Medical Image Analysis (1999) volume ?, number ?, pp 1–34

Fast Re-Rendering Of Volume and Surface Graphics By Depth, Color, and Opacity Buffering

Abhir Bhalerao[1*], Hanspeter Pfister[2], Michael Halle[3] and Ron Kikinis[4]

[1]University of Warwick, England. Email: abhir@dcs.warwick.ac.uk
[2]Mitsubishi Electric Research Laboratories, Cambridge, MA. E-mail: pfister@merl.com
[3]Media Laboratory, MIT, Cambridge. E-mail: halazar@bwh.harvard.edu
[4]Surgical Planning Laboratory, Brigham and Women's Hospital and Harvard Medical School, Boston. E-mail: kikinis@bwh.harvard.edu

Abstract
A method for quickly re-rendering volume data consisting of several distinct materials and intermixed with moving geometry is presented. The technique works by storing depth, color and opacity information, to a given approximation, which facilitates accelerated rendering of fixed views at moderate storage overhead without re-scanning the entire volume. Storage information in the ray direction (what we have called *super*-z depth buffering), allows rapid transparency and color changes of materials, position changes of sub-objects, dealing explicitly with regions of overlap, and the intermixing or separately rendered geometry. The rendering quality can be traded-off against the relative storage cost and we present an empirical analysis of output error together with typical figures for its storage complexity. The method has been applied to visualization of medical image data for surgical planning and guidance, and presented results include typical clinical data. We discuss the implications of our method for haptic (or tactile) rendering systems, such as for surgical simulation, and present preliminary results of rendering polygonal objects in the volume rendered scene.

Keywords: Volume Rendering, Classification in Volume Rendering, Combining Volume and Surface graphics, Haptic Rendering

Received December 1998; revised March 1999; accepted June 1999

1. Introduction

Visualization plays a central role in the presentation of 3D medical image data, such as Magnetic Resonance (MR), Computerized Tomography (CT), MR Angiography (MRA), or a combination of these (Wells *et al.*, 1996b), to the radiologist or surgeon for diagnosis and surgery planning. However, the typically large data sizes involved (anywhere between 10 and 70 million voxels for a single examination) is stretching the limits of the currently available rendering hardware. Also, with the growing use of pre-operative and intra-operative images during the surgical intervention itself, where the rate of update can be more critical, the problem of efficient rendering of volume and surface graphics is a still a pertinent topic for study.

In this paper, we describe a volume compositing scheme that uses a specialized depth buffer which facilitates the rapid re-rendering of fixed viewpoints without the traversal of the entire volume. For volumes characterized by disjoint materials, which is common for medical image data that is classified into separate anatomical parts, it is possible to quickly re-render the data after transparency and color changes of any material. By separately rendering each material, the intermediate structure can also support shape and position changes of the sub-objects without rendering the entire scene. Additionally, we demonstrate how moving surface geometry can be intermixed into the volume rendering.

Both volume and surface rendering methods are used for surgical planning and guidance tasks. Each has its

*Corresponding author, Department of Computer Science, University of Warwick, England.
(e-mail: abhir@dcs.warwick.ac.uk)

advantages and drawbacks, some of which are well known. Volume rendering is mainly favored because of its ability to give a qualitative feel for the density changes in the data and precludes the need for classifying or segmenting the data (Levoy, 1988) - indeed it is particularly adept at showing structurally weak and "fuzzy" information. One of the main sources of error in volume rendering is partial voluming (Jacq and Roux, 1997), when the voxels ostensibly represent multiple materials, or tissues types in the case of medical data. Another disadvantage is that currently special purpose hardware (Pfister and Kaufman, 1996) (Pfister *et al.*, 1999) or shared memory processor architectures (Lacroute, 1996) are required to achieve visualization speeds approaching real time (e.g. 30 frames a second).

Surface rendering has the advantage that accelerated hardware is commonly available, but the disadvantage that iso-surfaces have to be defined in the volume data in order to construct the triangle meshes (Lorensen and Cline, 1987). This may be satisfactory when the data to be visualized has already been segmented, e.g. using accurate statistical methods (Wells *et al.*, 1996a) (Warfield *et al.*, 1995), since the model surfaces have been defined. The time penalty for segmentation, whether manual or semi-automatic, and model generation is tolerated to gain real time performance during visualization.

A rendering pipeline that can incorporate the efficiency of polygonally defined objects into the realism of a volumetric scene is desirable, especially in medical applications (Kaufman *et al.*, 1990) (Ebert and Parent, 1990) such as virtual endoscopy surgery simulation (Geiger and Kikinis, 1995). In such a system, sampled volume data, such as CT or MR images can be directly combined with synthetic objects such as surgical instruments, probes, catheters, prostheses and landmarks displayed as glyphs. In some instances, preoperatively derived surface models for certain anatomical structures such as skin are can be more efficiently stored and better visualized as a polygon mesh. A straightforward way of mixing volume and polygonal graphics is to convert the polygonal models into sampled volumes and then render them using a volume rendering pipeline e.g. (Kaufman *et al.*, 1990). Another way is to simultaneously cast rays through both the polygonal and volume data, at the same sample intervals, and then composite the colors and opacities in depth sort order (Levoy, 1990). In the former, the all-volume approach simplifies the rendering but is expensive in speed and and storage. Both methods require compensation for aliasing if a realistic ray sample rate is to be used.

Two scenarios that require careful consideration of the trade-offs between the type, quality and speed of rendering used are: visualization during surgical intervention, and haptic rendering systems such as proposed for surgical simulation (Massie and Sailsbury, 1994) (Avila and Sobierajski, 1996). For example, using pre-operative data for intra-operative guidance in an Interventional MR (iMR) scanner, which is able to provide regular intra-operative updates to the imaged scene. It has already been demonstrated that it is possible to perform pre-operative segmentation by elastic matching of surfaces e.g. (Warfield *et al.*, 1995). However, for intra-operative segmentation and virtual surgery simulation, physical tissue modeling using voxels is being advocated (Gibson, 1997) - hence the need for volume graphics. Such systems raise the question of speed of update, i.e. being able to quickly reflect changes in the voxel volume, whether these are intermediate results from an elastic matching scheme or more simply the result of "volume sculpting". An important characteristic of both these situations, which we have sought to exploit in this work, is the observer's viewpoint being fixed, or not changed during interaction.

2. Super-Z Depth Buffering

The basic approach is to render multiple volumes separately, keeping the depth information for each volumetric object, and then compositing the projected data with information about object color and transparency. We have called this method *super*-z buffer rendering, as each intermediate image is a 2D pixel plane with 3D z-buffer information that controls the subsequent compositing process. One way to regard this approach is that each object rendering produces a separate view plane, with its own transparency value. Comparisons to "sprite" based graphics used in computer animation can be drawn. However, as described below, the depth information at each pixel is richer than a simple Z and alpha blending coefficient.

The idea of compositing individually rendered volumes is not new. Machiraju and Yagel (Machiraju and Yagel, 1993) employ a similar rendering method to parallelize the volume rendering pipeline, however they do not attempt to store intermediate images for re-compositing. More recently, Gortler et. al (Gortler *et al.*, 1997) suggest a Layered Depth Image (LDI), an almost identical structure to our super-z lists, as the basis of their image based rendering. They are principally interested in accelerating the image warping process to quickly generating multiple views. Also, our colleagues (Umans *et al.*, 1997), are using a Multi-Layer Image (MLI) for reducing the data bandwidth between off-line renderings of fixed views of a networked browsing application implemented in Java. In this work, the MLI allows pseudo-3D capabilities on a non-graphics workstation. What we report here generalizes the MLI and LDI compositing schemes to include volume rendered objects and separately rendered polygonal objects.

Brady et. al (Brady *et al.*, 1995) describe a virtual navigation system which uses a compositing structure they term Approximate Discrete Radon Transform (ARDT). This is a local, hierarchical based structure that is a a super-set of our global super-z lists and can be used to accelerate local view point transformations (translations, and rotations). As the user navigates, parts of the ARDT are updated, and the virtual endoscopic view can be rendered at interactive speeds.

Sobierajski and Kaufman (Sobierajski and Kaufman, 1994) described a much more generalized pipeline for accelerating volumetric ray tracing with global illumination. Different classes of volumetric objects can be defined (e.g. geometric, iso-surface, maximum intensity projection) and ray-intersection calculation and shading is performed separately. The intersection information is passed to the shader in a similar manner to that described here and we borrow their "segment" terminology to describe groups of contiguous voxels. Their method is general enough to produce illumination effects such as shadows, which can give important depth cue information - this is one limitation of our technique. However, they do not describe an intermediate storage methodology and solely use a volume rendering framework.

Fig. 19 shows an overview of super-z based rendering. Each volumetric object is separately rendered parameterized by the global camera and lighting model (the material properties controlling the shading of each volume can be defined differently if required). This does not necessarily imply multiple traversals of the volume. A voxel representation that includes the material labeling and/or tissue occupancy values allows the intermediate depth structures (see below) to be built in a single pass. During the rendering, a specialized depth buffer is formed which describes, in list form, the *segments* encountered by each cast ray. The final stage is to composite the multiple image and depth-buffer combinations controlled by color and transparency information.

Assume that the voxel represents multiple materials, such as in an MR or CT scan. Following the ray r for a picture plane pixel through the image volume as it enters and leaves an image voxel i, the accumulated opacity and color can be expressed as recursive formulae (Wilhelms and van Gelder, 1991):

$$A_{out}[r,i] = (1-A_{in}[r,i])\alpha[r,i] + A_{in}[r,i], \quad (1)$$

$$C_{\lambda out}[r,i] = (1-A_{in}[r,i])c_{\lambda}[r,i] + C_{\lambda in}[r,i]. \quad (2)$$

where,

$$\begin{aligned} \lambda &= \text{either } r, g, \text{ or } b, \\ i &= \text{Sample location along ray } r, \\ r &= \text{Ray through picture image pixel,} \\ \alpha[r,i] &= \text{Opacity at sample } i, \end{aligned} \quad (3)$$

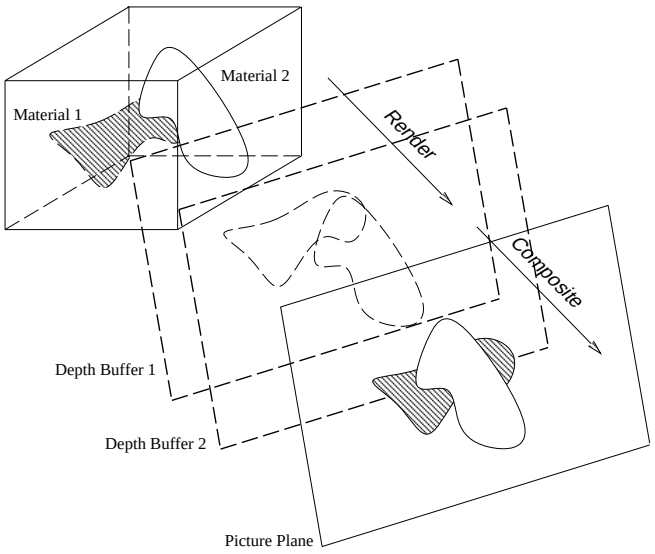


Figure 1. Overview of depth buffer volume rendering

$$\begin{aligned} c_{\lambda}[r,i] &= \text{Color at sample } i \\ A[r,i] &= \text{Accumulated opacity for ray } r \text{ upto sample } i, \\ C_{\lambda}[r,i] &= \text{Accumulated color for ray } r \text{ upto sample } i. \end{aligned} \quad (4)$$

The pair of equations represent a front-to-back (FTB) compositing through the voxels. The amount of light entering the voxel $(1-A_{in})$ is modulated by the opacity of the voxel itself in Equation 1 while Equation 2 separately sums the color contribution of the voxel. For the separate rendering/compositing to work it is necessary to store opacities and color information along each ray, for all materials. For each ray with n samples, there will be M sets of information consisting of opacity-color pairs at each ray sample point r:

$$\bigcup_{m}^{M} R[r,i] = \bigcup_{m}^{M} \{(\alpha_m[r,0], c_{\lambda m}[r,0]), (\alpha_m[r,1], c_{\lambda m}[r,1]), \ldots, (\alpha_m[r,n], c_{\lambda m}[r,n])\}. \quad (5)$$

If we make the assumption that materials will be exclusive, for the most part, the compositing along each ray will be restricted to a single material type, other than in the *overlap* or partial volume regions.

Two conditions are tested to reduce the ray set size:

1. Exclude opacity color pairs for voxels where the opacity is below some minimum threshold $\alpha[r,i] < \alpha_{min}$ (very transparent voxels) e.g. $\alpha_{min} = 0.005$.
2. Contiguous sets of voxels (*segments*) are grouped together if the *differential* opacity between pairs of voxels is less than an opacity-delta value, $\delta = \alpha[i+1] - \alpha[i]$.

Condition 1 is the scheme adopted by the Lacroute shear/warp factorization to reduce the storage overhead for the classified volume by run-length encoding the data (RLE) and speed up the subsequent compositing for each scanline - basically the renderer only does work when there is a non-transparent voxel run (Lacroute and Levoy, 1994). Condition 2 is an approximation control factor which deals with non-uniformity in the opacity over a voxel run.

The ray sets are considerably reduced by only keeping opacity-color pairs for contiguous, non-transparent voxels, where the opacity 'gradient' is below δ. Note that at $\delta = 1.0$, the poorest approximation is achieved and assumes uniform opacity over a voxel segment. With $\delta = 0$, all opacity variation is captured by the segment ray set and is in fact equivalent to a full voxel-by-voxel ray cast. The use of the linear differential opacity $\delta = \alpha[k+1] - \alpha[k]$ to control the opacity approximation is simple but crude. Examining the underlying density emitter model, opacity through one voxel traversal depends on the optical depth per unit length ρ; opacity of an unit length voxel may be expressed as $\alpha[k] = 1 - \exp(-\rho_k)$ (Blinn, 1982). In order to account for this intrinsic non-linearity, a better choice of threshold parameter is given by the magnitude of the quantity:

$$\delta = \rho[k+1] - \rho[k] = \log\left[\frac{1-\alpha[k+1]}{1-\alpha[k]}\right] \tag{6}$$

The simple linear form is used in the experiments presented below.

In this way we minimize the quantity of stored depth information, by allowing the FTB compositing to take place independently for each material, approximating the opacity variation to construct a ray set R_m which contains only the *start* and *end* opacity-color pairs.

$$R_m[r] = \{S_m[r,1], S_m[r,2], \ldots, S_{r,m}[N]\}, \tag{7}$$

consists of sets S_m of depth sorted and *accumulated* opacity-color pairs, $(A_m[r], C_m[r])$, one for each the start and end of a set of contiguous, non-transparent voxels. Thus

$$\begin{aligned} S_m[r,k] &= \{(A_m[r,s[k]], C_{\lambda m}[r,s[k]]), \\ &\quad (A_m[r,e[k]], C_{\lambda m}[r,e[k]])\}, e[k] \geq s[k]. \end{aligned} \tag{8}$$

The set subscript k indexes the start $s[k]$ and end $e[k]$ sample positions along the ray. The set S_m then contains the accumulated opacity-color pairs for the front-to-back compositing for material m at picture pixel r. Although only the end value is required for the compositing, the start values play a role when explicitly handling overlapping regions (Section 2.1).

The segment opacity and color values, $(\alpha[r,k], c_\lambda[r,k])$, can be obtained from the accumulated opacity and color values, $(A_m[r,i], C_m[r,i])$, by resetting the accumulation variables to zero at the start of a new segment (determined by conditions 1 and 2 above) as the rendering proceeds.

Fig. 20 illustrates an example of a ray cast front-to-back through two uniform materials, $m = 1$ and $m = 2$, showing the voxel segments encountered. The horizontal axis represents the depth i.e. z. This illustrates the ray set of Eqn. 7 for $m = 1$ and $m = 2$ showing the extent of the volume segments represented by the opacity values $\alpha[1], \alpha[2], \alpha[3]$.

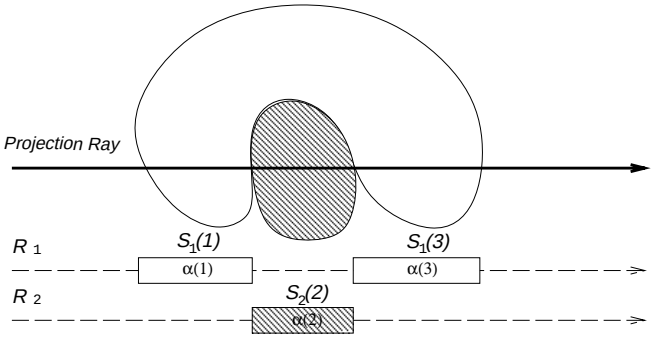


Figure 2. The same ray cast through two materials showing image segments encountered

To determine the output color for segment ray-sets therefore, which are an encoding of the voxels encountered during the rendering, we merge all the segment sets, sorting by depth order, and compositing in the same manner as Eqns. 1 and 2:

$$\begin{aligned} A[r,k+1] &= (1 - A[r,k])\alpha[r,k] + A[r,k], \\ C_\lambda[r,k+1] &= (1 - A[r,k])c_\lambda[r,k] + C_\lambda[r,k], \end{aligned} \tag{9}$$

where

$$\begin{aligned} A[0] &= 0, \\ C_\lambda[0] &= 0. \end{aligned} \tag{10}$$

This accumulation is performed over the total number of merged segments, after merge sorting the $S_m[r,k]$ ray sets, rather than a sampling of the ray at n points. The start z values are used for the merge sorting of segments along the ray, and the end opacity values are used to perform the compositing.

2.1. Overlapping segments

In our implementation, we ignore the problem of overlapping segments by merge sorting the super-z lists only on the front z value $s[k]$. So, for example, the rays in the top of Fig. 21 will be merge sorted to the depth line shown at the bottom. Compositing would take place in the order $S_1[1], S_2[2], S_1[3]$

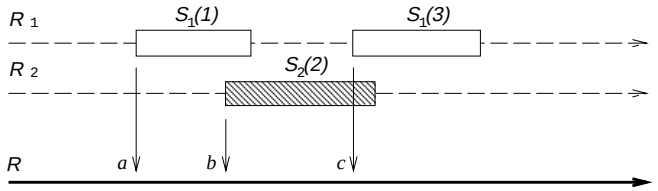


Figure 3. Merge sorting super-z lists on the front *z* value.

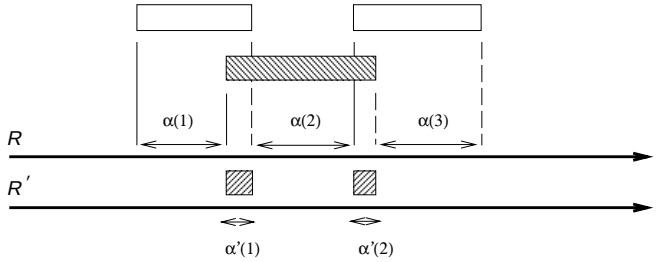


Figure 4. Overlapping sub-segments are handled by separate blending strategies.

for the given segments. Depth values a, c, b become sample points for the segment opacities and colors. This is clearly an approximation because if the individual volumes contain long runs of semi-transparent data, and two volumes have a large region of overlap, then the resulting image will be grossly wrong. However, the surgical type volumes presented here do not show visible artifacts, although for $\delta > 0$ some rendered data will be misrepresented. If necessary, a more refined blending strategy, such as that suggested by Jacq and Roux (Jacq and Roux, 1996), could be developed. Although we have not implemented this strategy, one obvious application is the study of registration error in multi-modality fusion.

2.2. Global Sub-volume transparencies

Global transparencies for the materials can be set to make them visible or not without re-rendering the material by adding an intensity multiplier term $t[m]$ to the material opacity $\alpha_m[r,k]$ in equation 9:

$$\alpha_m[r,k] \rightarrow t[m]\alpha_m[r,k]. \tag{11}$$

For a given viewpoint, it is possible to control multiple transparencies at the final compositing stage. Thus, once the z-lists have been created, it is trivial to adjust the final compositing scheme to suit the rendering required.

For $0 < t[m] < 1.0$ this final compositing will be an approximation since the segment opacities may represent multiple voxels. In the normal case, globally changing the opacity of a material would have a transmittance effect on the opacities of successive voxels along the ray. Our compositing approximates this process (depending on the quality set by δ), regarding all segments as surfaces (see Section 4.1 for an empirical analysis). This approximation problem does not arise for applications such as the Java based anatomy browser described by Umans (Umans *et al.*, 1997), where only pre-rendered surface models are interactively blended. Nevertheless, the result of modifying transparency are convincing and the super-z structure does enable interaction and necessary depth information in which to composite polygons.

3. Combining Volume Data and Surface Rendered Geometry

3.1. Merging Geometry with Super-z

In the method outlined below, polygonal data is rasterized separately by accelerated graphics and color, opacity and depth information is merged using the super-z depth information. Our method bears some similarity to that sketched by Lacroute (Lacroute, 1995). Fig. 23 illustrates the general scheme. The algorithm details are defined in Fig. 24.

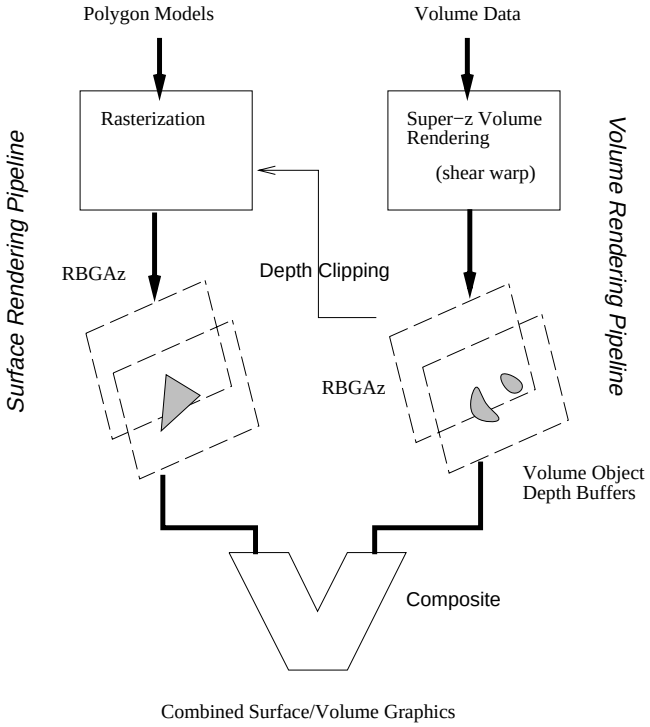


Figure 5. Mixing surface geometry and volume rendered graphics using super-z depth buffering. Polygon rasterization is controlled by the depth buffering. Polygon pixels and volume segment pixels are composited in an interleaved fashion by arithmetic and logic in the compositer.

1. Create a set Z of minimum and maximum segment start values taken over all output pixels for the current compositing.

   Z consists of values $Min(s[k])$ and $Max(s[k])$ from the merged ray sets $R[r,k]$, where $0 \leq k < N_{max}$, plus the depth values $(0,\infty)$ for the space up to and beyond the object:

$$Z = \{0, Min(s[0]), Max(s[0]), ..., Min(s[N_{max}-1]), Max(s[N_{max}-1]), \infty\} \tag{12}$$

   where N_{max} is the total number of merged ray sets.

2. Sort Z into ascending order:

$$Sorted(Z) = \{z[0], z[1], ..., z[N_z - 1]\} \tag{13}$$

   where $z[0] = 0$ and $z[N_z - 1] = \infty$.

3. For each pair of planes $z[p], z[p+1] \in Sorted(Z)$, $0 \leq p < N_z - 1$, interleave the geometry-pixel and the segment-pixel compositing as follows. All pixel operations are clipped against the current depth planes $z[p]$ and $z[p+1]$.

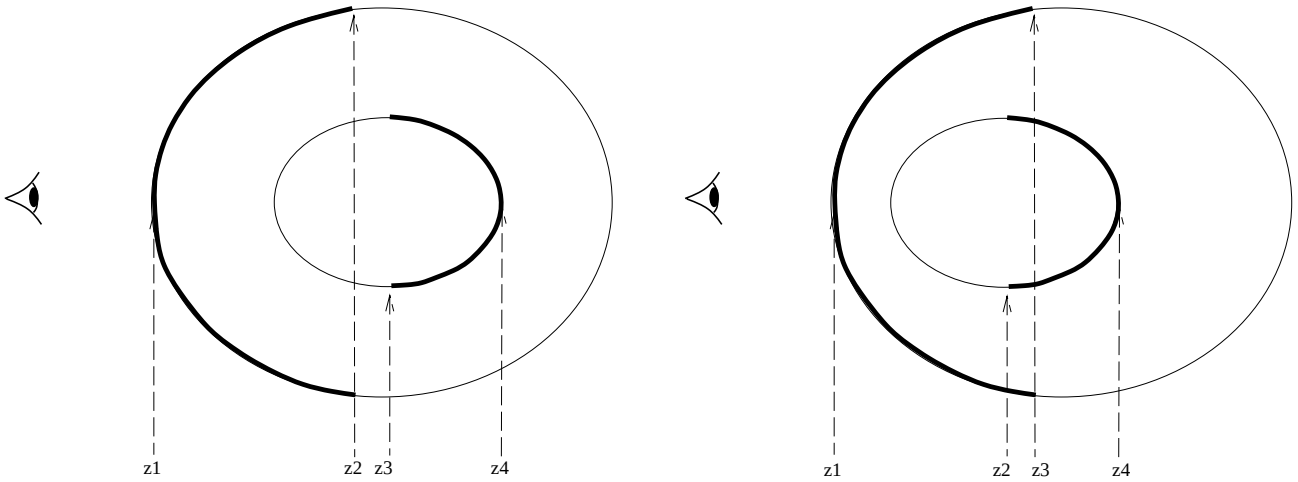
   (a) Send the depth hull: the $s[k]$ values of the segment-pixels, across the entire picture plane.

   (b) FTB composite rasterized geometry that is in front of the depth hull, but behind the current front plane $> z[p+1]$.

   (c) FTB composite segment-pixels

   (d) FTB composite rasterized geometry that is behind the depth hull, but in front of the current back plane $< z[p+1]$.

Figure 6. Algorithm for combining volume and surface graphics using super-z



(a) (b)

Figure 8. Min-max segment values and depth hulls for torus shaped volumes seen side on. In (b) the inner hull overlaps the outer hull and z2 and z3 planes are reversed in comparison to (a).

Stage 1 determines the depth clipping planes which are used to *gate* the sending of geometry through the rasterization. Consecutive segments across the output picture plane are considered together: all first, all second, all third, and so on. Fig. 25 illustrates the positions of the depth planes for a volume which has at most 2 voxel segment along any ray (this torus shaped volume is viewed side on). All the first segments have a range of start depth values between $z1$

and $z2$. Similarly, all the second segments have a range of start depth values between $z3$ and $z4$. Figs. 26(a) and 26(b) illustrate the segment voxel *depth hulls* for two different torus shaped volumes, which are to be merged with any geometry. These appear as backface culled surfaces. Note that this is a limitation for multiple transparent objects as it may be misleading not being able to see both front and back facing polygons of the back-face culling.

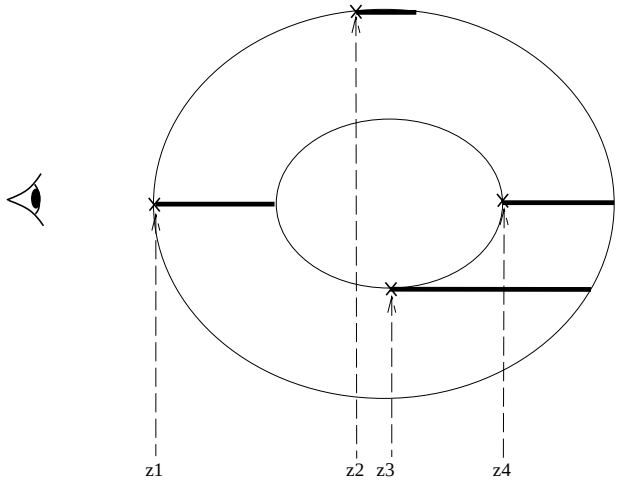


Figure 7. Minimum and maximum segment start depth values for a simple torus shaped volume are at: $z[1,2]$ for the first segment of all rays, and $z[3,4]$ for the second segment. Therefore, $z[1,2,3,4]$ define depth planes used to control the order of polygon rasterization. The pseudo planes $z = 0$ and $z = \infty$ are added to this list.

In figure 27(a), six possible positions of triangles within the volumetric scene are shown. Types 1 and 2 fall outside the depth bounds of the volume and do not require any special treatment. They will be sent first and last, respectively. Types 3 and 4 intersect the depth hull and are handled at stage 3 of the merge. Triangle 6 overlaps two clip planes, and therefore must be rasterized three times because pixels within it lie in front, in between, and behind planes $z2$ and $z3$.

Fig. 27(b) illustrates all the possible ways triangles can be encountered by a pair of planes. Types 3, 4, 5 and 6 will be sent during the segment-geometry compositing between pair of planes. Some geometry will have to be sent more than once, however, only those rasterized pixels which pass the depth test at stage 3(c) and 3(d) need to be composited. In our implementation, the compositing is being done outside the geometry engine. However, we invision that it will be possible to use the primitive operations available in the OpenGL pipeline (Neider *et al.*, 1992) to perform the compositing wholly in the frame graphics (Westermann and Ertl, 1998). Amongst a number of issues being investigated is the performance of the operations involved in the merging process, such as the depth culling of the geometry every time its position and orientation is changed. These problems will definitely arise in cases where the geometry is complex in shape and/or changing at each frame.

4. Results and Discussion

Fig. 15(a)-(k) are volume/surface renderings of interpenetrating, anti-aliased solids within a $256 \times 256 \times 256$ bounding volume: a sphere, a cone and a cylindrical bar. In the left column, all the objects are volumetric. On the right, the sphere is replaced by a polygon model consisting of 1200 triangles. The first row is with all solids opaque. Next, the sphere is made semi-transparent, and then the cone. In the next row, both the cone and sphere are shown semi-transparent. Finally, the sphere is turned off to reveal the intersection of the bar with the cone in the volume rendered only scene, where as it is shown as wireframe in the combined rendering. Note that the combined volume-geometry rendering using super-z works regardless of the type of geometric primitive used, whether it be a surface shaded polygon or a line.

Fig. 15(k) shows the super-z buffer as a color-coded images (receding to the back). Where objects overlap the following color coding is used: red, green, blue, yellow, cyan, magenta, for overlaps of 1, 2, 3, 4, 5 and 6; and white for overlaps of greater than 6. Depth is shaded in the original object color from light (at the front) to dark at the back. Where the cone is occluded by the sphere, the color is bright red. Where the sphere occludes both the cone and vertical bar, the color is bright green.

4.1. Analysis of the Opacity Approximation Error

The degree of approximation used by super-z rendering is controlled by δ. If δ is small, more subtle inter-voxel opacity variation is admitted at a higher storage and computational cost. When $\delta = 0$ all voxels are encoded in the segment list and a full ray casting is achieved. Consequently, the chosen level of approximation will compound light transmittance errors as the global opacity of materials/geometrical objects is changed.

To quantify these effects, we used the synthetic volume data shown in cross-section in Fig. 28(a). This consists of 3 concentric anti-aliased spheres (data size $128 \times 128 \times 128$), with differing gray values. The full quality volume rendering ($\delta = 0$) at a global opacity of $t(m) = 0.5$ is shown in Fig. 28(b). For opacity error measurements with intermixed geometry, we used the cylindrical bar and cone shown in Fig. 28(c). Output opacity values of renderings at different values of $\delta > 0$ and t_{volume} or $t_{geometry}$ were

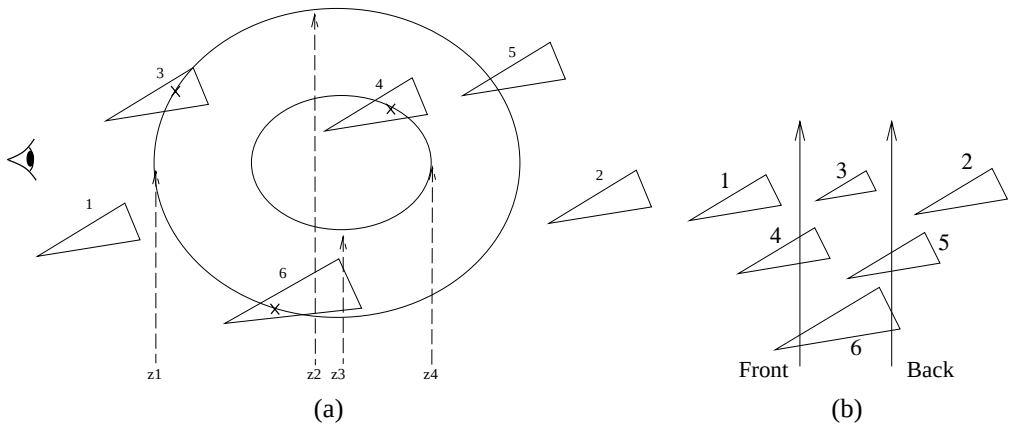


Figure 9. (a) Possible triangle positions rendered within an example volumetric scene. Types 1 and 2 lie in front and behind all voxel data. Triangle types 3 and 4 intersect surfaces of volume data. Type 5 intersects a "backface" and will be treated the same as type 2. Type 6 intersects the surface of a volume and straddles two clipping planes. (b) Triangle types clipped against a pair of depth planes. Types 1 and 2 are not sent. Types 3, 4, 5 and 6 are rasterized.

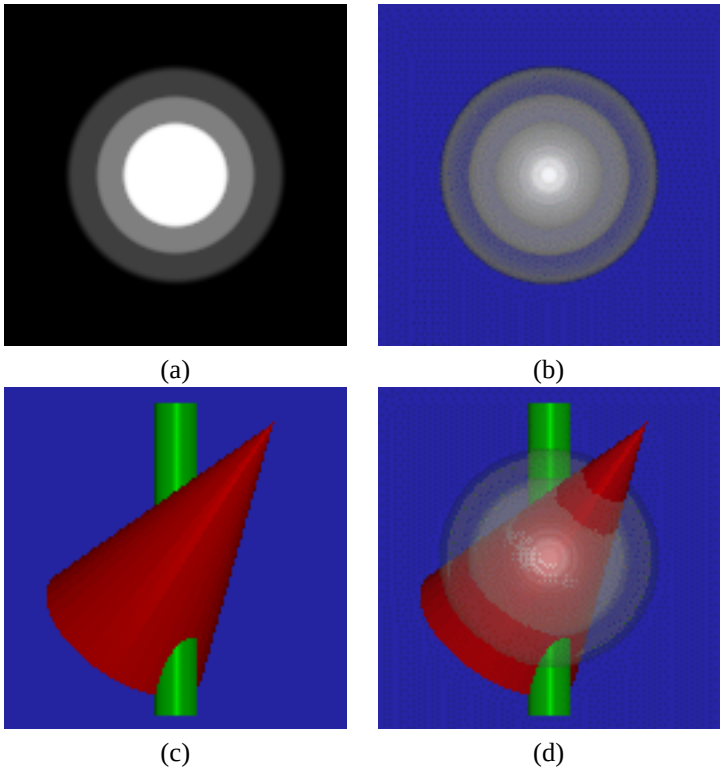


Figure 10. Synthetic volume and geometry data used for opacity error quantification. (a) Cross-section through the concentric spheres volume data. (b) Full ray-cast of (a) at global opacity 0.5. (c) Geometry data consisting of a cylindrical bar and cone. (d) Intermixed volume and geometry (volume opacity 0.5).

compared with renderings at $\delta = 0$. The mean opacity error, rendering/compositing time (ms), and average storage were measured.

Fig. 29 shows variation in mean opacity error against global opacity t at different approximations (δ). The comparison was made against a full ray cast image. The error

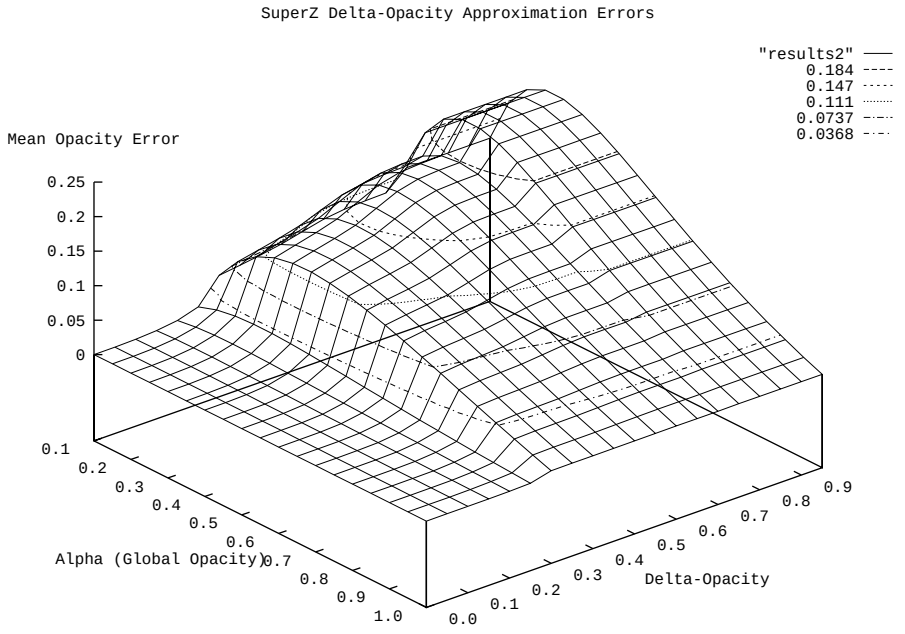


Figure 11. The variation in mean opacity error against global opacity t at different approximations (δ). The comparison was made against a full ray cast image. The error increases both with the degree of compositing approximation used ($\delta > 0$) and lower object opacity.

increases both with the degree of compositing approximation used ($\delta > 0$) and the lower the object opacity. This confirms the supposition that the light transmittance effects at higher transparencies are compromised at higher δ thresholds. The steps at $\delta = 0.3$ and $\delta = 0.7$ are probably data dependent. The contour lines show that the average per-pixel opacity error can be kept below roughly 0.1 at $\delta < 0.5$ and for $t_{volume} > 0.5$.

The variation in mean opacity error for different volume-geometry opacities for $\delta = 1.0$ (zeroth order super-z) is shown in Fig. 30. The comparison was made against a full ray cast image. The error minimizes with increasing volume and geometry opacity, i.e., the more solid the objects are. The error peaks at a geometry opacity of around 0.3. The effect of the volume opacity on the overall error is fairly linear. The mean error can be kept below about 0.1 per pixel for semi-transparent renderings when $t_{volume} > 0.5$ and $t_{geometry} > 0.4$.

4.2. Super-Z list sizes

The final FTB compositing in our scheme is relatively low both in storage and calculation cost because of the sparse sampling along the ray that the super-z list sets produce. For $\delta = 1$, we investigated typical list sizes for segmented and unsegmented data sets. Our segmented data was volume rendered as voxel shells of four voxel depth, which was particularly amenable to run-length encoding and our method. The unsegmented volumes were presented in raw form to the built-in classifier. The depth complexity of the super-z buffers for two clinical data sets are given in tables 1 and 2.

Table 1 shows the depth complexity of the segmented MR knee data shown in Fig. 16 which consists of approximately 10.8M voxels (equivalent to a cubic data set of size $220 \times 220 \times 220$). It shows that the rendering can be made with a maximum of 17 opacity segments at any pixel, out of a potential of around 256, and with an average complexity of just under 3 segments over the entire picture plane. The numbers for the anatomical parts are for a left view and the bottom two figures are for left and anterior views. Table 2 shows the depth complexity of the multi-modality surgical guidance case consisting of 25.6M voxels (equivalent to cubic a data set of size $295 \times 295 \times 295$). Again, the maximum super-z depth does not exceed 18 and the average is around 3. If the data is fairly homogeneous then only the interfaces between materials have an impact on the depth complexity which is linear with the number of possible objects encountered along the ray path, but if there is great variation in the opacity then the opacity approximations made might become visible.

An implementation requires, for each segment, the storage of only the start z value $s[k]$ and the end opacity $A_m[s[k]]$ and color values $C_\lambda[e[k]]$, which amounts to one short integer and 4 floating point values (16 bytes). In our examples with $\delta = 1$,

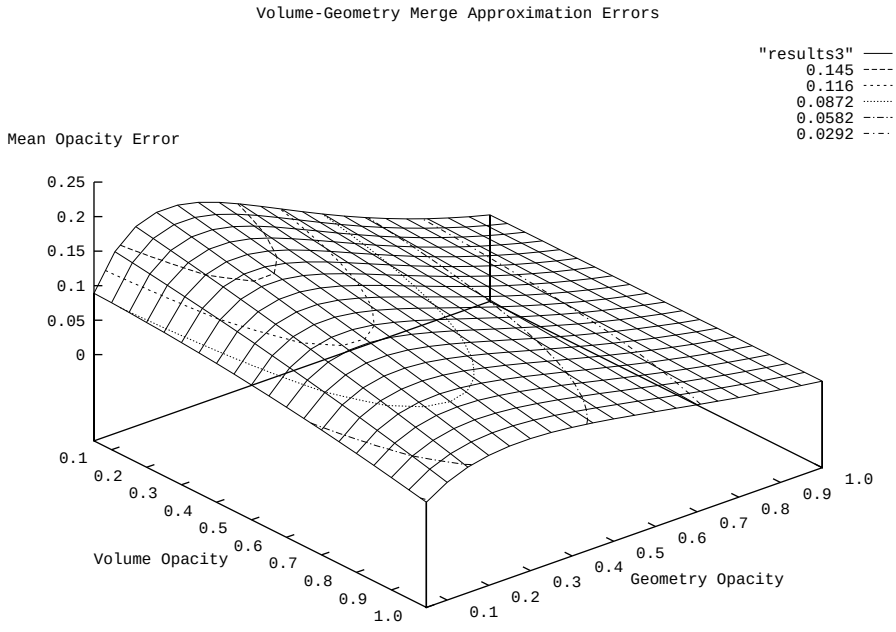


Figure 12. The variation in mean opacity error for different volume-geometry opacities for $\delta = 1.0$ (zeroth order super-z). The comparison was made against a full ray cast image. The error minimizes with increasing volume and geometry opacity i.e. the more solid are the objects. The error peaks at a geometry opacity of around 0.3. The effect of the volume opacity on the overall error is fairly linear.

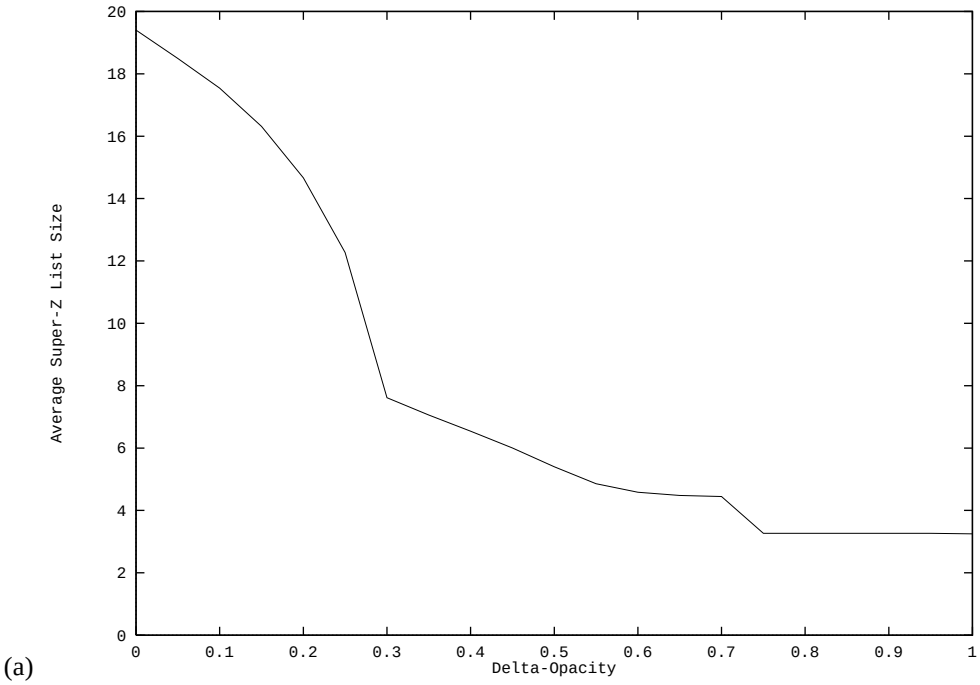
Sub-volume size	Anatomy	Voxel-Segments			(MBytes) Ave. Storage
		Total	Max.	Ave.	
$106 \times 162 \times 93$	femur	14305	8	1.16	1.7
$96 \times 112 \times 85$	tibia	14213	9	1.68	2.4
$50 \times 71 \times 33$	fibula	3008	5	1.22	1.8
$40 \times 72 \times 48$	patella	2702	7	1.23	1.8
$115 \times 87 \times 89$	fem. cart.	7023	8	1.49	1.7
$91 \times 42 \times 84$	tib. cart.	3336	10	1.25	1.8
$34 \times 59 \times 50$	pat. cart.	1599	6	1.09	1.6
$239 \times 237 \times 118$	skin	84545	17	1.93	2.9
$220 \times 220 \times 220$	all (left)	13071	17	2.91	4.2
10.8M voxels	all (anterior)	79402	15	2.98	4.3

Table 1. Depth complexity values (in total voxel-segments, and maximum/average per view-plane pixel) for rendering of the MR knee data. Average total storage calculated using 22 bytes per node for a 256×256 output image.

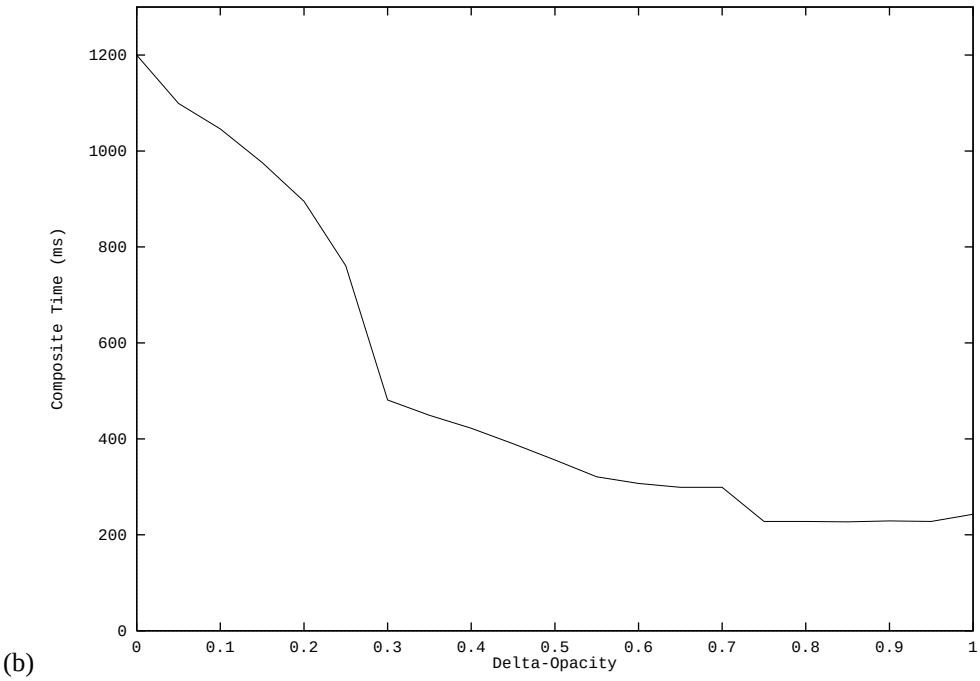
the average super-z list length (per output pixel) is 3, and if we assume the storage of one pointer (4 bytes) per depth list node yields a total of $(18 + 4) \times 3 = 66$ bytes per pixel. The total storage overhead for a 256×256 output image is approximately 4.3M bytes.

4.3. Computational Costs

Creation of the super-z list structure requires only compare operations (on average 3 per pixel). The lists are automatically sorted by insertion sorting, i.e., the next depth node placed in the correct point along any existing list. The tabulated figures from the typical data suggests at most 18 compares ("greater than" operation) for any segment insertion, and on average only 3 compares per output pixel. A



(a)



(b)

Figure 13. Plots showing effect of various opacity approximation thresholds (δ-Opacity) against: (a) Average Super-z list length, (b) Composite time (ms). Comparing (a) and (b) there is clearly a 1:1 relationship between compositing time and the length of the super-z list at each pixel. In (a), the list size reduces dramatically from an average of 20 to about 3 after an approximation $\delta = 0.3$ or greater. The steps at $\delta = 0.3$ and $\delta = 0.7$ are probably data dependent (see main text for explanation).

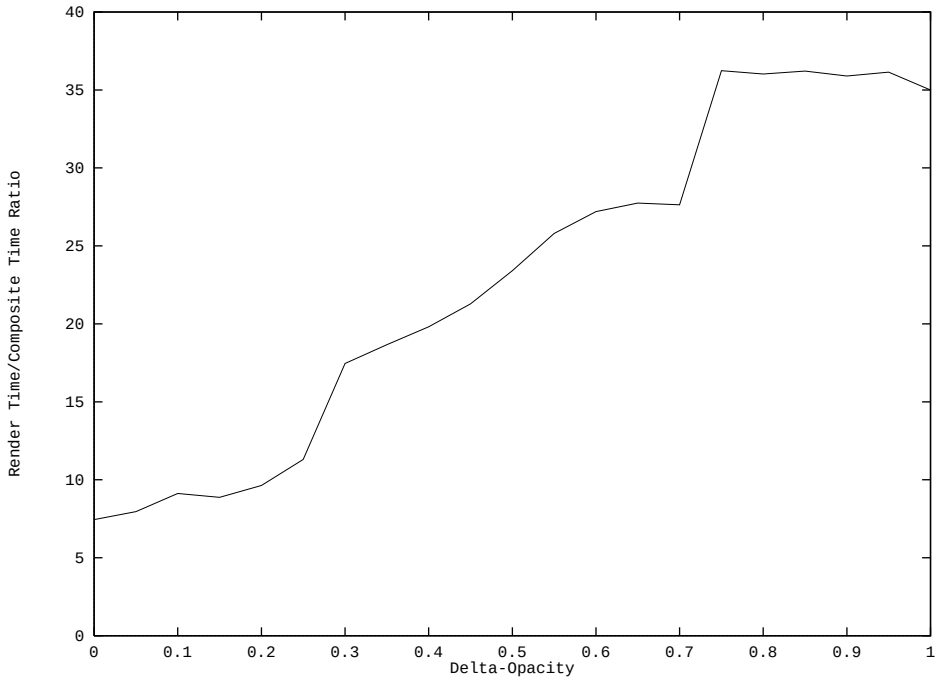


Figure 14. Plot showing effect of various opacity approximation thresholds (δ-Opacity) against ratio of Render/Composite time. The best Render/Composite ratios are at low approximation thresholds. The relationship is roughly linear ranging from 7 times to 35 times faster compositing.

Sub-volume size	Anatomy	Voxel-Segments			(MBytes) Ave. Storage
		Total	Max.	Ave.	
$256 \times 217 \times 113$	skin	47845	18	1.08	1.6
$225 \times 197 \times 101$	grey	19122	5	1.09	1.6
$223 \times 195 \times 102$	white	15688	4	1.07	1.5
$256 \times 256 \times 122$	vessels	18805	7	1.36	2.0
$109 \times 184 \times 59$	tumor	1550	2	1.01	1.5
$167 \times 161 \times 46$	cyst	169	1	1.00	1.4
$295 \times 295 \times 295$	all (left)	103179	18	2.33	3.4
25.6M voxels	all (anterior)	54419	17	2.69	3.9
	all (superior)	59894	13	2.86	4.1

Table 2. Depth complexity values (in total voxel-segments, and maximum/average per view-plane pixel) for rendering of surgical data: grey/white matter from post-contrast SPGR, tumor from T2 weighted MR, and vessels from MRA. Average total storage calculated using 22 bytes per node for a 256×256 output image.

binary chop-type sort method could be used to reduce the overall list creation time.

To assess the computation required for re-compositing, we apply eqns. 9 and 11, (5 multiplies and 6 additions) iteratively, again on average, 3 times per pixel. On average, this requires a total of 15 multiplies and 18 additions per pixel. For a fixed viewpoint the computation for re-compositing is thus quite low. We have found that it is possible to turn objects off and on, or change their opacities at interactive rates, on a moderately powered workstation (Sun Ultra Sparc 167MHz, with 256M RAM).

An analysis of typical timings for single and multi-CPU machines is given in Table 3. The timings are given for rendering, compositing, and warping stages for each frame using an implementation of our method with Lacroute's Shear-Warp factorization. Note that although Shear-Warp

Stage	Timing (ms)	
	Ultra SPARC	8-CPU Ultra
Render	10653	1506
Composite	422	60
Warp	65	16

Table 3. Example of computation costs for super-z rendering. An implementation using Lacroute's VolPack rendering system with shear-warp factorization was used. Timings are given for a 256×256 pixel output image on the knee data set consisting of 10.8M voxels in total (see Table 1). The right column gives results for a multi-threaded implementation on a 8-CPU SMP machine.

is view-independent, our method cannot be, as the super-z structure is not built until the compositing stage, which happens after the shear and per-slice interpolation within the volume. A detailed analysis of computation and timings for the rendering and warping are given by Lacroute in (Lacroute and Levoy, 1994) (Lacroute, 1996). Our results are given for color rendering (3 channel) and are proportionally slower than Lacroute's as our multi-threaded implementation was added as an extension to the original library and is therefore not fully optimized.

For the knee data ($\delta = 1$), the ratio for re-compositing to rendering is approximately $1:25$. The timings show a worst-case re-compositing when all object colors are re-composited. Further optimization can be achieved by restricting the 2D composite operation to be within the pre-calculated bounding boxes of the modified object. Fig. 31 shows experimental timings and storage requirements across a range of opacity approximation values $0 \leq \delta \leq 1$. This gives an indication of the effect of greater approximation on the average super-z list size (per pixel) and the corresponding composite time. Not surprisingly, comparing Figs. 31(a) and (b) reveals a 1:1 relationship between list size and composite time. In Fig. 31(a), the list size reduces dramatically from an average of 20 to about 3 after an approximation $\delta = 0.3$ or greater. The steps at $\delta = 0.3$ and $\delta = 0.7$ are probably data dependent due to the opacity peaks in the chosen image.

The ratio of Render/Composite time is plotted in Fig. 32 for a range of δ values, for the 'spheres' data set. As the approximation is made worse (increasing *delta*), the Render/Composite time ratios increase from about 7 to 35. The relationship is roughly linear over the $0 \leq \delta \leq 1$ range.

4.4. Results on Medical Image Data

Typical renderings of a segmented MR knee scan, size $256 \times 256 \times 118$ are presented in Figs. 16(a)-(c). Fig. 16(a) shows the skin semi-transparent revealing the bones (femur, tibia, fibula and patella). The skin has been peeled away in Fig. 16(b). In Fig. 16(c), the femur and tibia have been made semi-transparent to showing the inside surfaces of the femural and tibial cartilage as it wraps around the ends of the joint. Fig. 16(d) shows the overlaps in the super-z depth buffer for the same view. Most of the image has the original anatomical colors, with mainly red (indicating an overlap of 2), in the bone-cartilage areas and then green (overlap of 3) and blue (overlap of 4) at the joint and behind the knee cap. There is sporadic color indicating higher depth complexity (white is greater than 6). A polygon rendered sphere is added to the knee renderings in Fig. 16(e) and (f). Correct interpenetration of the surface of the sphere with the bones is achieved. In Fig. 16(f) the sphere is wireframed and the shading and correct depth compositing are unaffected.

A series of renderings for a multi-modality surgical planning/guidance case are presented in Fig. 17. The grey and white matter were segmented from post-contrast SPGR MR, the tumor from a T2 weighted MR. The blood vessels were from a separate MR Angiogram (MRA) that was later co-registered to the tissue scans. Moving from Fig. 17(a) to (d), superficial anatomy has been successively made transparent and then turned off to reveal the tumor and its relation to the vasculature in 17(d). Note that the vessels in this rendering are presented unsegmented as they have a sufficiently high contrast. In Fig. 17(e) and (f), the volumetric model of the skin (rendered as a 4-voxel thick shell) are replaced by a polygon surface model of 43,104 triangles. Comparing Fig. 17(e) and (a) shows how much smoother the skin surface is as it does not show the aliasing artifacts visible in (a) which is a result of the poorer out-of-plane resolution of the scanned data. In Fig. 17(f), the skin is made semi-transparent to reveal the unclassified volume rendered blood vessels and the volumetric, segmented tumor.

By way of illustration of a haptic scenario, in Fig. 18 we show a simulated probing of a volumetric rendered tibia from the knee data set. In these images, the probe is represented by a polygonal cylinder and sphere (a total of 1160 triangles), which is separately rasterized by standard frame buffer graphics and combined using super-z rendering. As the haptic probe is moved around the scene, only re-compositing in the bounding box of the polygon rendered objects is required. We have not attempted to perform collision detection and haptics (in its true sense) in this simulation (Pflesser *et al.*, 1991). Note that although the depth coding that the super-z produces is more efficient for segmented data (such as the knee bones), it does not exclude mixing unclassified volumes. If a fast volume renderer is available then qualitative explorations of heterogeneous data is possible, independent of subsequent interaction in the fixed view.

5. Conclusions

A technique for quickly re-rendering volume data consisting of several distinct materials and intermixed with moving geometry was presented. An intermediate depth based buffering scheme, which stores color and differential opacity information to a given approximation was described. The method enables rapid transparency and color changes of materials, position changes of sub-objects, dealing explicitly with regions of overlap, and the intermixing or separate rendering of moving geometry, without re-scanning the entire volume. The main assumption throughout is that the viewpoint is fixed, or at least still during interaction. We presented rendering results using both synthetic volumes and models, and typical clinical data which was used for surgery planning.

The rendering quality can be traded-off against the relative storage cost, and we gave an empirical analysis of output error together with typical figures for its storage complexity. Reducing the approximation threshold, which controls the degree of opacity encoded by the depth buffering, correspondingly reduces the output error. At the same time, the amount of intermediate storage and compositing time increases. Although the error increases for higher transparency compositing, our results show that average per-pixel opacity errors can be kept below 0.1 for variations in transparency above about half for both volume and geometry. In our experiments, the render/re-compositing time ratio ranged between 7 and 40 for the best and worst case approximation.

Both the ability to rapidly re-generate renderings of changing data, and to effectively combine volume and surface graphics have important implications for intra-operative visualization and haptic rendering systems, e.g., surgical simulators, where the fixed view limitation is less important. Although our methodology can support strategies for dealing with overlapping regions, we have not investigated their effect on quality and performance, and this might be a good topic for further study. We acknowledge that acceleration methodologies like ours may become obsolete as general-purpose techniques become faster or less expensive, but believe that our method will fill a gap in current technology. Furthermore, the use of intermediate depth structures provide one practical way of combining volume and surface graphics using separate volume and geometry rendering engines.

Acknowledgements

We would like to sincerely thank Carl-Fredrik Westin (SPL) for discussions on this work and his help in revision of this manuscript. We also thank the referees for their constructive and helpful comments.

The work is funded in part by the collaborative Sun Microsystems/SPL Cluster Computing Project.

References

Avila, R. and Sobierajski, L. (1996). A haptic interaction method for volume visualization. In *Proc. of IEEE VIS'96*, pp. 197–204.

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

Brady, M. L., Higgins, W. E., and Ramaswamy, K. (1995). Interactive navigation inside 3d radiological images. In *Proc. IEEE Visualization 95*, pp. 33–40.

Ebert, D. and Parent, R. (1990). Rendering and animation of gaseous phenomena by combining fast volume and scanline A-buffer techniques. In *Proc. SIGGRAPH'90*.

Geiger, B. and Kikinis, R. (1995). Simulation of endoscopy. In *Comp. Vision Virtual Reality and Robotics in Med.*, pp. 227–281, Nice, France.

Gibson, Sarah F. F. (1997). 3D chainmail: a fast algorithm for deforming volumetric objects. In Cohen, Michael and Zeltzer, David (eds), *1997 Symposium on Interactive 3D Graphics*, pp. 149–154. ACM SIGGRAPH. ISBN 0-89791-884-3.

Gortler, S. J., He, L-W, and Cohen, M. F. (1997). Rendering layered depth images. Technical Report MSTR-TR 97-09, Microsoft Inc.

Jacq, J-J and Roux, C. (1996). A direct multi-volume rendering method. application to visual assessment of 3-D image registration algorithms. In *Proc. of Vis. in Biomed. Computing. Lecture Notes in Computer Sciences 1131*, pp. 53–62, Berlin, Germany.

Jacq, J-J and Roux, C. (1997). A direct multi-volume rendering method aiming at comparisons of 3-D images and models. *IEEE Trans. on Information Technology in Biomedicine*, 1(1), 30–43.

Kaufman, A., Yagel, R., and Cohen, R. (1990). Intermixing surface and volume rendering. In *K. H. Hoehne et. al. (eds), 3D Imaging in Medicine: Algorithms, Systems, applications*, Vol. 60, pp. 217–227, Berlin. Springer-Verlag Berlin.

Lacroute, P. (1995). *Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transformation*. *Ph.D. Thesis*, Stanford University.

Lacroute, P. (1996). Analysis of a parallel volume rendering system based on the shear-warp factorization. *IEEE Visualization and Computer Graphics*, 2(3), 218–231.

Lacroute, P. and Levoy, M. (1994). Fast volume rendering using a shear-warp factorization of the viewing transform. In *Comp. Graph. Proc., Ann. Conf. Series SIGGRAPH '94*, pp. 451–458, Orlando.

Levoy, M. (1988). Display of surface from volume data. *IEEE Computer Graphics and Applications*, 8(3), 29–37.

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

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

Machiraju, R. and Yagel, R. (1993). Efficient feed-forward volume rendering techniques for vector and parallel processors. In *Proc. Supercomputing '93*, pp. 699–708.

Massie, T. and Sailsbury, K. (1994). The phantom haptic interface: A device for probing virtual objects. In *Proc. ASME Symp. on Haptic Interfaces for Virtual Environments and Teleoperator Systems*, Chicago.

Neider, J., Davis, T., and Woo, M. (1992). *OpenGL Programming Guide*. Addison-Wesley Reading, MA.

Pfister, H., Hardenbergh, J., Knittel, J., Lauer, H., and Seiler, L. (1999). The VolumePro real-time ray casting system. In *Proc. SIGGRAPH 1999*.

Pfister, H. and Kaufman, A. (1996). Cube-4 - a scalable architecture for real-time volume rendering. In *Proc. ACM/IEEE Symposium on Volume Visualization*, pp. 47–54, San Francisco, CA.

Pflesser, B., Tiede, U., and Hoehne, K. H. (1991). Volume based object manipulation for simulation of hip joint motion. In *H. U. Lemke et. al. (eds), Proc. CAR'91*, pp. 329–335, Berlin. Springer-Verlag Berlin.

Sobierajski, L. M. and Kaufman, A. E. (1994). Volumetric ray tracing. In *Proc. Symposium on Volume Visualization*, Washington, D.C, USA.

Umans, C., Halle, M., and Kikinis, R. (1997). Multilayer images for interactive 3d visualization on the world wide web. Technical Report 51, Surgical Planning Lab., Harvard Medical School, Boston MA.

Warfield, J. Dengler S., Zaers, J., Guttmann, C. R. G., Wells, W. M., Ettinger, G. J., Hiller, J., and Kikinis, R. (1995). Automatic identification of grey matter structures from MRI to improve the segmentation of white matter lesions. In *Proc. Medical Robotics and Computer Assisted Surgery*, pp. 140–147, Baltimore, USA.

Wells, W. M., Grimson, W. E. L., Kikinis, R., and Jolez, F. A. (1996a). Adaptive segmentation of MRI data. *IEEE Trans. Med. Imag.*, 15(4), 429–443.

Wells, W. M., Viola, P., Atsumi, H., Nakajima, S., and Kikinis, R. (1996b). Multi-modal volume registration by maximization of mutual information. *Medical Image Analysis*, 1(1), 35–51.

Westermann, Rüdiger and Ertl, Thomas (1998). Efficiently using graphics hardware in volume rendering applications. In Cohen, Michael (ed.), *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pp. 169–178. ACM SIGGRAPH, Addison Wesley. ISBN 0-89791-999-8.

Wilhelms, J. and van Gelder, A. (1991). A coherent projection approach for direct volume rendering. *ACM Trans. on Computer Graphics*, 25(4), 275–284.

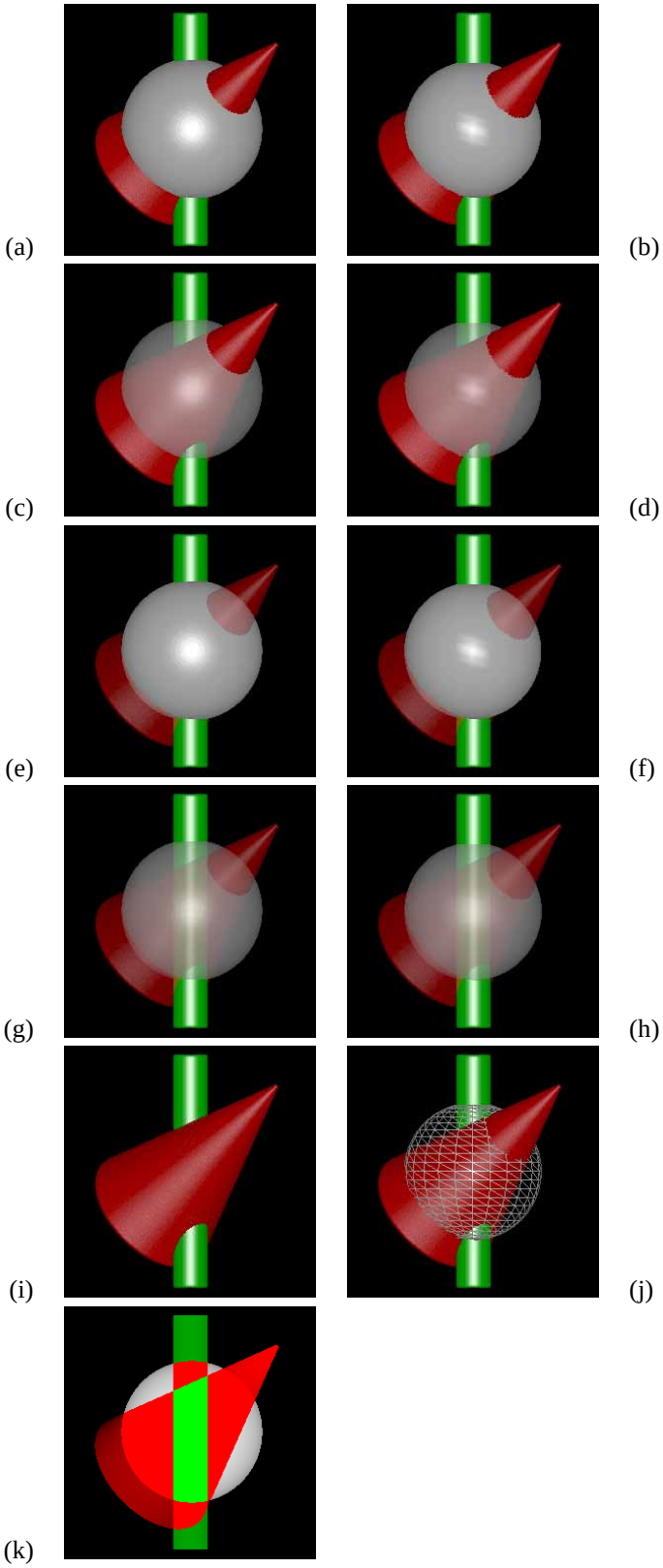


Figure 15. Left column: super-z volume renderings of synthetic volumes: sphere, cone and cylindrical bar. Right column: combined volume-geometry using super-z, with sphere polygon rendered. In (j) the wireframe (1200 triangles) defining the sphere is shown. In (k), where objects overlap the following color coding is used: red, green, blue, yellow, cyan, magenta, for overlaps of 1, 2, 3, 4, 5 and 6; and white for overlaps of greater than 6.

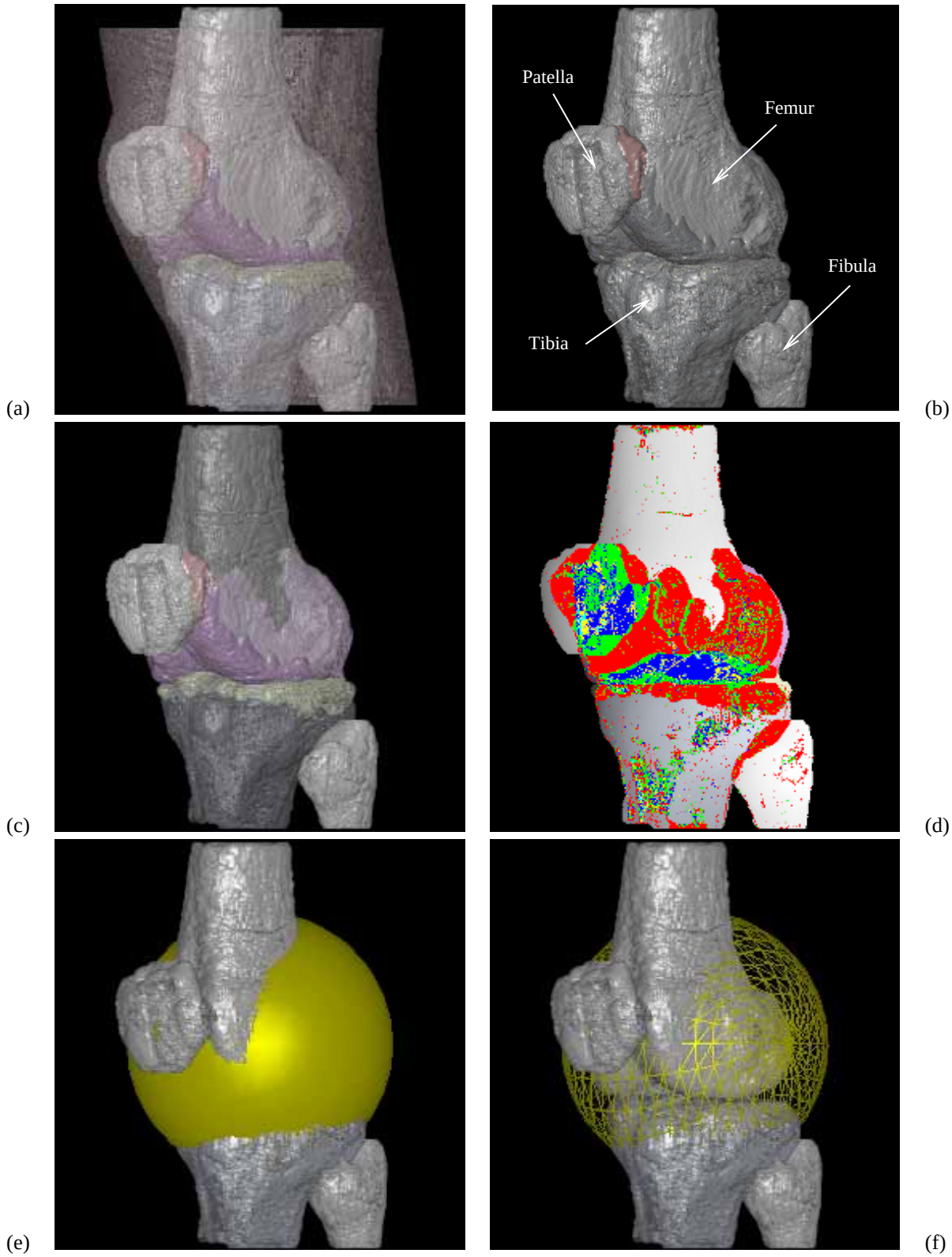


Figure 16. Super-z Volume Renderings of MR Knee data. Figs. (a)-(d) are all segmented volumetric models. In (d), where objects overlap the following color coding is used: red, green, blue, yellow, cyan, magenta, for overlaps of 1, 2, 3, 4, 5 and 6; and white for overlaps of greater than 6. Figs. (e) and (f), a surface rendered sphere is introduced showing correct interpenetration with bones. In (f) the sphere is show as a wireframe revealing triangle facets.

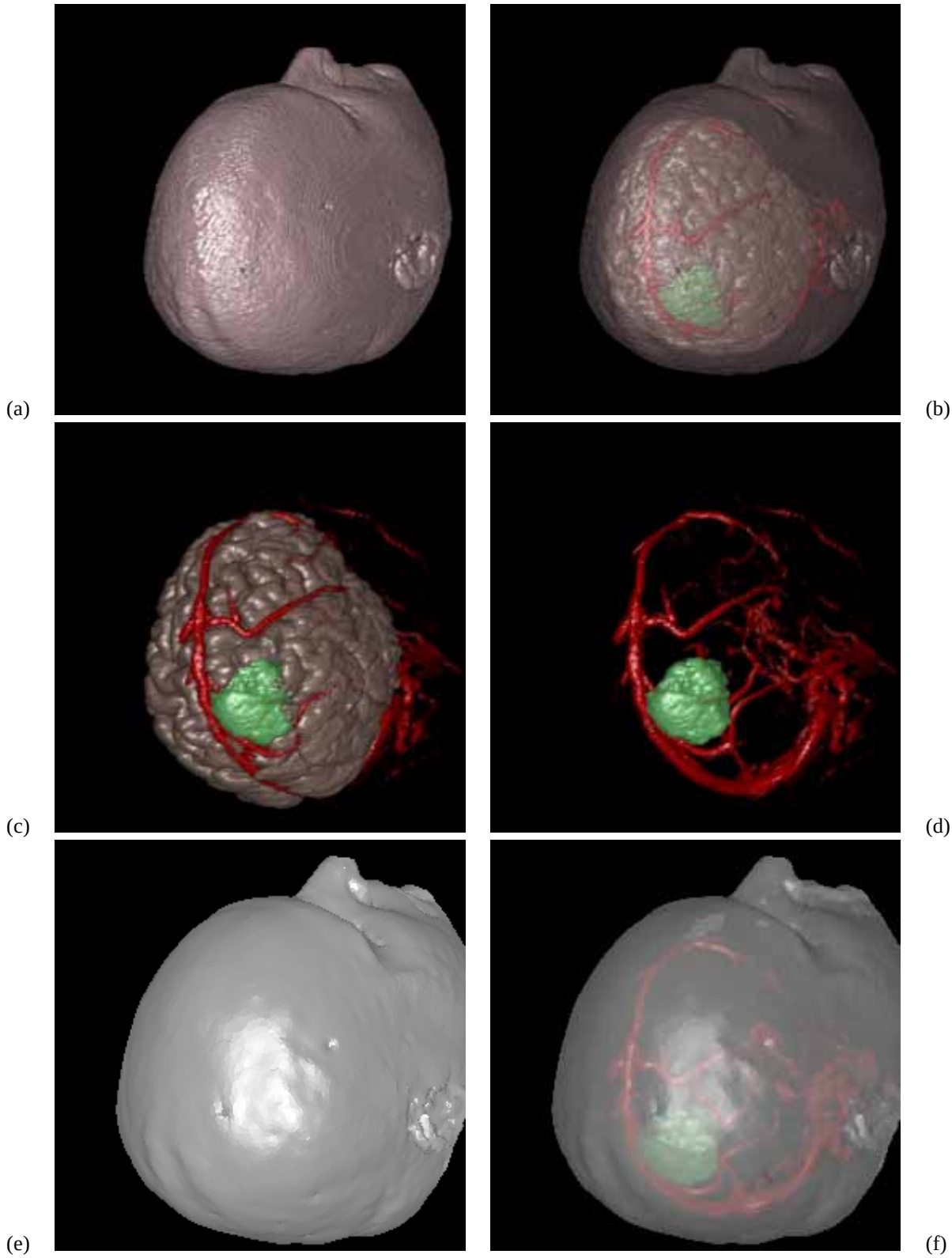


Figure 17. Super-z Volume/Surface Renderings of clinical surgical planning case. Figs. (a)-(d) with all volumetric models (skin as 4 voxel shell). Figs. (e) and (f) with skin model as surface shaded polygons (43104 triangles) and combined with unclassified MR angiogram and segmented tumor.

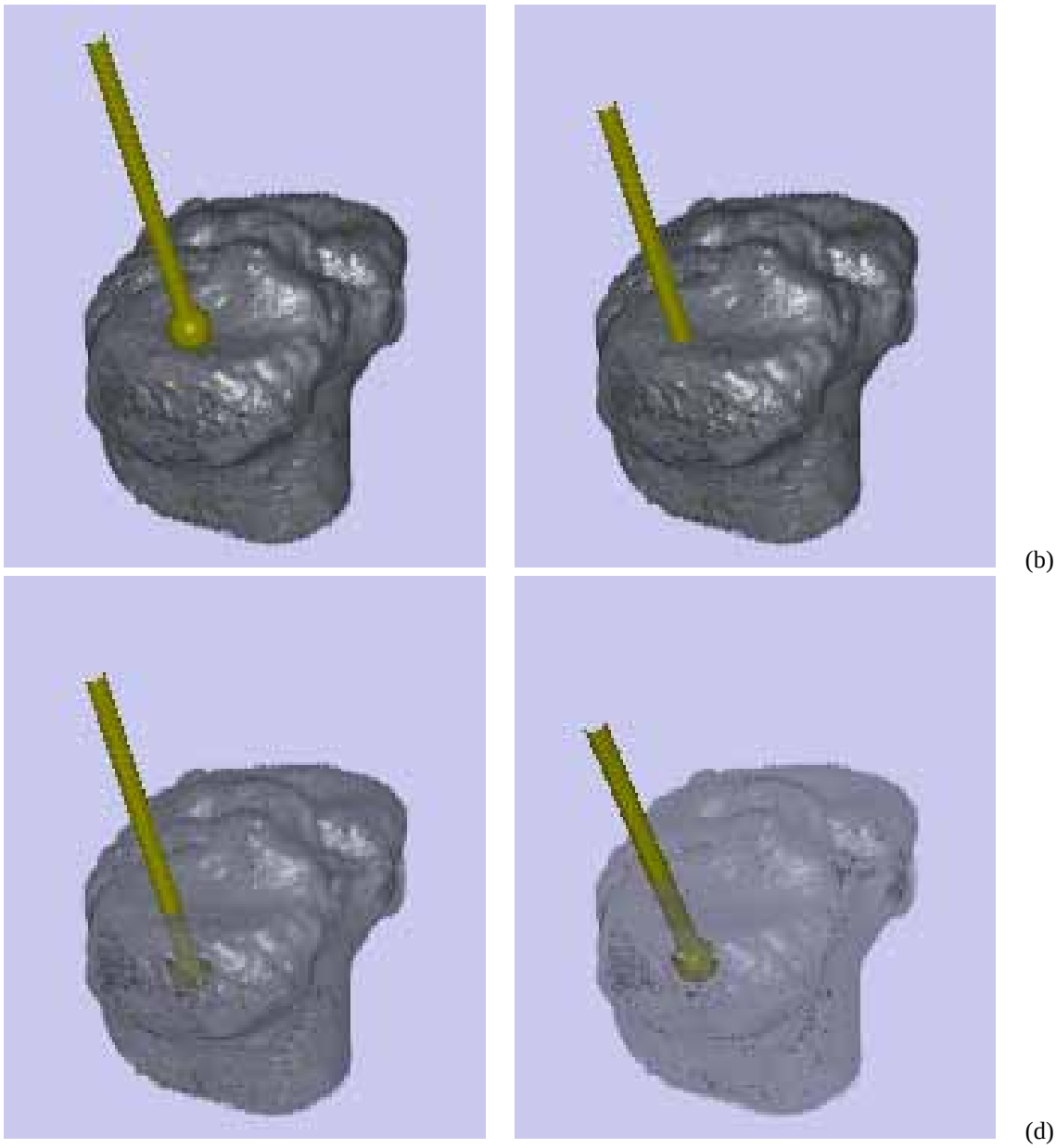


Figure 18. Four frames from a simulated haptic probing of the surface of a volume rendered tibia from the knee joint data set. The probe is represented by a polygonal cylinder and sphere (1160 polygons), separately rendered by polygon accelerated frame graphics. Super-z is used to combine the renderings as the probe is moved and breaks the tibial surface. In (d) transparency of the bone was reduced to show the probe inside the bone.

Abhir Bhalerao graduated in 1986 from the University of Warwick, UK, having read BSc Computer Systems Engineering. He worked as a programmer with a software-systems company (SD-Scicon) for two years being involved in real-time simulations software e.g. graphical, networked systems for air traffic control training. Between 1988-1991 he was a researcher at the Image and Signal Processing group at the Department of Computer Science, University of Warwick, where he completed his Ph.D. in Image Segmentation. He worked for an electronics company (Crosfield Electronics) for 2 years as an analyst programmer and was involved in a number of projects developing low-level software for communications systems used by the publishing industry. In 1993 he joined the Image Processing Group at Guy's and St.Thomas' Hospital, London, at part of the Advanced Medical Imaging project, sponsored by the hospital's Special Trustees. Since February 1996, he has been a Research Fellow with the Surgical Planning Laboratory, Brigham and Women's Hospital, Harvard Medical School, Boston working on segmentation methods for MR angiography data, volume rendering for surgical planning, and artifact correction in MRA.

Hanspeter Pfister is a Research Scientist at MERL - A Mitsubishi Electric Research Laboratory in Cambridge, MA. He is the chief architect of VolumePro, Mitsubishi Electric's real-time volume rendering system for PC-class computers. His research interests include computer graphics, scientific visualization, computer architecture, and VLSI design. Hanspeter Pfister received his PhD in Computer Science in 1996 from the State University of New York at Stony Brook. In his doctoral research he developed Cube-4, a scalable architecture for real-time volume rendering. He received his Dipl.-Ing. degree in electrical engineering from the Department of Electrical Engineering at the Swiss Federal Institute of Technology (ETH) Zurich in 1991. He is a member of the ACM, IEEE, the IEEE Computer Society, and the Eurographics Association.

Michael Halle is a Research Fellow at the Surgical Planning Laboratory in the Department of Radiology at the Brigham and Women's Hospital in Boston, Massachusetts. He received his Bachelor of Science in computer science and engineering from MIT in 1988. He studied the role of computer graphics in three-dimensional display technology at the Spatial Imaging Group of MIT's Media Laboratory from 1985 to 1997. He received his Master of Science degree from MIT in 1991 and his doctorate in 1997. His current research interests include 3D displays, computer graphics, user interface design, and image based rendering.

Ron Kikinis is the Director of the Surgical Planning Laboratory of the Department of Radiology, Brigham and Women's Hospital and Harvard Medical School, Boston, MA, and an Assistant Professor of Radiology at Harvard Medical School, as well as an Adjoint Professor of Biomedical Engineering at Boston University. His interests include the development of clinical applications for image processing, computer vision and interactive rendering methods. He is currently concentrating on developing fully automated segmentation methods and introducing computer graphics into the operating room. He is the author of 44 peer-reviewed articles. Before joining Brigham and Women's Hospital in 1988, he worked as a researcher at the ETH in Zurich and as a resident at the University Hospital in Zurich, Switzerland. He received his M.D. from the University of Zurich, Switzerland, in 1982.

6. Figures on single pages

Except Figures 15 – 18 which are already on single pages.

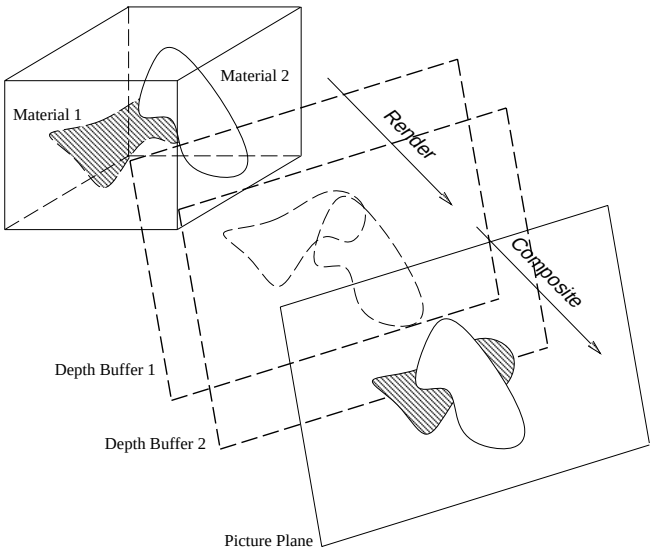


Figure 19. Overview of depth buffer volume rendering

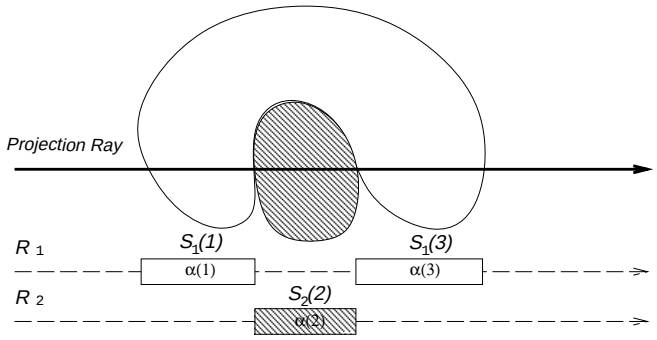


Figure 20. The same ray cast through two materials showing image segments encountered

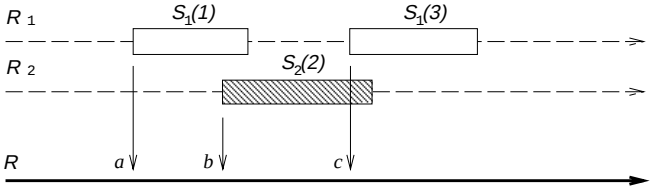


Figure 21. Merge sorting super-z lists on the front z value.

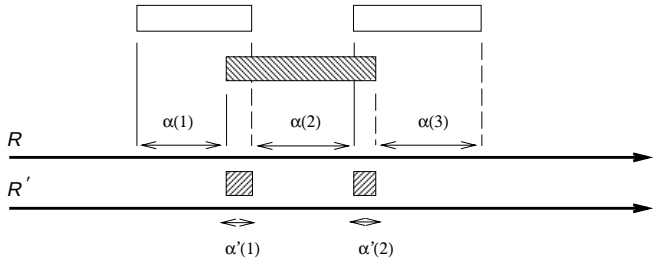


Figure 22. Overlapping sub-segments are handled by separate blending strategies.

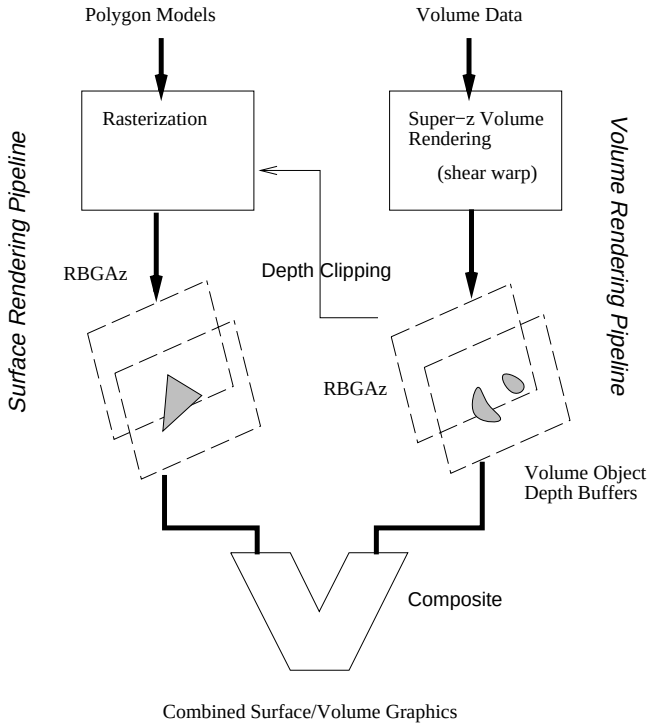


Figure 23. Mixing surface geometry and volume rendered graphics using super-z depth buffering. Polygon rasterization is controlled by the depth buffering. Polygon pixels and volume segment pixels are composited in an interleaved fashion by arithmetic and logic in the compositer.

1. Create a set Z of minimum and maximum segment start values taken over all output pixels for the current compositing.

   Z consists of values $Min(s[k])$ and $Max(s[k])$ from the merged ray sets $R[r,k]$, where $0 \leq k < N_{max}$, plus the depth values $(0,\infty)$ for the space up to and beyond the object:

$$Z = \{0, Min(s[0]), Max(s[0]), \ldots, Min(s[N_{max}-1]), Max(s[N_{max}-1]), \infty\} \tag{14}$$

   where N_{max} is the total number of merged ray sets.

2. Sort Z into ascending order:

$$Sorted(Z) = \{z[0], z[1], \ldots, z[N_z - 1]\} \tag{15}$$

   where $z[0] = 0$ and $z[N_z - 1] = \infty$.

3. For each pair of planes $z[p], z[p+1] \in Sorted(Z)$, $0 \leq p < N_z - 1$, interleave the geometry-pixel and the segment-pixel compositing as follows. All pixel operations are clipped against the current depth planes $z[p]$ and $z[p+1]$.

   (a) Send the depth hull: the $s[k]$ values of the segment-pixels, across the entire picture plane.

   (b) FTB composite rasterized geometry that is in front of the depth hull, but behind the current front plane $> z[p+1]$.

   (c) FTB composite segment-pixels

   (d) FTB composite rasterized geometry that is behind the depth hull, but in front of the current back plane $< z[p+1]$.

Figure 24. Algorithm for combining volume and surface graphics using super-z

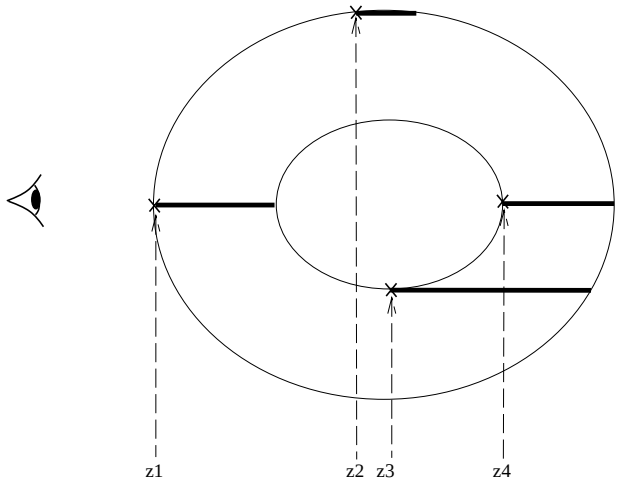
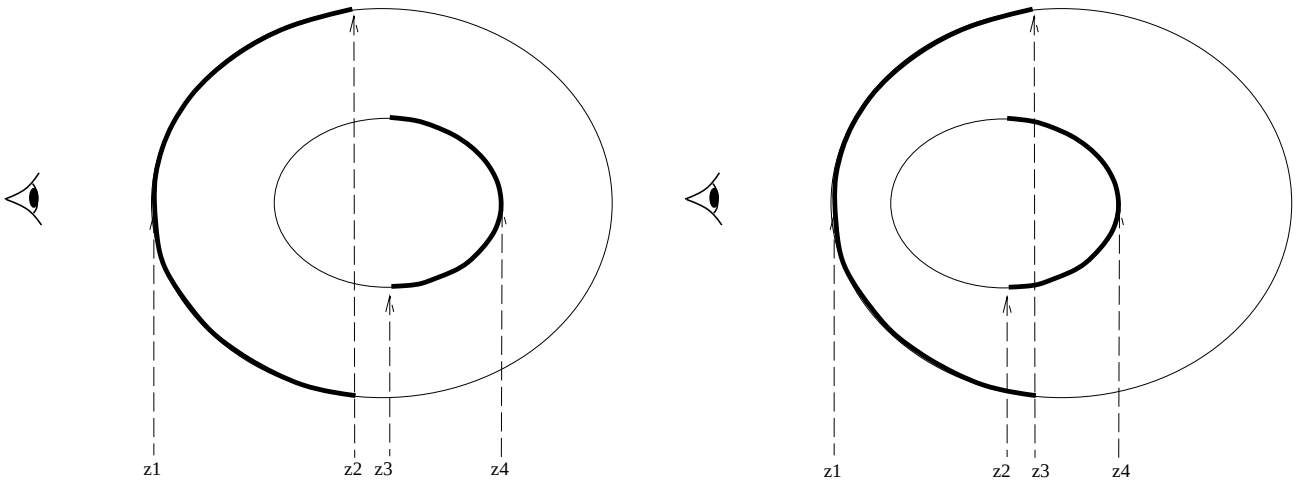


Figure 25. Minimum and maximum segment start depth values for a simple torus shaped volume are at: $z[1,2]$ for the first segment of all rays, and $z[3,4]$ for the second segment. Therefore, $z[1,2,3,4]$ define depth planes used to control the order of polygon rasterization. The pseudo planes $z = 0$ and $z = \infty$ are added to this list.



(a) (b)

Figure 26. Min-max segment values and depth hulls for torus shaped volumes seen side on. In (b) the inner hull overlaps the outer hull and z2 and z3 planes are reversed in comparison to (a).

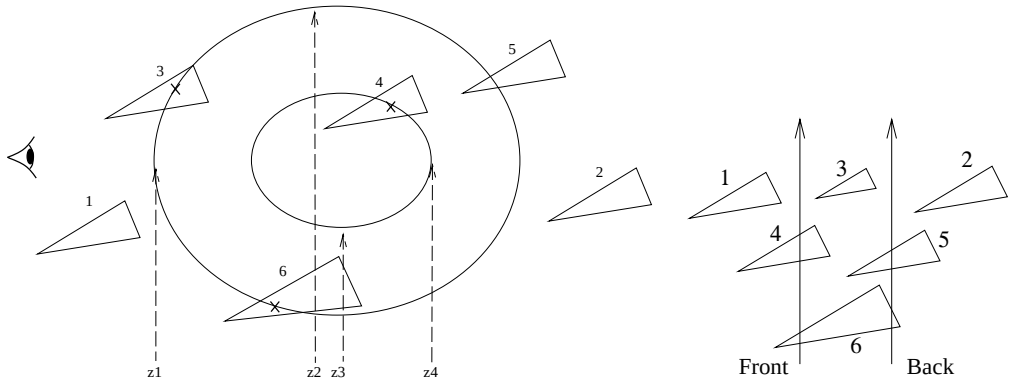


Figure 27. (a) Possible triangle positions rendered within an example volumetric scene. Types 1 and 2 lie in front and behind all voxel data. Triangle types 3 and 4 intersect surfaces of volume data. Type 5 intersects a "backface" and will be treated the same as type 2. Type 6 intersects the surface of a volume and straddles two clipping planes. (b) Triangle types clipped against a pair of depth planes. Types 1 and 2 are not sent. Types 3, 4, 5 and 6 are rasterized.

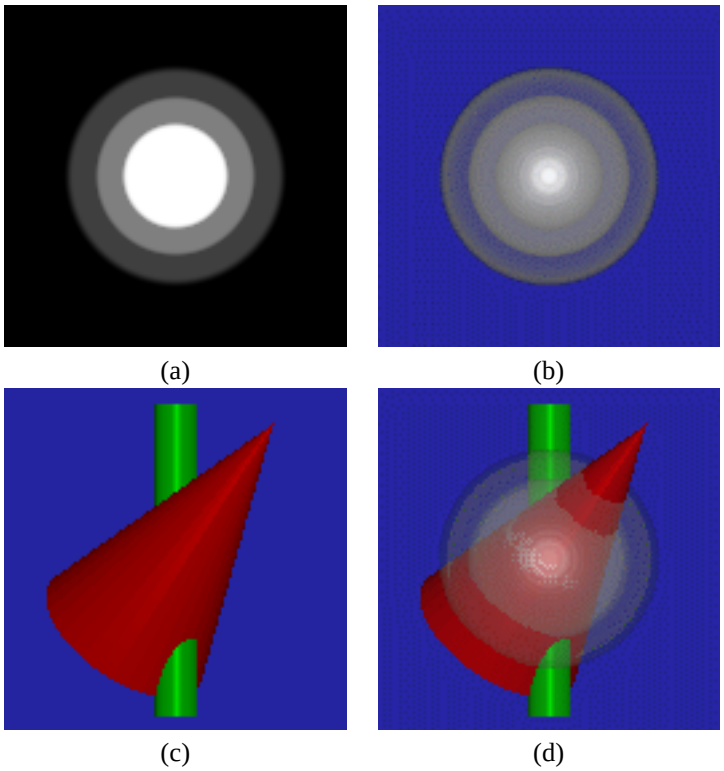


Figure 28. Synthetic volume and geometry data used for opacity error quantification. (a) Cross-section through the concentric spheres volume data. (b) Full ray-cast of (a) at global opacity 0.5. (c) Geometry data consisting of a cylindrical bar and cone. (d) Intermixed volume and geometry (volume opacity 0.5).

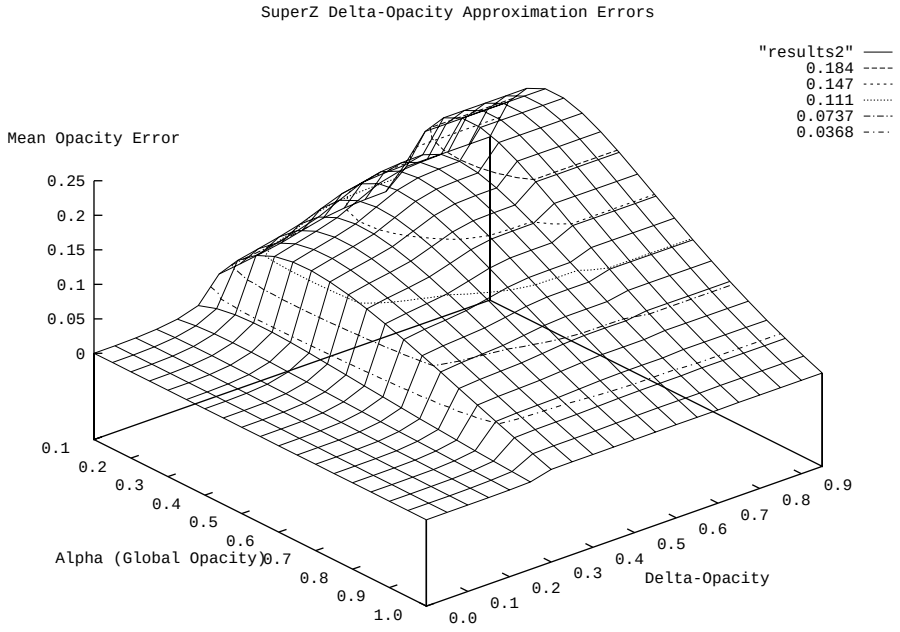


Figure 29. The variation in mean opacity error against global opacity t at different approximations (δ). The comparison was made against a full ray cast image. The error increases both with the degree of compositing approximation used ($\delta > 0$) and lower object opacity.

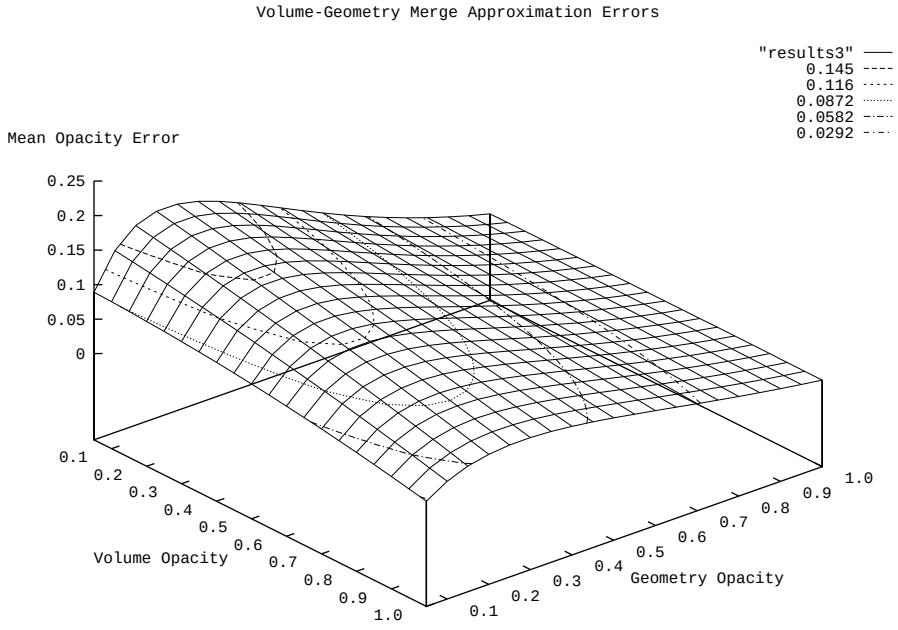


Figure 30. The variation in mean opacity error for different volume-geometry opacities for $\delta = 1.0$ (zeroth order super-z). The comparison was made against a full ray cast image. The error minimizes with increasing volume and geometry opacity i.e. the more solid are the objects. The error peaks at a geometry opacity of around 0.3. The effect of the volume opacity on the overall error is fairly linear.

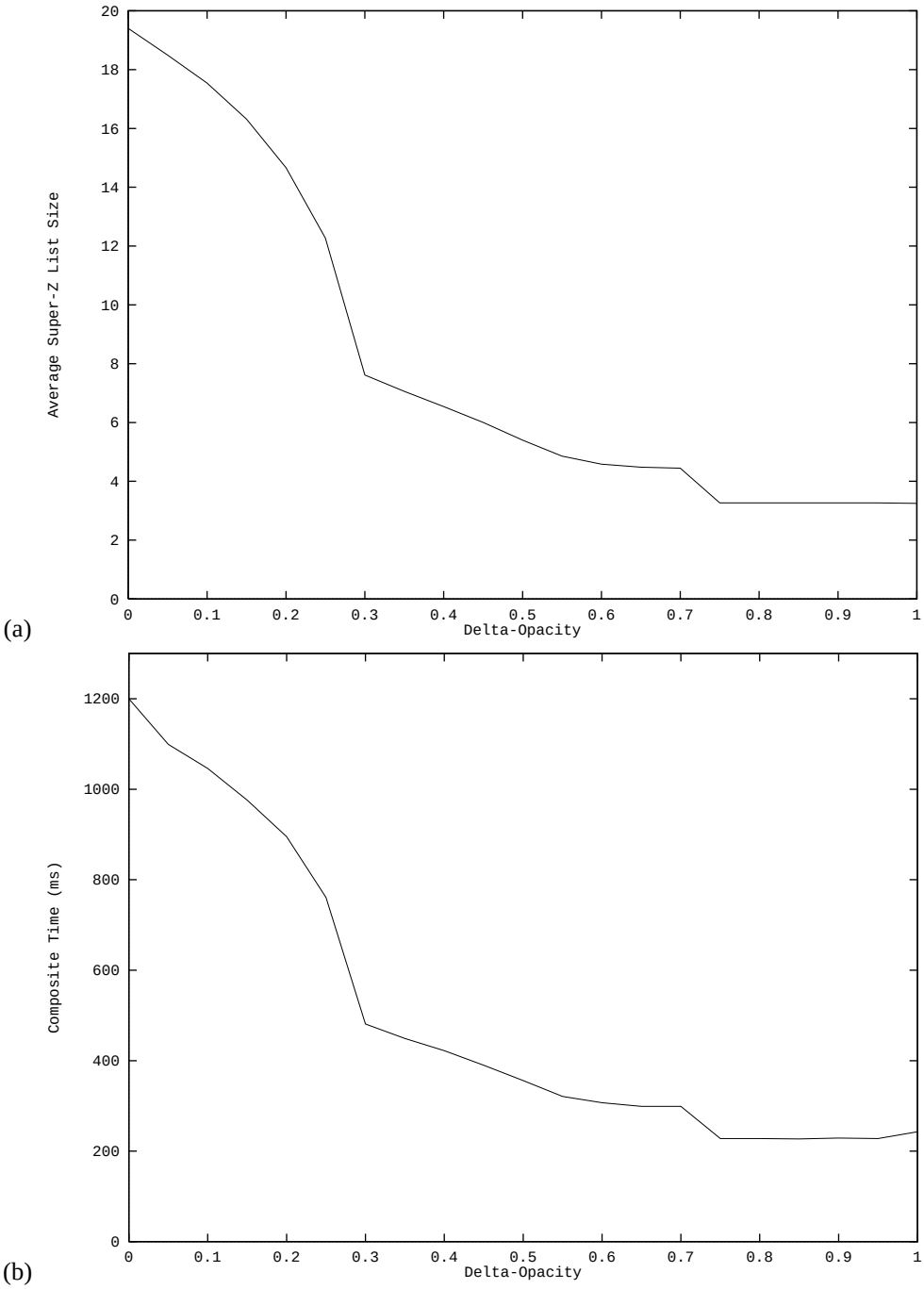


Figure 31. Plots showing effect of various opacity approximation thresholds (δ-Opacity) against: (a) Average Super-z list length, (b) Composite time (ms). Comparing (a) and (b) there is clearly a 1:1 relationship between compositing time and the length of the super-z list at each pixel. In (a), the list size reduces dramatically from an average of 20 to about 3 after an approximation $\delta = 0.3$ or greater. The steps at $\delta = 0.3$ and $\delta = 0.7$ are probably data dependent (see main text for explanation).

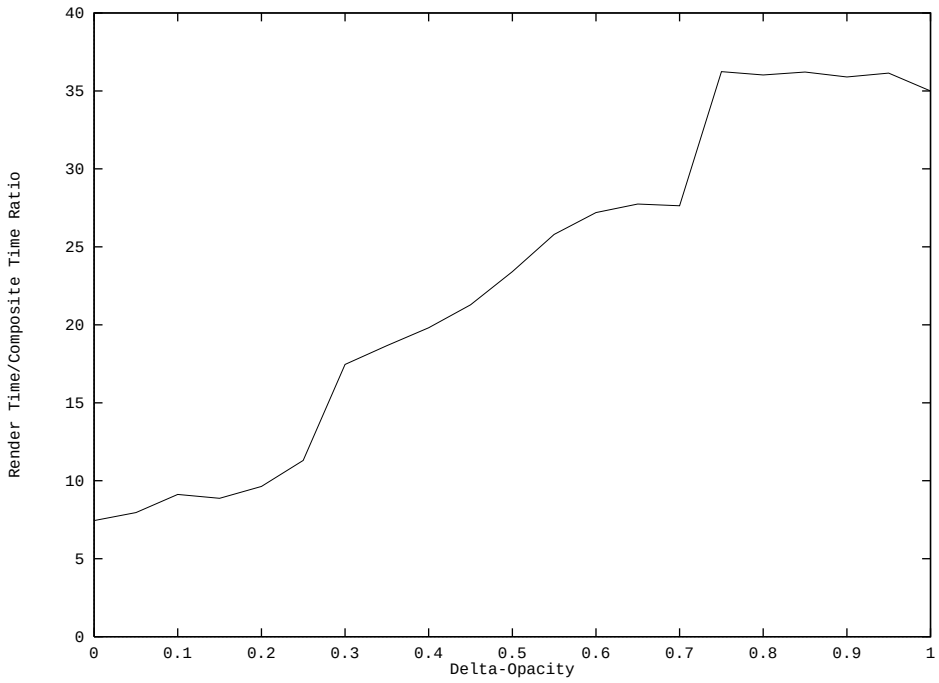


Figure 32. Plot showing effect of various opacity approximation thresholds (δ-Opacity) against ratio of Render/Composite time. The best Render/Composite ratios are at low approximation thresholds. The relationship is roughly linear ranging from 7 times to 35 times faster compositing.