a single voxel may contain a mixture of materials. Color, opacity, and "density characteristic" values are associated with each material, with the gradient magnitude of the density characteristic used as a measure of surface strength. A shaded volume is then computed using these parame-

ters, and the resulting voxel values are composited from back to front to produce the final image.

To reduce greatly the rendering time required by this method, the surface detection and shading steps can be omitted to produce reprojected ("unshaded") images (Ney *et al.*, 1990). In both approaches, the rotation/resampling step is usually performed just prior to compositing, so that multiple views of the data can be generated from a single intermediate volume.

A different kind of projection algorithm was designed to produce high-quality images from low-resolution data sets, exploiting coherence in the data to increase the speed of rendering (Upson and Keeler, 1988). In this algorithm, the basic unit of operation is the cell defined between eight corner voxels, and cells are processed in a front-to-back order with the results composited at each pixel to form the final image. Color and opacity values are first assigned to each voxel by a shading procedure, and these values are assumed to vary as a cubic function within each cell. A bounding box is computed for each cell that defines the area of the screen onto which the cell contents will project, and each scan line segment in this bounding box is subdivided into spans within which the depth of the projected cell will vary linearly. For each pixel in a span, the color and opacity functions are numerically integrated in depth through the projecting cell, along the lines defined by the four corners of the pixel. These four values are then averaged to yield the color and opacity contributions of the cell to that pixel.

A related algorithm (Shirley and Tuchman, 1990) projects tetrahedral cells in a volume, producing a set of semitransparent triangles that can be displayed relatively quickly on a general-purpose graphics workstation. The projection of each tetrahedron is decomposed to between one and four triangles, depending on the orientation of the cell relative to the image plane, and color and opacity values at each triangle vertex are determined the basis of the thickness of the projecting tetrahedron along the line of sight through the vertex and on the data values at the entry and exit points of this ray. Processing is fairly quick, because for each set of projected triangles there will be exactly one (possibly shared) vertex corresponding to a nonzero tetrahedral thickness. An additional feature of this algorithm is that it gracefully handles volumes containing data samples that are not equally spaced on a rectangular grid.

A similar algorithm (Wilhelms and Van Gelder, 1991) operates on volume data that are regularly sampled in each dimension, and achieves greater speed by exploiting the fact that each rectangular cell projects onto the image plane in exactly the same way. This work also includes a discussion of the tradeoffs between image generation time and image quality when different methods are used for interpolation between pro-

jected vertices and for integration in depth through a cell. These faster projection methods do not incorporate directional shading, and they make no attempt to display surface information, resulting in images with a rather fuzzy, cloudlike appearance.

A different object–space technique, called "splatting" (Westover, 1990), reconstructs the intensity distribution in the volume occupied by each voxel and projects these reconstructed data onto an area defined as the image plane "footprint" of the voxel. A Gaussian is usually used for the reconstruction, although other functions might also be appropriate. If a parallel projection is used, the footprint function will be identical for all voxels in the data set and can be precomputed and stored in a look-up table. Each entry in the footprint table will represent the weight of the contribution of a voxel to the subpixel area covered by that part of the footprint, and the size of the footprint table can be varied to trade off image quality for rendering speed. If the data are equally distributed in all three dimensions, the footprint will be circular, but unequal spacing can be easily handled by stretching the footprint into an ellipse. Once each voxel is assigned a color and opacity according to some shading algorithm, these values are projected, using the footprint function, into an accumulation buffer for each slice. After all voxels in a slice have been projected, the values in the accumulation buffer are composited into the final image. Processing can proceed in either front-to-back or back-to-front order, and the intermediate results can be displayed interactively, making it possible to view the image as it is being constructed.

A similar method, which operates on a hierarchical representation of the volume, uses semitransparent polygons to approximate the footprint functions (Laur and Hanrahan, 1991). The resulting images are not of as high quality, but they can be displayed rapidly on a fast polygon-rendering workstation.

The choice of whether to use an object–space or image–space method will depend on a number of factors. Object–space methods are typically concerned with how a single data sample maps onto multiple pixels in the image, whereas image–space methods look at how a single pixel of the image is affected by multiple data samples in the volume. Thus an image–space method such as ray casting could process each pixel in parallel, although it would be difficult to partition the volume between multiple processors because the rays typically require random access to the data. With object–space algorithms it is easy to partition the data, but bottlenecks can occur in accessing the image and care must be taken to maintain correct ordering when compositing.

The desire to enhance the interactivity of these high-quality methods has also led to the increasing use of a technique called "progressive refinement," in which coarser images are displayed at more rapid update

rates while the data are being oriented and higher resolution images are generated in a longer amount of time once the motion has stopped. Ray-casting methods implement progressive refinement by sending fewer rays through the data and interpolating from these results to fill in the remaining pixels. Splatting methods implement progressive refinement by using increasingly smaller reconstruction kernels, which map into smaller footprint functions.

E. Radiation Transport Methods

A physically based approach to rendering images from volume data sets applies concepts from transport theory to the problem of modeling how light interacts with clouds of particles.

The first algorithm to use such an approach (Blinn, 1982) worked with data that are modeled as a distribution of spherical reflecting particles, too small to be individually distinguished, which are randomly positioned in a layer. Images are generated by computing the amount of light that is reflected from the layer in a given direction E after entering from a direction L and hitting one or more particles inside the layer, and also the amount of light that is transmitted through the layer in the direction $-E$. A number of simplifying assumptions, such as uniform particle density, low particle reflectivity, and single scattering, are used in this algorithm to make the problem computationally tractable.

This method was later extended to volumes of particles and implemented as a ray-tracing algorithm, without the restriction to media of low albedo and with higher-order scattering effects included in the computations, to which approximate solutions must be found (Kajiya and von Herzen, 1984).

A subsequent approach achieves greater computational efficiency by assuming a model of varying density emitters, eliminating the shadowing and secondary scattering effects (Sabella, 1988). This method was applied to the visualization of density distributions in three-dimensional solids, with color used to highlight density extrema in the data set by defining hue according to the maximum value encountered along a ray, and with saturation tied to distance for use as a depth cue.

A different formulation of this method operates by probing a data set with a beam of particles, tracing the piecewise linear transfer of these particles through the volume, and generating images from the resulting patterns of absorption and scattering derived using equations from linear transport theory (Krueger, 1990). This approach has the advantage of providing a mathematically rigorous and physically meaningful framework for the definition of visualization parameters, and it has been shown

that many earlier volume-rendering techniques can be understood as specific mappings or approximations of this method.

VI. OBJECT DEFINITION AND SEGMENTATION

In nearly all of the rendering methods discussed so far, some kind of object segmentation is used to emphasize certain parts of the data set—most often in order to display object surfaces. Thus, in many cases, what is shown in a rendering represents a specific model of the data rather than simply the data themselves. Conclusions drawn from the rendered image must carry with them a justification for the specific model used. Recognizing this, we can develop a variety of complex models to demonstrate relationships that could not be shown using only a direct or simple mapping between voxel intensity and pixel color.

For example, if we wish to evaluate the connectivity of the vasculature of a renal glomerulus, we find that the capillary lumens cannot be seen

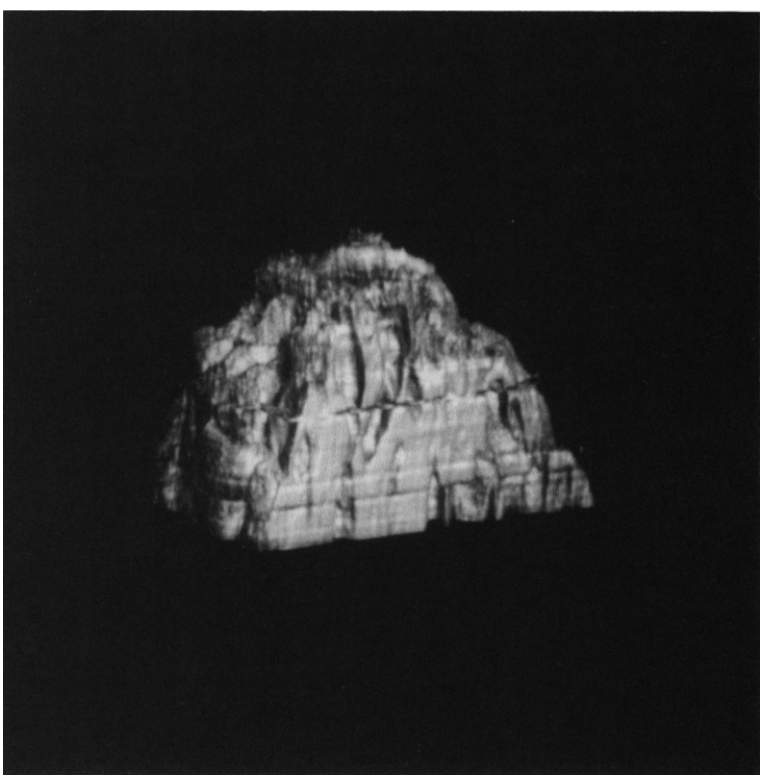

Fig. 6.4. Cutaway image of a portion of a human renal glomerulus.

clearly through the surrounding mesangium and connective tissue. One solution, shown in Fig. 6.4, is to remove a portion of the volume to expose local features in the interior. In another data set, we find that it is impossible to use any simple combination of parameters for surface classification to illustrate clearly the relationship of nucleoli and chromocenters to the nuclear membrane in a human lymphocyte. Because the voxels containing nuclear membrane are of equal or higher intensity than the voxels containing nucleoli and chromocenters, any straightforward rendering would leave the nucleoli obscured by the nuclear rim. One way to visualize both of these structures in a single image, as shown in Fig. 6.5 (Color Section 1), is to remove a portion of the volume with cut planes and display the voxel intensity information on the cut surfaces. Alternately, as shown in Figs. 6.6 and 6.7 (Color Section 1), each structure could be extracted individually from the data volume and then rendered using a different set of parameters to produce a composite image. Such renderings cannot be considered "correct" in the sense of some simple relationship to the original signal, but they can at times be useful in demonstrating certain biological points. The usefulness of this type of rendering is limited, of course, by the accuracy of the segmentation used to extract the portions of the data to be displayed.

To provide the three-dimensional information shown in images like these we must identify, locate, and extract regions of interest (ROIs) from the volume data. Once the ROI is isolated, the results can be used for boundary determination and rendering. Alternatively the ROI mask itself can be rendered, as in Fig. 6.8, or manual determinations of surfaces from the spatial groupings can be done.

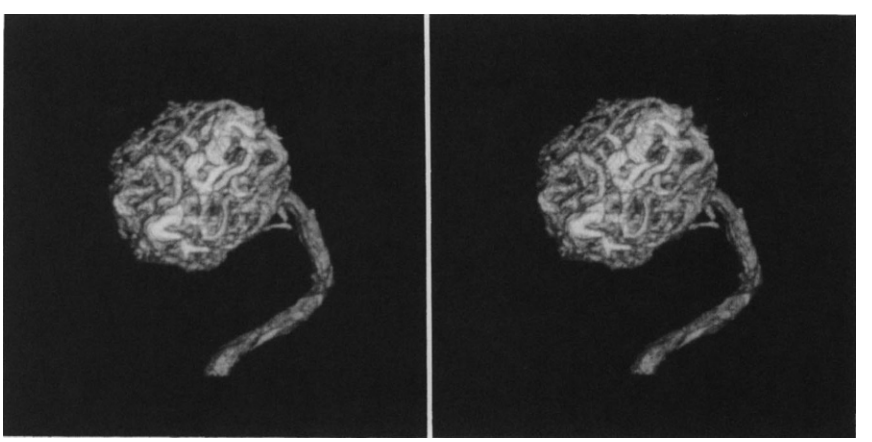

Fig. 6.8. Rendering of a (binary-valued) ROI mask representing the vascular channels of a mouse renal glomerulus, manually segmented from a 3D data set containing an eosin-stained, perfusion-fixed, disaggregated mouse kidney.

The problem of ROI segmentation is related to that of surface segmentation but is nonetheless different. In surface segmentation we are looking for some local measure of "boundariness" that can be used to locate surfaces in the volume. In contrast, ROI segmentation depends more on the object than on its surface, so connectivity and other global aspects need to be considered. Region of interest segmentation is important because it is not always the case that all objects are bounded by a well-defined surface; often the boundariness of any given point in space is best described probabilistically. Surface determination is a different thing than object classification, as a full specification of boundaries may not give a full specification of objects, and vice versa.

A. Primitives

All segmentation strategies rely first on the measure and organization of local geometric primitives. The choice of which local features to exploit depends on the properties of the particular data set being segmented. For instance, if we are working with a volume in which the ROI is clearly delineated by sharp edges, methods that measure edge strength might be most appropriate. In cases in which edges are weak, noisy, or in a complex surround, some other features, such as a measure of local homogeneity of object intensity or texture, might form the basis of the segmentation scheme. In the data set used to generate Figs. 6.5–6.7 (Color Section 1), simple edge-finding techniques might do a quick and easy job of finding nucleoli but fail at isolating chromatin clumps, whereas variable conductance diffusion (discussed in Section V,B) might be able to isolate the chromatin clumps nicely.

Ideally, primitives should be invariant under image rotation, translation, scaling, and illumination. For example, surface curvature measures do not change if one changes the viewing angle or coordinate system representation of a surface. Thus, one might segment nuclei from an image covering numerous cells by detecting their ovoidal shape.

The most basic primitive is simple intensity. Although it can be used directly, as with simple thresholding or template matching, it is not invariant to illumination and for template matching not invariant to rotation. Therefore, most often other measures are derived from it.

Edge strength as a feature has been intensively studied, both because of the importance of edges for delineating regions and because of the biological importance of edge strength in human vision (Levine, 1985). The measure can be as simple as the magnitude of the intensity gradient or it may depend on intensity gradient directions or on properties of contours of isointensity or isoedge strength (Fischler and Bolles, 1986; Asada and Brady, 1986; Pizer *et al.*, 1988). The evaluation of curvature has also been

extensively studied, and Koenderink has demonstrated a biological basis for such measures (Koenderink and van Doorn, 1987). Techniques based on differences of Gaussians or on maxima of gradient magnitude along gradient directions (the Canny method) also fall into this class (Canny, 1986). For both edge strength and curvature, the incorporation of scale (resolution) by Gaussian blurring allows the measurement to match the information in the data by computing only at supportable locational tolerances.

Other geometric features, such as intensity extrema and local shape features of intensity surfaces, have also been investigated and integrated into segmentation algorithms. An example of this is watershed region segmentation. The concept is most obvious in two dimensions, although extensions to three dimensions have also been implemented. In this approach, intensity is used as a topographical third dimension in a 2D image, where bright spots are peaks and dark spots are depressions. This topographical analogy can be extended to define watershed areas, analogous to drainage regions in geographic topography, to produce an image description that connects sets of intensity maxima (hilltops), minima (pits), ridges, and valleys. This connectivity can then be used to define image regions and relationships between image regions (Vincent and Soille, 1991).

Another common primitive is the medial axis, which provides a "skeleton" of the object (Nackman, 1982; Blum and Nagel, 1978). Originally defined for presegmented objects, medial axes have been computed in multiscale, for gray-scale images (Fritch *et al.*, 1991). Medial axis methods segment an object on the basis of the branching pattern of its skeleton, which can be organized in a manner invariant over translation, rotation, and size. These methods are presently most useful when applied to data sets in which simple region definition has already been accomplished and one wants to identify the subset of regions that forms an object ROI.

B. Strategies and Representations

Given a set of primitives and a focus on either surface or object geometry, one must then develop a strategy for organizing these primitives. The most common strategies roughly fall into one of two categories: those that perform some sort of spatial characterization and those that create a model of the object or image and perform an optimization based on the parameters of that model.

In parameter optimization methods, the image intensities or object properties within an object class form a set of parameters to be determined. In the Gibbs/Markov random field (MRF) methods, the parameter

value is intensity and the model (a priori) properties are given by the probability of the intensity of any given voxel, given the intensities of voxels in some local neighborhood. The a posteriori probability also reflects the way the computed segmentation fits the image data. These methods attempt to find the segmentation of the data that gives the maximum a posteriori estimate of the unknown field (Geman and Geman, 1984; Haslett, 1985; Qian and Titterington, 1989; Cohen, *et al.*, 1991). MRF methods work best in areas where the regions of interest are compact and the data are of limited complexity. They suffer from often having to resort to complicated optimization methods with a slow convergence.

In the active surfaces method, also based on optimization, an expression is defined that quantifies the "energy" of a surface relative to its geometric properties and to the way it fits image features, and the segmentation algorithm tries to find a minimum energy configuration (Terzopoulos and Metaxas, 1991; Pentland and Sclaroff, 1991; Terzopoulos *et al.*, 1988). Because they are often sensitive to initial parameter settings, active surfaces methods work best when the shape characteristics of the region of interest are known and when the region is fairly uncomplicated. There have been attempts to combine MRF and active surface methods, using the probability estimates as one of the features in the energy expression (Lin *et al.*, 1991).

A classic parameterized model is template matching via the Hough transform, in which feature matching is done in the parameter space itself rather than in image space (Xu *et al.*, 1990; Risse, 1989). The Hough transform is useful for finding images within a limited range of shape, orientation, and size, because the parameterizations are usually not invariant over these transformations. The image is matched against a family of templates by letting each voxel vote for those templates with which it is consistent. Brakenhoff *et al.* (1988) have used template-matching methods to locate chromosomes in confocal images.

Instead of parameter optimization, one can perform grouping on the basis of spatial characteristics. A common strategy is to blur the image progressively and follow features through the resulting scale space (Witkin, 1983: Crowley and Parker, 1984; Witkin, *et al.*, 1987). Such multiscale methods work well with noisy images, because noise blurs away quickly, and with images containing compact regions without thin connectors. Image features tend to blend together and details tend to disappear when an image is blurred, and one can follow the pattern of how this occurs. A large number of features have been examined in scale space, including edges, medial axes, and watershed regions. Pizer and others have investigated the loss, or annihilation, of features, while Coggins and others have followed the paths of features in scale space (Lifshitz and Pizer, 1990; Lu and Jain, 1989; Dill, *et al.*, 1987; Pizer, *et al.*, 1987;

Coggins, 1990; Lowe, 1988; Kropatsch, 1987). Pyramid methods are similar, but instead of just blurring they reduce spatial sampling at each blurring step (Samet, 1984). A particularly interesting pyramid method is based on the so-called "wavelet" representation (Mallat, 1989; Leitner *et al.*, 1991). Pyramid methods have the advantage of speed, data compression, and support of parallel algorithms, but their rigid resolution reduction causes them to suffer from variance with translation and rotation. Pyramidal approaches that use image-sensitive irregular tesselations have been proposed to reduce the effect of image orientation (Montanvert *et al.*, 1991).

If the region of interest is well demarcated from the surrounding image by some feature (such as an edge), one can modify the amount of blurring that occurs locally in order to take advantage of this feature. An example of such adaptive blurring is presented by Saint-Marc *et al.* (1991), who repeatedly blur small regions by an amount weighted by image feature strength. Similarly, variable conductance diffusion methods "diffuse" features into each other, using the heat equation, with heat conductance modulated by the strength of change in some image feature (Perona and Malik, 1987; Nordstrom, 1989). These methods are iterative and thus can be slow, and they may suffer from a dependence on initial parameters, but they can also be extremely effective in separating objects that are fully surrounded by a locus of adequate change in the feature in question.

In some cases, the difference between regions is based on texture rather than on some easily defined geometric primitive. Here, features may be more easily examined in the frequency domain than in the spatial domain. Examination of the frequency domain alone, however, carries with it the problem of loss of spatial localization. Whitaker (1991) has attacked this problem by performing variable conductance blurring using locally derived frequency space features. Wecshler (1990) argues convincingly that one should look to both the spatial and frequency domains when examining images, and describes a variety of cojoint representations in the space–frequency plane that can be convolved with a kernel in much the same way as the spatial blurring described above, providing a texturally based scale–space representation. Wecshler (1990) shows how operators such as the periodogram, Gabor, Laplacian zero-crossings, and differences of Gaussians can be expressed as cojoint representations in space and frequency.

Moment representations have been of great interest because they are invariant over size, orientation, and position, and a moment tensor-based template matching method described by Cyganski and Orr (1985) has been used to match objects over a wide range of affine transformations in the data (Dirilten and Newman, 1977). Hu (1962) derived a set of moment invariants for use in image segmentation, and these have been extended to

three dimensions by Sadjadi and Hall (1980) and modified by Reiss (1991). Moment representations are best used to recognize and choose among objects that have already been separated from the background image.

Finally, methods that are a hybrid of parameterized optimization and spatial characterization have also been tried. For instance, Bouman and Liu (1991) have proposed a multiresolution Gibbs approach in an attempt to avoid the overemphasis on local features common in classic MRF methods.

C. Region Delineation

Given a collection of primitive regions, perhaps organized into a graph or hierarchy, or a set of primitive feature values, the tactics of delineating the ROI must be considered.

Thresholding is a method that simply selects as the ROI all voxels between two levels of feature strength. Simple thresholding is sensitive to noise and even to smooth spatial variation in feature strength (Parker, 1991; Brink, 1989). However, it is useful when used with features that are measured to avoid just this kind of variation, such as the output of variable conductance diffusion. Thresholding is also useful for quickly "coloring" a large number of voxels that can then be edited by hand. Montag *et al.* (1990) used this method with success in combination with morphological operations (discussed later in this section) for the segmentation of intranuclear chromatin. To produce surface descriptions for surface-based rendering, Brakenhoff *et al.* (1988) used threshold operators on edge-based intermediate images to fit shape models to look for surfaces. Schormann *et al.* (1988; Schormann and Jovin, 1990) proposed using local blurring followed by thresholding for nuclear segmentation in confocal images.

Clustering algorithms attempt to segment a feature space into statistically significant groups. For instance, Baxter and Coggins (1990) cluster voxels according to their intensity paths in scale space, and Manjunath and Chellapa (1991) evaluate the MRF parameters over nonoverlapping small regions of an image and cluster regions on the basis of those parameters. Most clustering methods are sensitive to initial parameterization and can be iterative and slow (Jolion *et al.*, 1991). They are most useful when there are a large number of disparate, relatively uncorrelated parameters being evaluated and when measures of similarity are well defined (Xie and Beni, 1991; Jain and Dubes, 1988; Duda and Hart, 1973).

Merging algorithms take small regions and accrete them into larger regions on the basis of measures of similarity (Hong and Rosenfeld, 1984). Splitting algorithms take large regions and break them into smaller regions on the basis of measures of dissimilarity. These are often combined in an

iterative manner that, the user hopes, will converge to some meaningful representation of regions. They are often used in conjunction with multiscale representations and graph-based methods (Gelfand *et al.*, 1991). Split-and-merge algorithms work well only with fairly simple images and are most often used as a preprocessing step for other methods.

Graph-building methods try to organize primitive regions in a manner recognizing interregion geometry, such as adjacency, containment, directional relations, and watershed relations (Moret, 1982; Chou, 1991). Regions can then be found by graph-matching algorithms matching subgraphs to model graphs. These methods or those in which a decision tree is formed by clustering feature vectors form the basis for most artificial intelligence (rule-based or grammar-based) approaches. Graph-building methods can also be used to connect edges into closed contours that can be used to delineate regions. They are rarely ends in themselves but most often form a scaffolding on which further processing or interactive modification can be done.

Set operations form the basis for methods such as mathematical morphology, which can be used to select regions satisfying certain shape requirements or to edit regions according to their shape (Jang and Chin, 1990; Haralick *et al.*, 1987). Mathematical morphology is applicable both to binary masks and to regions with intensity values. In these methods a kernel or structuring element (often a sphere or cube) is moved through an image, much like a convolution, but here the portion of the image covered by the structuring element is included in or deleted from a region. Mathematical morphological methods are often used as a postprocessing step following application of some other method in order to clean up the data—to impose connectivity, fill in gaps, and so on. A number of common operations have been described, including opening (which tends to delete small objects, break apart thinly connected regions, smooth contours, and such), closing (opening the object complement), skeletonization, thinning, thickening, pruning, and operations to find the convex hull of an object. Variations in the structuring element, including making it a function that is applied to the "earth" under the topographical region intensity surface, allow sensitivity to particular shapes or orientations. The above-mentioned threshold method of Montag *et al.* (1990) applied one level of thinning in a way maintaining connectivity. Schormann *et al.* (1988; Schormann and Jovin, 1990) and Forsgren (1990) also use thresholding in combination with mathematical morphological operations for segmentation of confocal images.

The interest in neural network approaches has been directed toward segmentation, for the most part because it represents a highly parallel implementation of the algorithms already described. For instance, Yamaguchi and Kropatsch (1989) have implemented a neural network in a

pyramid, Hsieh (1989) has used neural network implementations for the extractions of subjective contours, and Pedrycz (1991) has proposed a neural implementation of relational clustering.

D. Interactive Modification

We have pointed out that most methods do not stand on their own, but must be used in conjunction with other methods to solve a specific problem. For instance, a possible sequence of methods to segment nuclear features in a confocal image might consist of (1) thresholding or active contouring to isolate nuclei, (2) edge-finding, active contouring, or watershed analysis to isolate nucleoli, (3) watershed analysis or variable conductance blurring to isolate chromocenters, (4) mathematical morphologic methods to clean up the data, and (5) interactive editing. Furthermore, no single approach will likely work for all problems.

Currently, most methods are best used as tools to make the process of interactive segmentation less difficult and time consuming. For example, when defining boundaries by contouring it is generally easier to edit automatically contoured boundaries than to draw all boundaries from scratch. The same principle exists for object segmentation. To be most useful, then, a segmentation method should be computationally efficient, so that it can be run interactively, with any lengthy preprocessing done before the user becomes involved. Also, the user should be allowed to direct the segmentation process or to modify the results in a manner consistent with the concepts driving the segmentation method and with the needs of the biological problem.

Most biologists and physicians have neither the time nor the expertise to develop new methods of segmentation and will most likely never have the time or resources to develop customized segmentation software. Instead, they must use previously or easily implemented segmentation methods and commercially available or public domain software systems. One example of such a semiautomated system is the Interactive Hierarchy Viewing system developed at the University of North Carolina at Chapel Hill, in which regions are found using an automatically precomputed region containment hierarchy. Masters and Paddock (1990) have used the interactive capabilities of Voxel View in the rendering of confocal images of the rabbit cornea, and Jones *et al.* (1990) describe a 3D interactive segmenter in their processing of confocal images of chromosomes.

In the final analysis, there is as yet no philosopher's stone of segmentation. But the techniques available are useful, and when integrated into a reasonably interactive front end they can help overcome the time-consuming and dreary work of manual segmentation.

VII. CONCLUSIONS

Each of the methods we have described has its own advantages and disadvantages, and the choice of which to use for a particular application will depend on a number of different factors. Some of the issues that need to be considered are the visualization objectives (what kind of information we would like to display), image quality, speed, ease of use, and ease of specification (how difficult it is to achieve desired results, using a particular method). Because indirect methods often require a costly preprocessing step to create the intermediate polygons, it can be difficult to modify interactively the surface representation of data portrayed with these techniques. Nevertheless, indirect methods are still commonly used in applications designed to run on graphics workstations that specialize in fast polygon rendering. Direct methods that use segmentation to produce a binary volume before rendering sacrifice surface orientation information, making it difficult to obtain smoothly shaded images and impossible to represent detail at subvoxel resolution. In addition, any method that makes a binary decision about the location of a surface is more likely to give unsatisfactory results when applied to data sets in which surface boundaries are not well defined; nonbinary methods handle this problem by reflecting the uncertainty of an edge decision through the use of partial transparency, at the cost of a certain amount of fuzziness in the final image. However, methods that render opaque objects will generally run much faster than methods in which the final pixel values are affected by multiple voxels or primitives.

ACKNOWLEDGMENTS

The code used to render the images in Fig. 6.1–6.4 and 6.8 was written by Marc Levoy, and extended by the authors to generate Figs. 6.5–6.7 in Color Section 1. The nuclear data in Figs. 6.1–6.3 in Figs. 6.5–6.7 in Color Section 1 were supplied by Dr. Gary Smith, obtained with partial funding from NIH-CA24144. Doctor Robert Becker (Armed Forces Institute of Pathology) and Dr. Carlo Pesce (National Institutes of Health) provided the glomerulus data shown in Fig. 6.8. Doctor Charles Jennette provided important aid in the acquisition and preparation of the tissue rendered in Fig. 6.4. Ulrich Neumann offered valuable comments on a draft version of this article.

REFERENCES

Artzy, E., Frieder, G., and Herman, G. T. (1981). The theory, design, implementation and evaluation of a three-dimensional surface detection algorithm. *Comput. Graphics and Image Process.* **15**(1), 1–24.

Asada, H., and Brady, M. (1986). The curvature primal sketch. *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-8**(1), 2–14.

Barillot, C., Gibaud, B., Scarabin, J. M., and Coatrieux, J. L. (1985). 3D reconstruction of cerebral blood vessels. *IEEE Comput. Graphics Appl.* **5**(12), 13–19.

Baxter, L. C., and Coggins, J. M. (1990). Supervised pixel classification using a feature space derived from an artificial visual system. *SPIE Proc. Ser.* **1381,** 459–469.

Blinn, J. F. (1982). Light reflection functions for simulation of clouds and dusty surfaces. *Comput. Graphics* **16**(3), 21–29.

Blum, H., and Nagel, R. N. (1978). Shape description using weighted symmetric axis features. *Pattern Recognition* **10**(3), 167–180.

Boissonnat, J. D. (1988). Shape reconstruction from planar cross sections. *Comput. Vision, Graphics, Image Process.* **44**(1), 1–29.

Bouman, C., and Liu, B. (1991). Multiple resolution segmentation of textured images. *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-13**(2), 99–113.

Brakenhoff, G. J., van der Voort, H. T. M., Baarslag, M. W., Mans, B., Oud, J. L., Zwart, R., and van Driel, R. (1988). Visualization and analysis techniques for three dimensional information acquired by confocal microscopy. *Scanning Microsc.* **2**(4), 1831–1838.

Brink, A. D. (1989). Grey-level thresholding of images using a correlation criterion. *Pattern Recognition Lett.* **9**(5), 335–341.

Canny, J. (1986). A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-8**(6), 679–698.

Chen, L.-S., Herman, G. T., Reynolds, R. A., and Udupa, J. K. (1985). Surface shading in the cuberille environment. *IEEE Comput. Graphics Appl.* **5**(12), 33–43.

Chen, S.-Y., Lin, W.-C., Liang, C.-C., and Chen, C.-T. (1990). Improvement on dynamic elastic interpolation technique for reconstructing 3-D objects from serial cross sections. *IEEE Trans. Med. Imaging* **MI-9**(1), 71–83.

Chou, P. A. (1991). Optimal partitioning for classification and regression trees. *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-13**(4), 340–354.

Christiansen, H. N., and Sederberg, T. W. (1978). Conversion of complex contour line definitions into polygonal element mosaics. *Comput. Graphics* **12**(3) 187–192.

Cline, H. E., Lorensen, W. E., Ludke, S., Crawford, C. R., and Teeter, B. C. (1988). Two algorithms for the three-dimensional reconstruction of tomograms. *Med. Phys.* **15**(3), 320–327.

Coggins, J. M. (1990). A multiscale description of image structure for segmentation of biomedical images. *Proc. Conf. Visual. Biomed. Comput., 1st,* Atlanta, GA, 1990, pp. 123–130.

Cohen, F. S., Fan, Z., and Patel, M. A. (1991). Classification of rotated and scaled textured images using Gaussian Markov random field methods. *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-13**(2), 192–202.

Crowley, J. L., and Parker, A. C. (1984). A representation for shape based on peaks and ridges in the difference of low-pass transform. *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-6**(2), 156–170.

Cyganski, D., and Orr, J. A. (1985). Applications of tensor theory to object recognition and orientation determination. *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-7**(6), 662–673.

Dill, A. R., Levine, M. D., and Noble, P. B. (1987). Multiple resolution skeletons. *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-9**(4), 495–504.

Dirilten, H., and Newman, T. G. (1977). Pattern matching under affine transformations. *IEEE Trans. Comput.* **C-26**(3), 314–317.

Doctor, L. J., and Torborg, J. G. (1981). Display techniques for octree-encoded objects. *IEEE Comput. Graphics Appl.* **1**(3), 29–38.

Drebin, R. A., Carpenter, L., and Hanrahan, P. (1988). Volume rendering. *Comput. Graphics* **22**(4), 65–74.

Duda, R. O., and Hart, P. E. (1973). "Pattern Classification and Scene Analysis." Wiley, New York.

Farrell, E. J., Zappulla, R., and Yang, W. C. (1984). Color 3-D imaging of normal and pathologic intracranial structures. *IEEE Comput. Graphics Appl.* **4**(9), 5–17.

Fischler, M. A., and Bolles, R. C. (1986). Perceptual organization and curve partitioning. *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-8**(1), 100–105.

Forsgren, P. O. (1990). Visualization and coding in three-dimensional image processing. *J. Microsc. (Oxford)* **159**(Pt. 2), 195–202.

Frieder, G., Gordon, D., and Reynolds, R. A. (1985). Back-to-front display of voxel-based objects. *IEEE Comput. Graphics Appl.* **5**(1), 52–60.

Fritsch, D. S., Coggins, J. M., and Pizer, S. M. (1992). A multiscale medial description of greyscale image structure. *SPIE Proc. Ser.* **1607,** 324–335.

Fuchs, H., Kedem, Z. M., and Uselton, S. P. (1977). Optimal surface reconstruction from planar contours. *Commun. ACM* **20**(10), 693–702.

Gallagher, R. S., and Nagtegaal, J. C. (1989). An efficient 3-D visualization technique for finite element models and other course volumes. *Comput. Graphics* **23**(3), 185–194.

Ganepathy, S., and Dennehy, T. G. (1982). A new general triangulation method for planar contours. *Comput. Graphics* **16**(3), 69–75.

Gelfand, S. B., Ravishankar, C. S., and Delp, E. J. (1991). An iterative growing and pruning algorithm for classification tree design. *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-13**(2), 163–174.

Geman, S., and Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-6**(6), 721–741.

Glenn, W. V., Johnston, R. J., Morton, P. E., and Dwyer, S. J. (1975). Image generation and display techniques for CT scan data. *Invest. Radiol.* **10**(5) 403–416.

Haralick, R. M., Sternberg, S. R., and Zhuang, X. (1987). Image analysis using mathematical morphology. *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-9**(4), 532–550.

Harris, L. D., Robb, R. A., Yuen, T. S., and Ritman, E. L. (1979). Display and visualization of three-dimensional reconstructed anatomic morphology: Experience with the thorax, heart, and coronary vasculature of dogs. *J. Comput. Assist. Tomogr.* **3**(4), 439–446.

Haslett, J. (1985). Maximum likelihood discriminant analysis on the plane using a Markovian model of spatial context. *Pattern Recognition* **18**(3/4), 287–296.

Herman, G. T., and Liu, H. K. (1977). Display of three-dimensional information in computed tomography. *J. Comput. Assist. Tomogr.* **1**(1), 155–160.

Herman, G. T., and Liu, H. K. (1979). Three-dimensional display of human organs from computed tomograms. *Comput. Graphics Image Process.* **9**(1), 1–21.

Hoehne, K. H., and Bernstein, R. (1986). Shading 3d-images from CT using gray-level gradients. *IEEE Trans. Med. Imaging* **MI-5**(1), 45–47.

Hoehne, K. H., Delapaz, R. L., Bernstein, R., and Taylor, R. C. (1987). Combined surface display and reformatting for the three-dimensional analysis of tomographic data. *Invest. Radiol.* **22**(8), 658–664.

Hoehne, K. H., Bomans, M., Tiede, U., and Riemer, M. (1988). Display of multiple 3D-objects using the generalized voxel-model. *SPIE Proc. Ser.* **914**(B), 850–854.

Hong, T. H., and Rosenfeld, A. (1984). Compact region extraction using weighted pixel linking in a pyramid. *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-6**(2), 222–229.

Hsieh, C.-H. (1989). A connectionist algorithm for image segmentation. Ph.D. Dissertation, University of North Carolina at Chapel Hill.

Hu, M.-K. (1962). Visual pattern recognition by moment invariants. *IRE Trans. Inf. Theory* **IT-8**(2), 179–187.

Jain, A. K., and Dubes, R. C. (1988). "Algorithms for Clustering Data." Prentice-Hall, Englewood Cliffs, NJ.

Jang, B. K., and Chin, R. T. (1990). Analysis of thinning algorithms using mathematical morphology. *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-12**(6), 541–551.

Jolion, J.-M., Meer, P., and Bataouche, S. (1991). Robust clustering with applications in computer vision. *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-13**(8), 791–802.

Jones, S. J., Taylor, M. L., Baarslag, W., Krol, J. J., Mosterd, B., Mans, A., Brakenhoff, J., and Nanninga, N. (1990). Image processing techniques for 3-D chromosome analysis. *J. Microsc. (Oxford)* **158**(Pt. 2), 235–248.

Kajiya, J. T., and Von Herzen, B. P. (1984). Ray tracing volume densities. *Comput. Graphics* **18**(3), 165–173.

Keppel, E. (1975). Approximating complex surfaces by triangulation of contour lines. *IBM J. Res. Dev.* **19**, 2–11.

Koenderink, J. J., and van Doorn, A. J. (1987). Representation of local geometry in the visual system. *Biol. Cybernet.* **55**(6), 367–375.

Kropatsch, W. G. (1987). Curve representations in multiple resolutions. *Pattern Recognition Lett.* **6**(3), 179–184.

Krueger, W. (1990). Volume rendering and data feature enhancement. *Comput. Graphics* **24**(5), 21–26.

Laur, D., and Hanrahan, P. (1991). Hierarchical splatting: A progressive refinement algorithm for volume rendering. *Comput. Graphics* **25**(4), 285–288.

Leitner, F., Marque, I., Lavallee, S., and Cinquin, P. (1991). Dynamic segmenta-

tion: Finding the edge with snake splines. *In* "Curves and Surfaces" (P.-J. Laurent, A. Le Mehaute, and L. L. Schumaker, eds.), pp. 279–284. Academic Press, Boston.

Levine, M. D. (1985). "Vision in Man and Machine." pp. 151–209. McGraw-Hill, New York.

Levoy, M. (1988a). Direct visualization of surfaces from computed tomography data. *SPIE Proc. Ser.* **914** (B), 828–839.

Levoy, M. (1988b). Display of surfaces from volume data. *IEEE Comput. Graphics Appl.* **8**(3), 29–37.

Levoy, M. (1990a). A hybrid ray tracer for rendering polygon and volume data. *IEEE Comput. Graphics Appl.* **10**(2), 33–40.

Levoy, M. (1990b). Efficient ray tracing of volume data. *ACM Trans. Graphics* **9**(3), 245–261.

Lifshitz, L. M., and Pizer, S. M. (1990). A multiresolution hierarchical approach to image segmentation based on intensity extrema. *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-12**(6), 529–540.

Lin, W.-C., Liang, C.-C., and Chen, C.-T. (1988). Dynamic elastic interpolation for 3-D medical image reconstruction from serial cross sections. *IEEE Trans. Med. Imaging* **MI-7**(3), 225–232.

Lin, W.-J., Pizer, S. M., and Johnson, V. E. (1991). Boundary estimation in ultrasound images. *Proc. Int. Conf. Inf. Process. Med. Imaging, 12th,* Wye, UK, *1991*, pp. 285–299.

Liu, H. K. (1977). Two- and three-dimensional boundary detection. *Comput. Graphics Image Process.* **6**(2), 123–134.

Lorensen, W. E., and Cline, H. E. (1987). Marching cubes: A high resolution 3D surface construction algorithm. *Comput. Graphics* **21**(4), 163–169.

Lowe, D. G. (1988). Organization of smooth image curves at multiple scales. *Proc. Int. Conf. Comput. Vision, 2nd, 1988,* pp. 558–567.

Lu, Y., and Jain, R. C. (1989). Behavior of edges in scale space. *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-11**(4), 337–356.

Mallat, S. G. (1989). A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-11**(7), 674–693.

Manjunath, B. S., and Chellapa, R. (1991). Unsupervised texture segmentation using Markov random field models. *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-13**(5), 478–482.

Masters, B. R., and Paddock, S. W. (1990). Three-dimensional reconstruction of the rabbit cornea by confocal scanning optical microscopy and volume rendering. *Appl. Opt.* **29**(26), 3816–3822.

Meagher, D. (1982). Geometric modeling using octree encoding. *Comput. Graphics Image Process.* **19**(2) 129–147.

Meinzer, H.-P., Meetz, K., Scheppelmann, D., Engelmann, U., and Baur, H. J. (1991). The Heidelberg ray tracing model. *IEEE Comput. Graphics Appl.* **11**(6), 34–43.

Montag, M., Spring, H., Trendelenburg, M. F., and Kriete, A. (1990). Methodical aspects of 3-D reconstruction of chromatin architecture in mouse trophoblast giant nuclei. *J. Microsc. (Oxford)* **158**(Pt. 2), 225–233.

Montanvert, A., Meer, P., and Rosenfeld, A. (1991). Hierarchical image analysis using irregular tesselations. *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-13**(4), 307–316.

Moret, B. M. E. (1982). Decision trees and diagrams. *ACM Comput. Surv.* **14**(4), 593–623.

Nackman, L. R. (1982). Three-dimensional shape description using the symmetric axis transform. Ph.D. Dissertation, University of North Carolina at Chapel Hill.

Ney, D. R., Fishman, E. K., Magid, D., and Drebin, R. A. (1990). Volumetric rendering of computed tomography data: Principles and techniques. *IEEE Comput. Graphics Appl.* **10**(2), 24–32.

Nordström, K. N. (1989). "Biased Anisotropic Diffusion—A Unified Regularization and Diffusion Approach to Edge Detection," Tech. Rep. UCB/CSD 89/514. Computer Science Division, University of California, Berkeley.

Novins, K. L., Sillion, F. X., and Greenberg, D. P. (1990). An efficient method for volume rendering using perspective projection. *Comput. Graphics* **24**(5), 95–102.

Parker, J. R. (1991). Gray level thresholding in badly illuminated images. *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-13**(8), 813–819.

Pedrycz, W. (1991). Neurocomputations in relational systems. *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-13**(3), 289–296.

Pentland, A., and Sclaroff, S. (1991). Closed-form solutions for physically based shape modeling and recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-13**(2), 715–729.

Perona, P., and Malik, J. (1987). Scale space and edge detection using anisotropic diffusion. *IEEE Workshop Comput. Vision,* Miami, FL, *1987,* pp. 16–22.

Pizer, S. M., Fuchs, H., Mosher, C., Lifshitz, L., Abram, G. D., Ramanathan, S., Whitney, B. T., Rosenman, J. G., Staab, E. V., Chaney, E. L., and Sherouse, G. (1986). 3-D shaded graphics in radiotherapy and diagnostic imaging. *Proc. 7th Ann. Conf. Expos. Nat. Comput. Graphics Assoc.,* Vol. 3, pp. 107–113.

Pizer, S. M., Oliver, W. R., and Bloomberg, S. H. (1987). Hierarchical shape description via the multiresolution symmetric axis transform. *IEE Trans. Pattern Anal. Mach. Intell.* **PAMI-9**(4), 505–511.

Pizer, S. M., Gauch, J. M., and Lifshitz, L. M. (1988). Interactive 2D and 3D object definition in medical images based on multiresolution image descriptions. *SPIE Proc. Ser.* **914**(Pt. A), 438–444.

Pommert, A., Tiede, U., Wiebecke, G., and Hoehne, K. H. (1989). Image quality in voxel-based surface shading. *Proc. Int. Symp. Comput. Assist. Radiol., 1989,* pp. 737–741.

Qian, W., and Titterington, D. M. (1989). On the use of Gibbs Markov chain models in the analysis of images based on second-order pairwise interactive distributions. *J. Appl. Stat.* **16**(2), 267–281.

Reiss, T. H. (1991). The revised fundamental theorem of moment invariants. *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-13**(8), 830–834.

Reynolds, R. A., Gordon, D., and Chen, L.-S. (1987). A dynamic screen tech-

nique for shaded graphics display of slice-represented objects. *Comput. Vision, Graphics, Image Process.* **38**(3), 275–298.

Rhodes, M. L., Kuo, Y.-M., Pang, A. T., and Rothman, S. L. G. (1987). An image coherence technique for ray traced three-dimensional presentations in clinical radiology. *Proc. Int. Symp. Comput. Assist. Radiol., 1987,* pp. 615–623.

Risse, T. (1989). Hough transform for line recognition: Complexity of evidence accumulation and cluster detection. *Comput. Vision, Graphics, Image Process.* **46**(3), 327–345.

Sabella, P. (1988). A rendering algorithm for visualizing 3D scalar fields. *Comput. Graphics* **22**(4), 51–55.

Sadjadi, F. A., and Hall, E. L. (1980). Three-dimensional moment invariants. *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-2**(2), 127–136.

Saint-Marc, P., Chen, J.-S., and Medioni, G. (1991). Adaptive smoothing: A general tool for early vision. *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-13**(6), 514–529.

Samet, H. (1984). The quadtree and related hierarchical data structures. *Comput. Sur.* **16**(2), 187–260.

Schlusselberg, D. S., and Smith, W. K. (1986). Three-dimensional display of medical image volumes. *Proc. 7th Ann. Conf. Expos. Nat. Comput. Graphics Assoc.,* Vol. 3, pp. 114–123.

Schormann, T., and Jovin, T. M. (1990). Optical sectioning with a fluorescence confocal SLM: Procedures for determination of the 2-D digital modulation transfer function and for 3-D reconstruction by tessellation. *J. Microsc. (Oxford)* **158**(Pt. 2), 153–164.

Schormann, T., Robert-Nicoud, M., and Jovin, T. M. (1988). Improved stereovisualization method for confocal laser scanning microscopy. *Eur. J. Cell Biol.* **48,** Suppl. 25, 53–54.

Shirley, P., and Tuchman, A. (1990). A polygonal approximation to direct scalar volume rendering. *Comput. Graphics* **24**(5), 63–70.

Sunguroff, A., and Greenberg, D. (1978). Computer generated images for medical applications. *Comput. Graphics* **12**(3), 196–202.

Terzopoulos, D., and Metaxas, D. (1991). Dynamic 3D models with local and global deformations: Deformable superquadrics. *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-13**(2), 703–714.

Terzopoulos, D., Witkin, A., and Kass, M. (1988). Constraints on deformable models: Recovering 3D shape and nonrigid motion. *Artif. Intell.* **36**(1), 91–123.

Tiede, U., Hoehne, K. H., Bomans, M., Pommert, A., Riemer, M., and Wiebecke, G. (1990). Investigation of medical 3D-rendering algorithms. *IEEE Comput. Graphics Appl.* **10**(2), 41–53.

Trousset Y., and Schmitt, F. (1987). Active-ray tracing for 3D medical imaging. *Proc. Eur. Comput. Graphics Conf. Exhib., EUROGRAPHICS '87,* pp. 139–150.

Tuy H. K., and Tuy, L. T. (1984). Direct 2-D display of 3-D objects. *IEEE Comput. Graphics Appl.* **4**(10), 29–33.

Udupa, J. K., and Hung, H.-M. (1990). Surface versus volume rendering: A

comparative assessment. *Proc. Conf. Visual. Biomed. Comput., 1st,* Atlanta, GA, *1990,* pp. 83–91.

Upson, C., and Keeler, M. (1988). V-BUFFER: Visible volume rendering. *Comput. Graphics* **22**(4), 59–64.

van der Voort, H. T. M., Brakenhoff, G. J., Valkenburg, J. A. C., and Nanninga, N. (1985). Design and use of a computer-controled confocal microscope. *Scanning* **7,** 66–78.

van der Voort, H. T. M., Brakenhoff, G. J., and Baarslag, M. W. (1989). Three-dimensional visualization methods for confocal microscopy. *J. Microsc. (Oxford)* **153**(Pt. 2), 123–132.

Vincent, L., and Soille, P. (1991). Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-13**(6), 583–598.

Wecshler, H. (1990). "Computational Vision." Academic Press, Boston.

Westover, L. (1990). Footprint evaluation for volume rendering. *Comput. Graphics* **24**(4), 367–376.

Whitaker, R. T., and Pizer, S. M. (1993). A multi-scale approach to nonuniform diffusion. *CVGIP: Image Understanding* **57**(1), 99–110.

Wilhelms, J., and Van Gelder, A. (1990a). "Topological Considerations in Isosurface Generation," Tech. Rep. 90-14. Computer Research Laboratory, University of California at Santa Cruz.

Wilhelms, J., and Van Gelder, A. (1990b). "Octrees for Faster Isosurface Generation," Tech. Rep. 90–28. Computer Research Laboratory, University of California at Santa Cruz.

Wilhelms, J., and Van Gelder, A. (1991). A coherent projection approach for direct volume rendering. *Comput. Graphics* **25**(4), 275–284.

Witkin, A. (1983). Scale-space filtering. *Proc. Int. J. Conf. Artif. Intell., 8th,* Karlsruhe, West Germany, *1983,* Vol. 2, pp. 1019–1022.

Witkin, A., Terzopoulos, D., and Kass, M. (1987). Signal matching through scale space. *Int. J. Comput. Vision* **1**(2), 133–144.

Xie, X. L., and Beni, G. (1991). A validity measure for fuzzy clustering. *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-13**(8), 841–947.

Xu, L., Oja, E., and Kultanen, P. (1990). A new curve detection method: Randomized Hough Transform (RHT). *Pattern Recognition Lett.* **11**(5), 331–338.

Yamaguchi, T., and Kropatsch, W. G. (1989). A vision by neural network or by pyramid? *Proc. Scand. Conf. Image Anal., 6th,* Oulu, Finland, pp. 104–111. 1989.