Volume Rendering

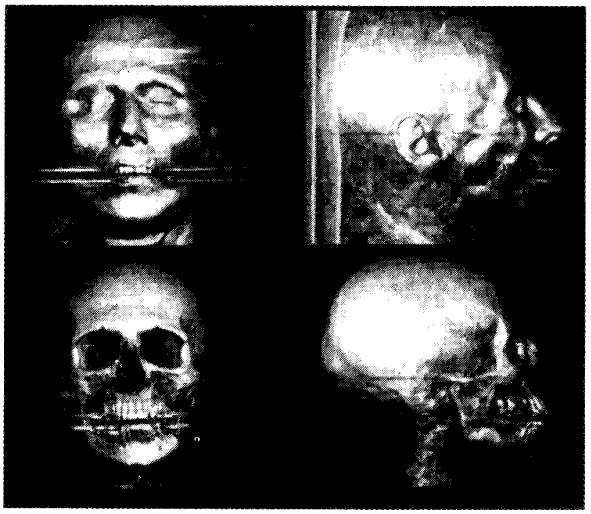
Display of Surfaces from Volume Data

Marc Levoy, University of North Carolina

In this article we will explore the application of *volume rendering* techniques to the display of surfaces from sampled scalar functions of three spatial dimensions. It is not necessary to fit geometric primitives to the sampled data. Images are formed by directly shading each sample and projecting it onto the picture plane.

Surface-shading calculations are performed at every voxel with local gradient vectors serving as surface normals. In a separate step, surface classification operators are applied to compute a partial opacity for every voxel. We will look at operators that detect isovalue contour surfaces and region boundary surfaces. Independence of shading and classification calculations ensure an undistorted visualization of 3D shape. Nonbinary classification operators ensure that small or poorly defined features are not lost. The resulting colors and opacities are composited from back to front along viewing rays to form an image.

The technique is simple and fast, yet displays surfaces exhibiting smooth silhouettes and few other aliasing artifacts. We will also describe the use of selective blurring and supersampling to further improve image quality. Examples from two applications are given: molecular graphics and medical imaging.



Visualization of scientific computations is a rapidly growing application of computer graphics. A large subset of these applications involves sampled functions of three spatial dimensions, also known as volume data. Surfaces are commonly used to visualize volume data because they succinctly present the 3D configuration of complicated objects. In this article we explore the use of isovalue contour surfaces to visualize electron density maps for molecular graphics, and the use of region boundary surfaces to visualize computed tomography (CT) data for medical imaging.

The currently dominant techniques for displaying surfaces from volume data consist of applying a surface detector to the sample array, fitting geometric primitives to the detected surfaces, then rendering these primitives using conventional surface-rendering algorithms. The techniques differ from one another mainly in the choice of primitives and the scale at which they are defined.

In the medical imaging field, a common approach is to apply thresholding to the volume data. The resulting binary representation can be rendered by treating 1-voxels as opaque cubes having six polygonal faces.[1] If this binary representation is augmented with the local gray scale gradient at each voxel, substantial improvements in surface shading can be achieved.[2-5] Alternatively, edge tracking can be applied on each slice to yield a set of contours defining features of interest. Then a mesh of polygons can be constructed connecting the contours on adjacent slices.[6] As the scale of voxels approaches that of display pixels, it

0272-1716/88/0500-0029$01.00 1988 IEEE

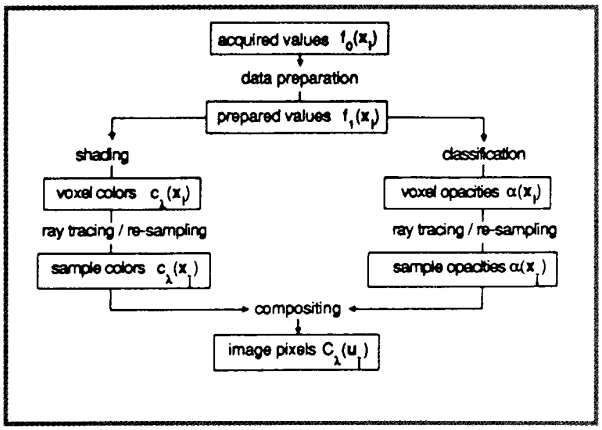


Figure 1. Overview of volume rendering pipeline.

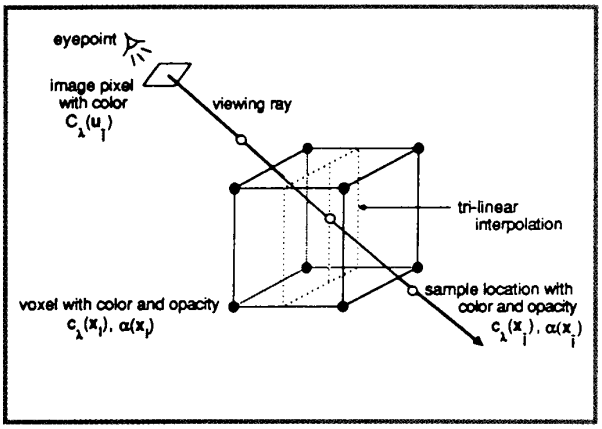


Figure 2. Ray tracing/resampling steps.

becomes feasible to apply a local surface detector at each sample location. This yields a very large collection of voxel-size polygons, which can be rendered using standard algorithms.[7]

In molecular graphics, methods for visualizing electron density maps include stacks of isovalue contour lines, ridge lines arranged in 3-space to connect local maxima,[8] and basket meshes representing isovalue contour surfaces.[9]

These techniques suffer from the common problem of having to make a binary classification decision: Either a surface passes through the current voxel or it does not. As a result, these methods often exhibit false positives (spurious surfaces) or false negatives (erroneous holes in surfaces), particularly in the presence of small or poorly defined features.

To avoid these problems, researchers have begun exploring the notion of *volume rendering*, wherein the intermediate geometric representation is omitted. Images are formed by shading all data samples and projecting them onto the picture plane. The lack of explicit geometry does not preclude the display of surfaces, as we will demonstrate in this article.

The key improvement offered by volume rendering is that it creates a mechanism for displaying weak or fuzzy surfaces. This capability allows us to relax the requirement inherent when using geometric representations, that a surface be either present or absent at a given location. This in turn frees us from the necessity of making binary classification decisions.

Another advantage of volume rendering is that it allows us to separate shading and classification operations. This separation implies that the accuracy of surface shading, hence the apparent orientation of surfaces, does not depend on the success or failure of classification. This robustness can be contrasted with rendering techniques in which only voxels lying on detected surfaces are shaded. In such systems, any errors in classification result in incorrectly oriented surfaces.

Smith has written an excellent introduction to volume rendering.[10] Its application to CT data has been demonstrated by Pixar,[11] but no details of their approach have been published. The technique described in this article grew out of my earlier work on the use of points as a rendering primitive.[12] Its application to CT data was first reported in June 1987,[13] and was presented at the SPIE Medical Imaging II conference in February 1988.[14]

Rendering pipeline

The volume-rendering pipeline used in this article is summarized in Figure 1. We begin with an array of acquired values $f_0(\mathbf{x_i})$ at voxel locations $\mathbf{x_i} = (x_i, y_j, z_k)$. The first step is data preparation, which may include correction for nonorthogonal sampling grids in electron density maps, correction for patient motion in CT data, contrast enhancement, and interpolation of additional samples.

The output of this step is an array of prepared values $f_1(\mathbf{x_i})$. This array is used as input to the shading model described in the section on shading. The output is an array of voxel colors $c_\lambda(\mathbf{x_i})$, $\lambda = r,g,b$. In a separate step, the array of prepared values is used as input to one of the classification procedures described in the section on classification, yielding an array of voxel opacities $\alpha(\mathbf{x_i})$.

Rays are then cast into these two arrays from the observer eyepoint. For each ray a vector of sample colors $c_\lambda(\mathbf{x_i})$ and opacities $\alpha(\mathbf{x_i})$ is computed by resampling the voxel database at K evenly spaced locations $\mathbf{x_i} = (x_i, y_j, z_k)$ along the ray and trilinearly interpolating from the colors and opacities in the eight voxels closest to each sample location, as shown in Figure 2. Finally, a fully opaque background of color $c_{bkg,\lambda}$ is draped behind the dataset and the resampled colors and opacities are merged with each other and with the back-

ground by compositing in back-to-front order to yield a single color $C_\lambda(\mathbf{u}_{\tilde{i}})$ for the ray, and since only one ray is cast per image pixel, for the pixel location $\mathbf{u}_{\tilde{i}} = (u_{\tilde{i}}, v_{\tilde{j}})$ as well.

The compositing calculations referred to above are simply linear interpolations. Specifically, the color $C_{out,\lambda}(\mathbf{u}_{\tilde{i}})$ of the ray as it leaves each sample location is related to the color $C_{in,\lambda}(\mathbf{u}_{\tilde{i}})$ of the ray as it enters and the color $c_\lambda(\mathbf{x}_i)$ and opacity $\alpha(\mathbf{x}_i)$ at that sample location by the transparency formula

$$C_{out,\lambda}(\mathbf{u}_{\tilde{i}}) = C_{in,\lambda}(\mathbf{u}_{\tilde{i}})(1 - \alpha(\mathbf{x}_{\tilde{i}})) + c_\lambda(\mathbf{x}_{\tilde{i}})\alpha(\mathbf{x}_{\tilde{i}}).$$

Solving for pixel color $C_\lambda(\mathbf{u}_{\tilde{i}})$ in terms of the vector of sample colors $c_\lambda(\mathbf{x}_{\tilde{i}})$ and opacities $\alpha(\mathbf{x}_{\tilde{i}})$ along the associated viewing ray gives

$$C_\lambda(\mathbf{u}_{\tilde{i}}) = C_\lambda(u_{\tilde{i}}, v_{\tilde{j}}) =$$

$$\sum_{\tilde{k}=0}^{K} \left[c_\lambda(x_{\tilde{i}}, y_{\tilde{j}}, z_{\tilde{k}})\alpha(x_{\tilde{i}}, y_{\tilde{j}}, z_{\tilde{k}}) \prod_{\tilde{m}=\tilde{k}+1}^{K} (1 - \alpha(x_{\tilde{i}}, y_{\tilde{j}}, z_{\tilde{m}})) \right] \quad (1)$$

where $c_\lambda(x_{\tilde{i}}, y_{\tilde{j}}, z_0) = c_{bkg,\lambda}$ and $\alpha(x_{\tilde{i}}, y_{\tilde{j}}, z_0) = 1$.

Shading

Using the rendering pipeline presented above, the mapping from acquired data to color provides 3D shape cues but does not participate in the classification operation. Accordingly, a shading model was selected that provides a satisfactory illusion of smooth surfaces at a reasonable cost. It is not the main point of this article and is presented mainly for completeness. The model chosen was developed by Phong:[15]

$$c_\lambda(\mathbf{x}_i) = c_{p,\lambda}k_{a,\lambda} +$$

$$\frac{c_{p,\lambda}}{k_1 + k_2 d(\mathbf{x}_i)} \left[k_{d,\lambda}(\mathbf{N}(\mathbf{x}_i)\cdot\mathbf{L}) + k_{s,\lambda}(\mathbf{N}(\mathbf{x}_i)\cdot\mathbf{H})^n \right] \quad (2)$$

where

$c_\lambda(\mathbf{x}_i)$ = λ'th component of color at voxel location $\mathbf{x}_i$, $\lambda = r,g,b$,

$c_{p,\lambda}$ = λ'th component of color of parallel light source,

$k_{a,\lambda}$ = ambient reflection coefficient for λ'th color component,

$k_{d,\lambda}$ = diffuse reflection coefficient for λ'th color component,

$k_{s,\lambda}$ = specular reflection coefficient for λ'th color component,

n = exponent used to approximate highlight,

k_1, k_2 = constants used in linear approximation of depth-cueing,

$d(\mathbf{x}_i)$ = perpendicular distance from picture plane to voxel location $\mathbf{x}_i$,

$\mathbf{N}(\mathbf{x}_i)$ = surface normal at voxel location $\mathbf{x}_i$,

$\mathbf{L}$ = normalized vector in direction of light source,

$\mathbf{H}$ = normalized vector in direction of maximum highlight.

Since a parallel light is used, $\mathbf{L}$ is a constant. Furthermore,

$$\mathbf{H} = \frac{\mathbf{V} + \mathbf{L}}{|\mathbf{V} + \mathbf{L}|}$$

where

$\mathbf{V}$ = normalized vector in direction of observer.

Since an orthographic projection is used, $\mathbf{V}$ and hence $\mathbf{H}$ are also constants. Finally, the surface normal is given by

$$\mathbf{N}(\mathbf{x}_i) = \frac{\nabla f(\mathbf{x}_i)}{|\nabla f(\mathbf{x}_i)|}$$

where the gradient vector $\nabla f(\mathbf{x}_i)$ is approximated using the operator

$$\nabla f(\mathbf{x}_i) = \nabla f(x_i, y_j, z_k) \approx$$

$$\left[\frac{1}{2}\left[f(x_{i+1}, y_j, z_k) - f(x_{i-1}, y_j, z_k) \right], \right.$$

$$\frac{1}{2}\left[f(x_i, y_{j+1}, z_k) - f(x_i, y_{j-1}, z_k) \right],$$

$$\left. \frac{1}{2}\left[f(x_i, y_j, z_{k+1}) - f(x_i, y_j, z_{k-1}) \right] \right].$$

Classification

The mapping from acquired data to opacity performs the essential task of surface classification. We will first consider the rendering of isovalue contour surfaces in electron density maps, i.e., surfaces defined by points of equal electron density. We will then consider the rendering of region boundary surfaces in computed tomography (CT) data, i.e., surfaces bounding tissues of constant CT number.

Isovalue contour surfaces

Determining the structure of large molecules is a difficult problem. The method most commonly used is *ab initio* interpretation of electron density maps, which represent the averaged density of a molecule's electrons as a function of position in 3-space. These maps are created from X-ray diffraction studies of crystallized samples of the molecule.

One obvious way to display isovalue surfaces is to render opaquely all voxels with values greater than some threshold. This produces 3D regions of opaque voxels, the outermost layer of which is the desired isovalue surface. Unfortunately, this solution prevents display of multiple concentric semitransparent surfaces, a very useful capability. Using a window in place of a threshold does not solve the problem. If the window is too narrow, holes appear. If it is too wide, the display of multiple surfaces is constrained. In addition, the use of thresholds and

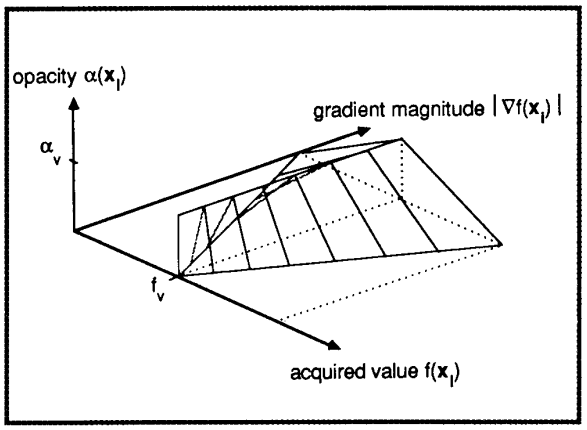


Figure 3. Calculation of opacities for isovalue contour surfaces.

windows introduces artifacts into the image that are not present in the data.

The classification procedure employed in this study begins by assigning an opacity α_v to voxels having selected value f_v, and assigning an opacity of zero to all other voxels. To avoid aliasing artifacts, we would also like voxels having values close to f_v to be assigned opacities close to α_v. The most pleasing image is achieved if the thickness of this transition region stays constant throughout the volume. We approximate this effect by having the opacity fall off as we move away from the selected value at a rate inversely proportional to the magnitude of the local gradient vector.

This mapping is implemented using the expression

$$\alpha(\mathbf{x}_i) = \alpha_v \begin{cases} 1 & \text{if } |\nabla f(\mathbf{x}_i)| = 0 \text{ and} \\ & f(\mathbf{x}_i) = f_v \\ 1 - \frac{1}{r}\left|\frac{f_v - f(\mathbf{x}_i)}{|\nabla f(\mathbf{x}_i)|}\right| & \text{if } |\nabla f(\mathbf{x}_i)| > 0 \text{ and} \\ & f(\mathbf{x}_i) - r\,|\nabla f(\mathbf{x}_i)| \le f_v \le \\ & f(\mathbf{x}_i) + r\,|\nabla f(\mathbf{x}_i)| \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

where r is the desired thickness in voxels of the transition region, and the gradient vector is approximated using the operator given in the section on shading. A graph of $\alpha(\mathbf{x}_i)$ as a function of $f(\mathbf{x}_i)$ and $|\nabla f(\mathbf{x}_i)|$ for typical values of f_v, α_v, and r is shown in Figure 3.

If more than one isovalue surface is to be displayed in a single image, they can be classified separately and their opacities combined. Specifically, given selected values $f_{v_n}, n = 1,\ldots,N, N \ge 1$, opacities α_{v_n}, and transition region thicknesses r_n, we can use Equation 3 to compute $\alpha_n(\mathbf{x}_i)$, then apply the relation

$$\alpha_{tot}(\mathbf{x}_i) = 1 - \prod_{n=1}^{N} (1 - \alpha_n(\mathbf{x}_i)).$$

Region boundary surfaces

From a densitometric point of view, the human body is a complex arrangement of biological tissues, each of which is fairly homogeneous and of predictable density. Clinicians are interested mostly in the boundaries between tissues, from which the sizes and spatial relationships of anatomical features can be inferred.

Although many researchers use isovalue surfaces for the display of medical data, it is not clear that they are well suited for that purpose. The cause can be explained briefly as follows: Given an anatomical scene containing two tissue types A and B with values f_{v_A} and f_{v_B} where $f_{v_A} < f_{v_B}$, data acquisition will produce voxels having values $f(\mathbf{x}_i)$ such that $f_{v_A} \le f(\mathbf{x}_i) \le f_{v_B}$. Thin features of tissue type B may be represented by regions in which all voxels bear values less than f_{v_B}. Indeed, there is no threshold value greater than f_{v_A} guaranteed to detect arbitrarily thin regions of type B, and thresholds close to f_{v_A} are as likely to detect noise as signal.

The procedure employed in this study is based on the following simplified model of anatomical scenes and the CT scanning process. We assume that scenes contain an arbitrary number of tissue types bearing CT numbers falling within a small neighborhood of some known value. We further assume that tissues of each type touch tissues of at most two other types in a given scene. Finally, we assume that, if we order the types by CT number, then each type touches only types adjacent to it in the ordering. Formally, given N tissue types bearing CT numbers $f_{v_n}, n = 1,\ldots,N, N \ge 1$ such that $f_{v_m} < f_{v_{m+1}}, m = 1,\ldots,N-1$, then no tissue of CT number $f_{v_{n_1}}$ touches any tissue of CT number $f_{v_{n_2}}, |n_1 - n_2| > 1$.

If these criteria are met, each tissue type can be assigned an opacity, and a piecewise linear mapping can be constructed that converts voxel value f_{v_n} to opacity α_{v_n}, voxel value $f_{v_{n+1}}$ to opacity $\alpha_{v_{n+1}}$, and intermediate voxel values to intermediate opacities. Note that all voxels are typically mapped to some nonzero opacity and will thus contribute to the final image. This scheme ensures that thin regions of tissue will still appear in the image, even if only as faint wisps. Note also that violation of the adjacency criteria leads to voxels that cannot be unambiguously classified as belonging to one region boundary or another and hence cannot be rendered correctly using this method.

The superimposition of multiple semitransparent surfaces such as skin and bone can instantially enhance the comprehension of CT data. To get such effects using volume rendering, we would like to suppress the opacity of tissue interiors while enhancing the opacity of their

bounding surfaces. We implement this by scaling the opacities computed above by the magnitude of the local gradient vector.

Combining these two operations, we obtain a set of expressions

$$\alpha(\mathbf{x}_i) = |\nabla f(\mathbf{x}_i)| \begin{cases} \alpha_{v_{n+1}} \left[\dfrac{f(\mathbf{x}_i) - f_{v_n}}{f_{v_{n+1}} - f_{v_n}} \right] + \\ \alpha_{v_n} \left[\dfrac{f_{v_{n+1}} - f(\mathbf{x}_i)}{f_{v_{n+1}} - f_{v_n}} \right] & \text{if } f_{v_n} \le f(\mathbf{x}_i) \le f_{v_{n+1}} \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

for $n = 1,\ldots,N-1, N \ge 1$. The gradient vector is approximated using the operator given in the section on shading. A graph of $\alpha(\mathbf{x}_i)$ as a function of $f(\mathbf{x}_i)$ and $|\nabla f(\mathbf{x}_i)|$ for three tissue types, *A*, *B*, and *C*, having typical values of $f_{v_A}, f_{v_B}, f_{v_C}, \alpha_{v_A}, \alpha_{v_B}$, and α_{v_C} is shown in Figure 4.

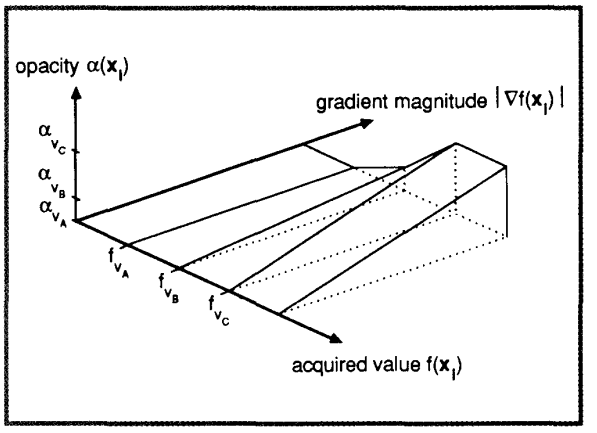


Figure 4. Calculation of opacities for region boundary surfaces.

Discussion

One of the strengths of the rendering method presented in this article is modularity.

Computational complexity

By storing intermediate results at various places in the pipeline, the cost of generating a new image after a change in shading or classification parameters is minimized. Let us consider some typical cases.

Given acquired value $f(\mathbf{x}_i)$ and gradient magnitude $|\nabla f(\mathbf{x}_i)|$, the mapping to opacity $\alpha(\mathbf{x}_i)$ can be implemented with one lookup-table reference. This implies that if we store gradient magnitudes for all voxels, computation of new opacities following a change in classification parameters entails only generation of a new lookup table followed by one table reference per voxel.

The cost of computing new colors $c_\lambda(\mathbf{x}_i)$ following a change in observer direction **V**, light source direction **L**, or other shading parameters is more substantial. Effective rotation sequences can be generated, however, using a single set of colors. The visual manifestation of fixing the shading is that light sources appear to travel around with the data as it rotates, and highlights are incorrect. Since we are visualizing imaginary or invisible phenomena anyway, observers are seldom troubled by this effect.

The most efficient way to produce a rotation sequence is to hold both colors and opacities constant and alter only the direction in which rays are cast. If we assume a square image n pixels wide and use orthographic projection, in which case sample coordinates can be efficiently calculated using differencing, the combined cost of ray tracing, resampling, and compositing to compute n^2 pixels is $3Kn^2 2Kn^2$ trilinear interpolations, and Kn^2 linear interpolations, where K is the number of sample locations along each ray.

Image quality

Although the notation used in Equation 1 has been borrowed from the literature of image compositing,[16] the analogy is not exact, and the differences are fundamental. Volume data consists of samples taken from a band-limited 3D scene, whereas the data acquired from an image digitizer consists of samples taken from a band-limited 2D projection of a 3D scene. Unless we reconstruct the 3D scene that gave rise to our volume data, we cannot compute an accurate projection of it. Volume rendering performs no such reconstruction. Image quality is therefore limited by the number of viewing rays. In the current implementation, we cast one ray per pixel. Such point sampling would normally produce strong aliasing, but by using nonbinary classification decisions, we carry much of the band-limiting inherent in the acquired data over into the image, substantially reducing aliasing artifacts. Stated another way, we are depending on 3D band-limiting to avoid aliasing in 2D projections.

Within these limitations, there are two ways to improve image quality, blurring, and supersampling. If the array of acquired values is blurred slightly during data preparation, the oversharp surface silhouettes occasionally exhibited by volume renderings are softened. Alternatively, we can apply blurring to the opacities generated by the classification procedure, but leave the shading untouched. This has the effect of softening silhouettes without adversely affecting the crispness of surface detail.

The decision to reduce aliasing at the expense of resolution arises from two conflicting goals: generating

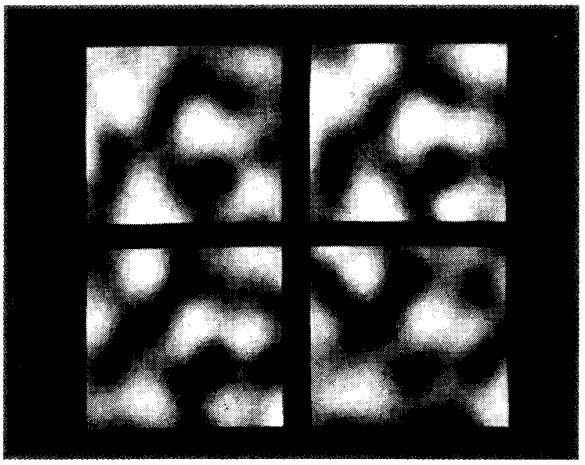


Figure 5. Slices from 113 × 113 × 113 voxel electron density map.

Figure 6. Volume rendering of isovalue contour surface from dataset shown in Figure 5.

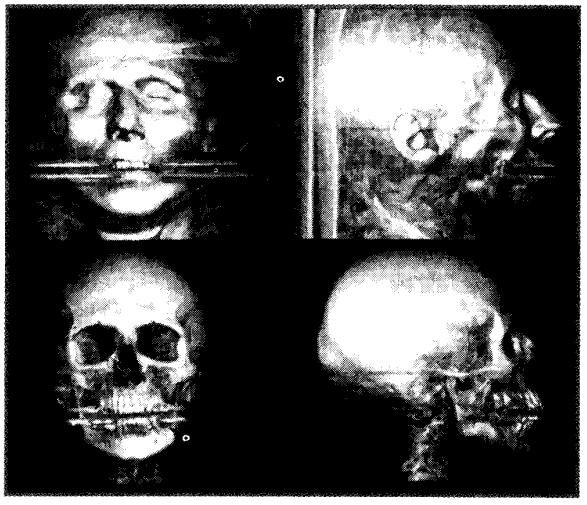


Figure 7. Volume rendering of region boundary surfaces from 256 × 256 × 113 voxel CT dataset.

artifact-free images and keeping rendering costs low. In practice, the slight loss in image sharpness might not be disadvantageous. Indeed, it is not clear that the accuracy afforded by more expensive visibility calculations is useful, at least for the types of data considered in this study. Blurry silhouettes have less visual impact, but they reflect the true imprecision in our knowledge of surface locations.

An alternative means for improving image quality is supersampling. The basic idea is to interpolate additional samples between the acquired ones prior to compositing. If the interpolation method is a good one, the accuracy of the visibility calculations is improved, reducing some kinds of aliasing. Another option is to apply this interpolation during data preparation. Although this alternative substantially increases computational expense throughout the remainder of the pipeline, it improves the accuracy of our shading and classification calculations as well as our visibility.

Implementation and results

The dataset used in the molecular graphics study is a 113 × 113 × 113 voxel portion of an electron density map for the protein Cytochrome B5. Figure 5 shows four slices spaced 10 voxels apart in this dataset. Each whitish cloud represents a single atom. Using the shading and classification calculations described in the sections on shading and the subsection on isovalue contour surfaces, colors and opacities were computed for each voxel in the expanded dataset. These calculations required five minutes on a Sun 4/280 having 32M bytes of main memory. Ray tracing and compositing were performed as described in the section on the rendering pipeline and took 40 minutes, yielding the image in Figure 6.

The dataset used in the medical imaging study is a CT study of a cadaver and was acquired as 113 slices of 256 × 256 samples each. Using the shading and classification calculations described in the sections on shading region boundary surfaces, two sets of colors and opacities were computed, one showing the air-skin interface and a second showing the tissue-bone interface. The computation of each set required 20 minutes. Using the compositing algorithm described in the section on the rendering pipeline, two views were then computed from each set of colors and opacities, producing four images in all, as shown in Figure 7. The computation of each view required an additional 20 minutes.

The horizontal bands through the cadaver's teeth in all of these images are artifacts due to scattering of X rays from dental fillings and are present in the acquired data. The bands across her forehead and under her chin in the air-skin images are gauze bandages used to immobilize her head during scanning. Her skin and nose cartilage are rendered semitransparently over the bone surface in the tissue-bone images.

Figure 8 was generated by combining halves from each

of the two sets of colors and opacities already computed for Figure 7. Heightened transparency of the temporal bone and the bones surrounding the maxillary sinuses—more evident in moving sequences than in a static view—is due to generalized osteoporosis. It is worth noting that rendering techniques employing binary classification decisions would likely display holes here instead of thin, wispy surfaces.

The dataset used in Figures 9 and 10 is of the same cadaver, but was acquired as 113 slices of 512 × 512 samples each. Figure 9 was generated using the same procedure as for Figure 7, but casting four rays per slice in the vertical direction to correct for the aspect ratio of the dataset. Figure 10 was generated by expanding the dataset to 452 slices using a cubic B-spline in the vertical direction, then generating an image from the larger dataset by casting one ray per slice. As expected, more detail is apparent in Figure 10 than Figure 9.

Figure 8. Rotated view of same dataset as in Figure 7.

Conclusions

Volume rendering has proved to be an effective modality for the display of surfaces from sampled scalar functions of three spatial dimensions. As demonstrated by the figures, it can generate images exhibiting approximately equivalent resolution, yet containing fewer interpretation errors than techniques relying on geometric primitives.

Despite its advantages, volume rendering has several problems: The omission of an intermediate geometric representation makes selection of appropriate shading parameters critical to the effectiveness of the visualization. Slight changes in opacity ramps or interpolation methods radically alter the features that are seen as well as the overall quality of the image. For example, the thickness of the transition region surrounding the isovalue contour surfaces described in the section on isovalue contour surfaces stays constant only if the local gradient magnitude stays constant within a radius of r voxels around each point on the surface. The time and ensemble averaging inherent in X-ray crystallography usually yields suitable data, but there are considerable variations among datasets. Algorithms are needed that automatically select an optimum value for r based on the characteristics of a particular dataset.

Volume rendering is also very sensitive to artifacts in the acquisition process. For example, CT scanners generally have anisotropic spatial sensitivity. This problem manifests itself as striping in images. With live subjects, patient motion is also a serious problem. Since shading calculations are strongly dependent on the orientation of the local gradient, slight misalignments between adjacent slices produce strong striping.

An issue related specifically to region boundary surfaces is the rendering of datasets not meeting the adjacency criteria described in the section on region boundary surfaces. This includes most internal soft tissue organs in CT data. One simple solution would be for users to clip or carve away unwanted tissues interactively, thus isolating subsets of the acquired data that meet the adjacency criteria. Since the user or algorithm is not called on to define geometry, but merely to iso-

Figure 9. Volume rendering of 512 × 512 × 113 voxel CT dataset.

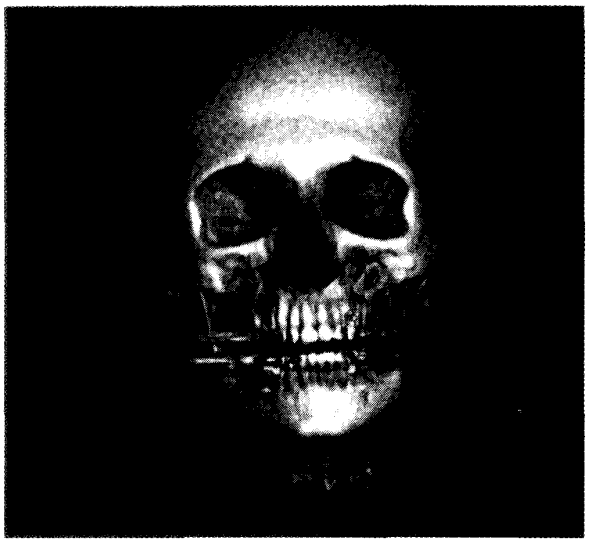


Figure 10. Supersampled volume rendering of same dataset as in Figure 9.

late regions of interest, this approach promises to be easier and to produce better images than techniques involving surface fitting. A more sophisticated approach would be to combine volume rendering with high-level object definition methods in an interactive setting. Initial visualizations, made without the benefit of object definition, would be used to guide scene analysis and segmentation algorithms, which would in turn be used to isolate regions of interest, producing a better visualization. If the output of such segmentation algorithms included confidence levels or probabilities, they could be mapped to opacity and thus modulate the appearance of the image.

Although this article focuses on display of surfaces, the same pipeline can easily be modified to render interstitial volumes as semitransparent gels. Color and texture can be added to represent such variables as gradient magnitude or anatomical tissue type. Visualizations combining sampled and geometric data also hold much promise. For example, it might be useful to superimpose ball-and-stick molecular models onto electron density maps or medical prostheses onto CT scans. To obtain correct visibility, a true 3D merge of the sampled and geometric data must be performed. One possible solution is to scan-convert the geometry directly into the sampled array and render the ensemble. Alternatively, classical ray tracing of the geometry can be incorporated directly into the volume-rendering pipeline. Another useful tool would be the ability to perform a true 3D merge of two or more visualizations, allowing, for example, the superimposition of radiation treatment planning isodose surfaces over CT data.

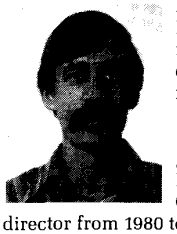
The prospects for rapid interactive manipulation of volume data are encouraging. By precomputing colors and opacities and storing them in intermediate 3D datasets, we simplify the image generation problem to one of geometrically transforming two values per voxel and compositing the results. One promising technique for speeding up these transformations is to combine a three-pass version of two-pass texture mapping[17] with compositing. By resampling separately in each of three orthogonal directions, computational expense is reduced. Given a suitably interpolated sample array, it might be possible to omit the third pass entirely, compositing transformed 2D slices together to form an image. This further suggests that hardware implementations might be feasible. A recent survey of architectures for rendering voxel data is given by Kaufman.[18] A suitably interconnected 2D array of processors with sufficient backing storage might be capable of producing visualizations of volume datasets in real-time or near real-time. ■

Acknowledgments

I wish to thank Frederick P. Brooks, Jr., Henry Fuchs, Steven M. Pizer, and Turner Whitted for their encouragement on this article, and John M. Gauch, Andrew S. Glassner, Mark Harris, Lawrence M. Lifshitz, Andrew Skinner, and Lee Westover for enlightening discussions and advice. The comments of Nelson Max on an early version were also helpful. The electron density map used in this study came from Jane and Dave Richardson of Duke University, and the CT scan was provided by the Radiation Oncology Department at North Carolina Memorial Hospital. This work was supported by NIH grant RR02170 and ONR grant N00014-86-K-0680.

References

1. G.T. Herman and H.K. Liu, "Three-Dimensional Display of Human Organs from Computer Tomograms," *Computer Graphics and Image Processing*, Jan. 1979, pp. 1-21.
2. K.H. Hoehne and R. Bernstein, "Shading 3D-Images from CT Using Gray-Level Gradients," *IEEE Trans. Medical Imaging*, Mar. 1986, pp. 45-47.
3. D.S. Schlusselberg and W.K. Smith, "Three-Dimensional Display of Medical Image Volumes," *NCGA 86 Conf. Proc.*, NCGA, Fairfax, Va., pp. 114-123.
4. S. Goldwasser, "Rapid Techniques for the Display and Manipulation of 3D Biomedical Data," NCGA 86 Conf. tutorial, NCGA, Fairfax, Va.
5. Y. Trousset and F. Schmitt, "Active-Ray Tracing for 3D Medical Imaging," *Eurographics 87 Conf. Proc.*, Amsterdam, pp. 139-149.
6. S.M. Pizer et al., "3D Shaded Graphics in Radiotherapy and Diagnostic Imaging," *NCGA 86 Conf. Proc.*, NCGA, Fairfax, Va., 1986, pp. 107-113.
7. W.E. Lorensen and H.E. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm, *Computer Graphics* (Proc. SIGGRAPH), July 1987, pp. 163-169.
8. T.R. Williams, *A Man-Machine Interface for Interpreting Electron Density Maps*, doctoral dissertation, Univ. of North Carolina, Chapel Hill, N.C., 1982.
9. G.D. Purvis and C. Culberson, "On the Graphical Display of Molecular Electrostatic Force-Fields and Gradients of the Electron Density," *J. Molecular Graphics*, June 1986, pp. 89-92.
10. A.R. Smith, "Volume Graphics and Volume Visualization: A Tutorial," Tech. Memo 176, Pixar Inc., San Rafael, Calif., 1987.
11. R.A. Drebin, verbal presentation in *Computer Graphics and the Sciences*, tutorial at *SIGGRAPH 87*.
12. M. Levoy and T. Whitted, "The Use of Points as a Display Primitive," Tech. Report 85-022, Computer Science Dept., Univ. of North Carolina, Chapel Hill, 1985.
13. M. Levoy, "Rendering of Surfaces from Volumetric Data," Tech. Report 87-016, Computer Science Dept., Univ. of North Carolina, Chapel Hill, 1987.
14. M. Levoy, "Direct Visualization of Surfaces from Computed Tomography Data," *SPIE Medical Imaging II Conf. Proc.*, Newport Beach, Calif., Feb. 1988 (to appear).
15. P. Bui-Tuong, "Illumination for Computer-Generated Pictures," *CACM*, June 1975, pp. 311-317.
16. T. Porter and T. Duff, "Compositing Digital Images," *Computer Graphics* (Proc. SIGGRAPH), July 1984, pp. 253-259.
17. E. Catmull and A.R. Smith, "3D Transformations of Images in Scanline Order," *Computer Graphics*, (Proc. SIGGRAPH), July 1980, pp. 279-285.
18. A. Kaufman, "Voxel-Based Architectures for Three-Dimensional Graphics," *Proc. IFIP 10th World Computer Congress*, Amsterdam, Sept. 1986, pp. 361-366.



Marc Levoy is a PhD student at the University of North Carolina, Chapel Hill. His research interests include texture mapping, display of complex surfaces, volume rendering, medical imaging, and molecular graphics.

Levoy received a bachelor of architecture in 1976 and a master of science in computer graphics in 1978, both from Cornell University. He is principal developer of the Hanna-Barbera Computer Animation System and served as its director from 1980 to 1982. He is a member of ACM and IEEE.

Levoy can be contacted at the Department of Computer Science, Sitterson Hall, University of North Carolina, Chapel Hill, NC 27599-3175.