10

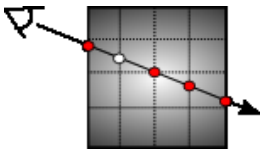
Direct Volume Rendering

For the visualization of three-dimensional scalar fields, direct volume rendering has emerged as a leading, and often preferred, method. In rendering volumetric data directly, a participating medium is composed of semi-transparent material that can emit, transmit, and absorb light, thereby allowing one to "see through" (or see inside) the data. By changing the optical properties of the material, different lighting effects can be achieved. In fact, isosurfacing can be considered a subset of direct volume rendering because the same effect can be achieved by specifying a small region around the isosurface value as opaque while everything else remains transparent.

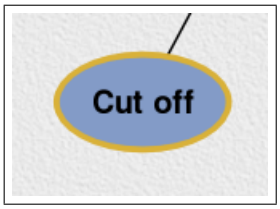
Direct volume rendering consists of three major components: sampling, classifying, and compositing. To compute an image, the effects of the optical properties must be continuously integrated throughout the volume. However, since the volume is represented by discrete cells, or voxels, this needs to be done in a piecewise manner. *Sampling* deals with selecting the piecewise steps that are taken through the volume, *classification* is the process of computing a color and opacity for each step, and *compositing* is the how these classified steps are blended together to form an image.

Consider a ray that originates from the eye and travels through the volume, sampling it in intervals. The space in each cell that the ray passes through represents a different color and opacity that contribute to the final image. The color and opacity for a single cell boundary intersection can be classified using a *transfer function*, which is just a map from the interpolated scalar values to colors and opacities. Performing this lookup before the rendering begins is called *pre-classification* and performing it at the time of intersection is called *post-classification*. The space inside a cell is integrated using the chosen optical model with the colors and opacities at the entry and exit intersections. This cell contribution is blended into a final image using alpha compositing and this process is repeated until the ray passes through the back of the volume.

Opacity is the inverse of transparency (also called alpha).



In this chapter, we describe this volume rendering process in more detail. In § 10.1, we describe the different optical models for classifying segements of the volume and the compositing techniques that are used to blend these segments together. In § 10.2, we cover some of the most common sampling algorithms that are used for volume rendering structured grids and in § 10.3 we cover unstructured grid techniques. Finally, in § 10.4 we describe transfer functions in more



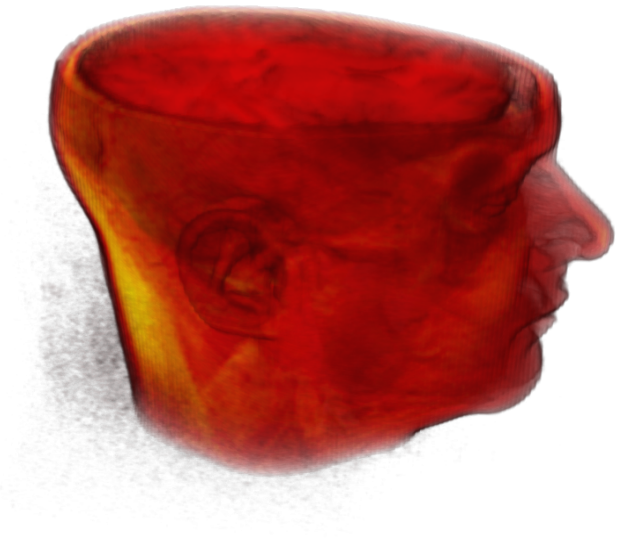


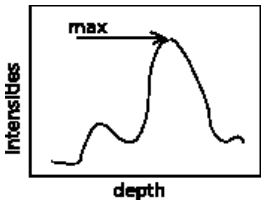
Figure 10.1: Direct Volume Rendering of an MRI scan of one of the authors' head. A cutting plane culls away the top of the head for a better view of the brain.

detail and how they can be used to create user-defined classifications that pull out features within the volume.

10.1 Optical Models

Direct volume rendering can use different optical models, depending on the participating media or the structures that need to be visualized. In this section, we summarize some of the most common techniques used for classifying steps within the volume by determining the color and opacity and we describe how these steps can be blended together using compositing to form a final image.

10.1.1 Maximum Intensity Projection



The simplest model for direct volume rendering is a maximum intensity projection (MIP) [Wallis and Miller 90, Wallis and Miller 91]. The basic idea is that if you project each voxel in the volume to the image plane, the MIP would be the maximum intensity that was projected for each pixel. In other words, the

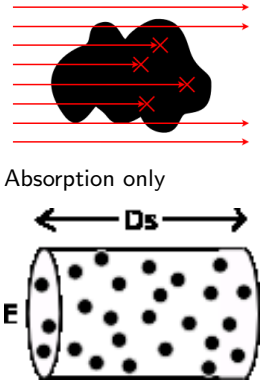
compositing operator for the opacities of each set of voxels is simply:

$$\alpha_i = max(\alpha_i, \alpha_{i-1}). \tag{10.1}$$

Because the composition is commutative, the voxels can be combined in any order.

Due to its simplicity, MIP rendering can be performed very quickly. Though not as powerful as other methods, it is commonly used for detecting high intensity structures (*i.e.,* vascular structures) or low intensity structures (*i.e.,* airways) within volumes from medical scanners.

10.1.2 Absorption



The most basic participating medium absorbs light going through it without emitting or scattering [Blinn 82, Max 95, Kajiya and Herzen 84]. An example of this type of medium is a cloud of black smoke.

Consider a cylindrical region of the volume with base B of area E and thickness Δs that contains ρ particles per unit that have a projected area A on B. The volume of the cylinder can be expressed as $E\Delta s$, from which the area occluded on the base can be computed: $\rho AE\Delta s$. This leads to a ratio of occluded area $\rho AE\Delta s/E$, or just $\rho A\Delta s$, from which the intensity of light I can be expressed, using the following differential equation:

$$\frac{dI}{ds} = -\rho(s)AI(s) \tag{10.2}$$

where $\rho(s)A$ is often referred to as the extinction coefficient and denoted as $\tau(s)$. The solution to this differential equation is

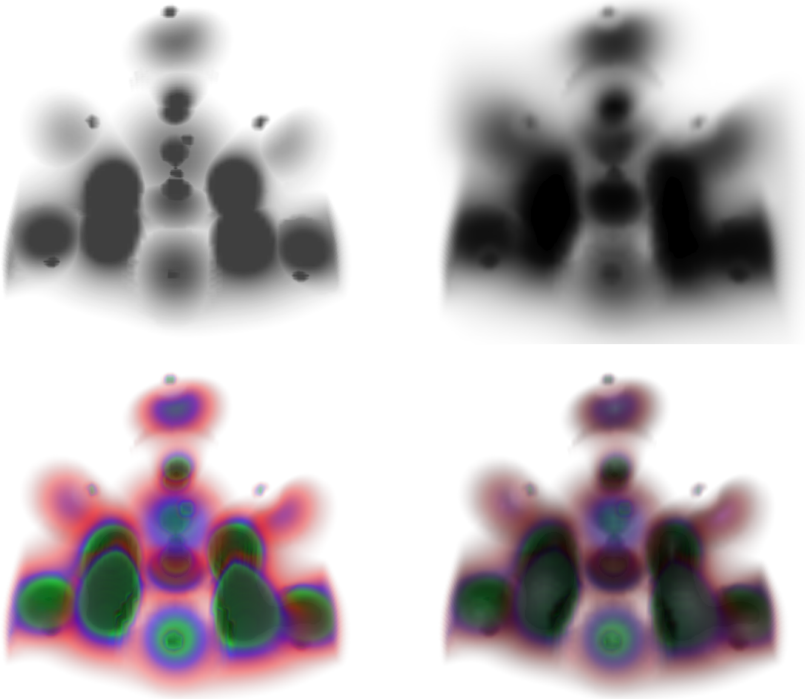
$$I(s) = I_0 e^{-\int_0^s \tau(t)dt} \tag{10.3}$$

which represents the amount of light absorbed between 0 and s. Compositing these regions together then just becomes an addition of the opacities:

$$\alpha_i \quad = \quad \alpha_i + \alpha_{i-1} \tag{10.4}$$

As with MIP, the compositing equation is commutative, and can thus occur in any order.

The absorption model is often used to show distributions of density in the volume. One example of this is X-ray imagery, which uses a photographic film to accumulate X-rays that pass through the body. Regions with dense tissues (*i.e.,* bone) block more X-rays than regions with soft tissue, which allow X-rays through and turn the film black.



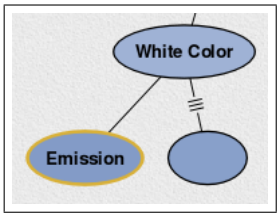
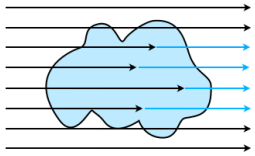


Figure 10.2: Optical models for direct volume rendering on a volume showing protein potentials. The models displayed are maximum intensity projection (top left), absorption only (top right), absorption and emission (bottom left), and absorption and emission with shading (bottom right).

10.1.3 Absorption and Emission

While absorption reduces the amount of light that exits the medium, emission increases it [Blinn 82, Max 95, Kajiya and Herzen 84, Sabella 88]. The effect is due to physical processes that convert energy to light, such as hot soot particles in a flame. Ignoring absorption, the differential equation can be expressed in terms of the intensity of the glow per unit projected area C:



Emission only

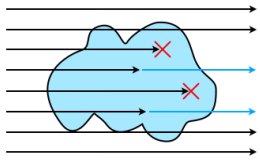
$$\frac{dI}{ds} = C(s)\rho(s)A, \tag{10.5}$$

which has the solution

$$I(s) = I_o + \int_0^s C(s)\rho(s)A. \tag{10.6}$$

Emission without absorption may produce unrealistic imagery especially with

high intensities. Thus it is common to incorporate absorption into the optical model to simulate the look of a real medium that both occludes light and adds to the light (such as a cloud). This optical model is generally used for direct volume rendering because it results in a high level of realism, but remains tractable because it does not involve mutliple scattering.



Absorption and emission

The differential equation for both terms becomes

$$\frac{dI}{ds} = C(s)\rho(s)A - \rho(s)AI(s) \tag{10.7}$$

which has the solution

$$I(D) = I_0 e^{-\int_0^D \rho(t)Adt} + \int_0^D C(s)\rho(s)Ae^{-\int_s^D \rho(t)Adt}ds \tag{10.8}$$

where $s = 0$ at the edge of the volume and $s = D$ at the eye. An approximation to this equation can be derived using a Reimann Sum. The results divides the integral into n equal segments of size Δx:

$$I(D) \approx I_0 \prod_{i=1}^{n} t_i + \sum_{i=1}^{n} g_i \prod_{j=i+1}^{n} t_j \tag{10.9}$$

where

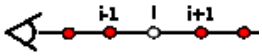
$$t_i = e^{-\rho(i\Delta x)A\Delta x}, \tag{10.10}$$

$$g_i = C(i\Delta x)\rho(i\Delta x)A. \tag{10.11}$$

Volume rendering with emission and absorption but without multiple scattering is also referred to as *low albedo* volume rendering. Albedo refers to the extent in which a surface reflects light.

This gives the back-to-front compositing algorithm that is commonly used in practice.

Alpha compositing takes a different form depending on the direction of the traversal [Porter and Duff 84]. For back-to-front compositing, the compositing equations are computed as a function of color and opacity:



$$\mathbf{c}_i = \mathbf{c}_i\alpha_i + \mathbf{c}_{i+1}(1-\alpha_i) \tag{10.12}$$

or similarly for front-to-back compositing:

$$\mathbf{c}_i = \mathbf{c}_{i-1} + \mathbf{c}_i\alpha_i(1-\alpha_{i-1}) \tag{10.13}$$

$$\alpha_i = \alpha_{i-1} + \alpha_i(1-\alpha_{i-1}) \tag{10.14}$$

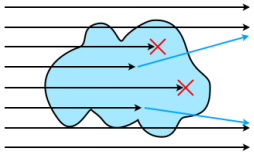
for the steps before $(i-1)$ or after $(i+1)$ the current step (i), RGB colors $(\mathbf{c})$, and transparencies (α). The resulting color is considered *pre-multiplied* by alpha. Compositing in this form requires a strict ordering, such as the one imposed by marching a ray through the volume or imposed by sorting voxels, because the equations are not commutative.

If the emissive term C is assumed to be constant along the ray, the differential equation can be simplified to:

$$I(D) = I_0 e^{-\int_0^D \rho(s)Ads} + \mathbf{c}(1 - e^{-\int_0^D \rho(s)Ads}) \tag{10.15}$$

which is simply a compositing of color $\mathbf{c}$ using an exponential extinction term.

10.1.4 Multiple Scattering



Multiple scattering

Light entering a participating medium may collide with particles and thus change direction. This scattering can be expressed as a function of the radiance distribution for measured particles, called a phase function [Glassner 95, Pharr and Humphreys 04, Bohren and Huffman 83, Denman et al. 66, McCartney 76]. Multiple scattering with phase functions is generally used to model atmospheric effects and full light simulations. However, it is less common for interactive volume rendering for visualization purposes due to the complications that reflection may cause computationally and because it offers limited benefit for exploring and understanding the data.

Phase functions are generally defined in terms of $a = cos(\theta)$, where θ is the angle between the original ray direction and the scattered direction. The phase function that expresses scattering in a medium is usually determined by characteristics of the particles in the medium. A ratio between particle size to wavelength of light can be used as a rule of thumb for selecting a phase function. If the radius r of the particles is much less than the wavelength of light λ, atmospheric absorption is used. If r is slightly less than λ, Rayleigh scattering is more appropriate. For r about the same size as λ, Mie scattering is used. Finally, for r much greater than λ, geometric optics are required.

Rayleigh scattering is generally used to simulate scattering when the particles are smaller than the wavelength of light. Some examples include water molecules in the atmosphere, smoke, or dust. The Rayleigh scattering function (R) is given by

$$P_R(a) = \frac{3}{4}(1+a)^2. \tag{10.16}$$

When the particle sizes are comparable to the wavelength of light, Mie scattering is often used. Some examples of this type of particles are water droplets or fog. The two most common Mie scattering functions are for haziness (HM) or murkiness (MM) and can be expressed as

$$P_{HM}(a) = 1+9\left(\frac{1+a}{2}\right)^8, \tag{10.17}$$

$$P_{MM}(a) = 1+50\left(\frac{1+a}{2}\right)^{32}. \tag{10.18}$$

An approximation to the Mie functions is the Heyney-Greenstein (HG) phase function:

$$P_{HG}(a) = \frac{1-g^2}{(1+g^2-2ga)^{\frac{3}{2}}} \tag{10.19}$$

where g is the assymetry parameter. Schlick's approximation (S) to HG is widely used in computer graphics because it avoids fractional exponentiation:

$$P_S(a) = \frac{1-g^2}{(1-ga)^2}. \tag{10.20}$$

Determining the appropriate phase function for scattering effects is thus a tradeoff between speed and quality.

10.1.5 Shading and Shadows

Volume absorption and emission are optical properties that are independent of the lighting source, and thus, do not give a good sense of depth within the volume. By incorporating shading into the rendering, this depth can be more perceptable. Intuitively, shading a transparent volume is similar to shading a single surfaces—except that instead there are an infinite number of surfaces. A multiple scattering technique, such as the Heyney-Greenstein function, can be used to attenuate the light to achieve this effect. A simpler method is to uses a method derived from Lambert's law for diffuse particles that are much larger than the light wavelength [Blinn 82]:

$$P_L(a) \quad = \quad (8/3\pi)(\sin\theta + (\pi-\theta)\cos\theta) \tag{10.21}$$

where θ in this case is the angle between the incoming ray r and the normal vector $\mathbf{n}$.

For a volume, computing the normal vector $\mathbf{n}$ is not as straightforward as it is for a surface. For any point within the volume, the normal is just the gradient of the scalar field f at that point, *i.e.,* $\mathbf{n} = \nabla f \| \nabla f \|$. For discrete samples, this can easily be computed using central differencing on a lattice defined by the neighbors of the samples [Höhne and Bernstein 86].

More advanced techniques such as shadows can also be used to give realistic imagery and improve depth cues. The most general form of shadows is performed in two passes [Kajiya and Herzen 84]. The first renders the volume from the light source using the absorption model (equation 10.3) and stores the light contribution for each point in space. The second pass then renders from the view point as normal using absorption and emmission (equation10.8) with shading, but using the results from the first pass as a multiplication factor at each step. This brightens the areas closer to the light and darkens those farther away.

10.1.6 Acceleration Techniques

The integration at each step in the volume rendering can be expensive. To mitigate this cost, methods have been developed to approximate the integral and move much of the computation to a pre-process [Engel et al. 01, Roettger and Ertl 02, Weiler et al. 03, Guthe et al. 02]. This method is called *pre-integration*, and results in substantial performance improvements. The basic idea is to compute a 3D table

of colors and opacities for a discrete number of front scalar values (x-axis), back scalar values (y-axis) and distances between them (z-axis). During rendering, this table is stored in a texture that can used for lookups at each step in the volume.

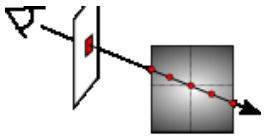
The main drawback to pre-integration as described is that it can be costly to compute the lookup table each time the mapping from scalar to color and opacity (transfer function) changes. To solve this problem, partial pre-integration can be used to store the static, computationally expensive components of the integration (*i.e.,* the exponentiation of distances) separately from the dynamic components (*i.e.,* color and opacity) [Moreland and Angel 04]. This allows interactive changes to the optical properties of the volume with virtually no overhead.

10.2 Structured Grids

Many techniques have been developed to volume render structured grids. These generally fall into three classifications: image-space, object-space, and hybrid techniques. Object-space algorithms are considered to be a *forward mapping* because they start with the volume and map it onto the image plane. Conversely, image-space algorithms are considered to be a *backward mapping* because they cast rays from the image plane into the volume. A complete coverage of all the techniques that have been is not in the scope of this book. Instead, we focus on the four most popular techniques and summarize some of the acceleration structures that have been used to improve their efficiency.

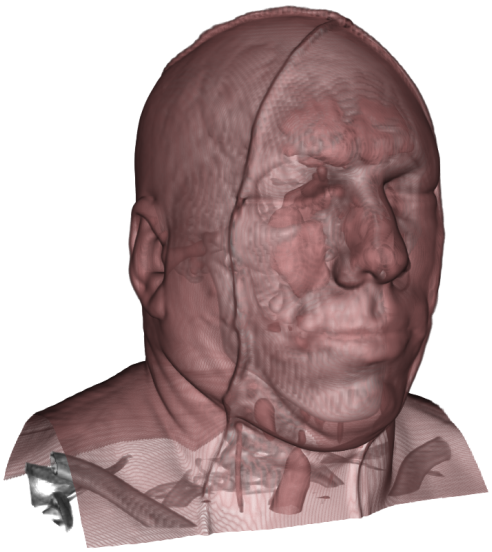
10.2.1 Image-Space Techniques

Ray-casting is perhaps the most conceptually simple sampling technique for volume rendering because it occurs completely in image-space [Drebin et al. 88, Upson and Keeler 88]. Ray-casting is similar to ray-tracing, but has no need for intersection testing, reflection, or refraction. The idea is to cast a ray, originating from the eye, through a pixel in the image plane and into the volume. As the ray passes through the volume, it samples the data incrementally to find the pixel's contribution to the final image.



At discrete steps within the volume, the sample of a point within the volume can be determined using trilinear interpolation within the voxel that contains it. For pre-classified data, this means interpolating between the colors and opacities of the voxel corners and for post-classified data it is an interpolating of the scalars before a transfer function lookup. An obvious choice for sample locations are the voxel boundaries. However, depending on the view, this may result in artifacts due to insufficient samples. Therefore, an intermediate step is generally used within a voxel, which satisfies the constraints given by sampling theory.

Once a sample is obtained, the integral can be computed using the current sample, previous sample, and distance between them. After the integral is computed for the step, lighting can be applied using the computed gradient. Finally,



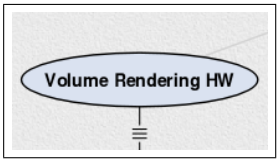


Figure 10.3: Direct volume rendering for structured grids. The rendering emphasizes the skin and bone inside the volume obtained from a CT scan.

the results are composited with the previous steps. This process is repeated for each pixel in the image plane.

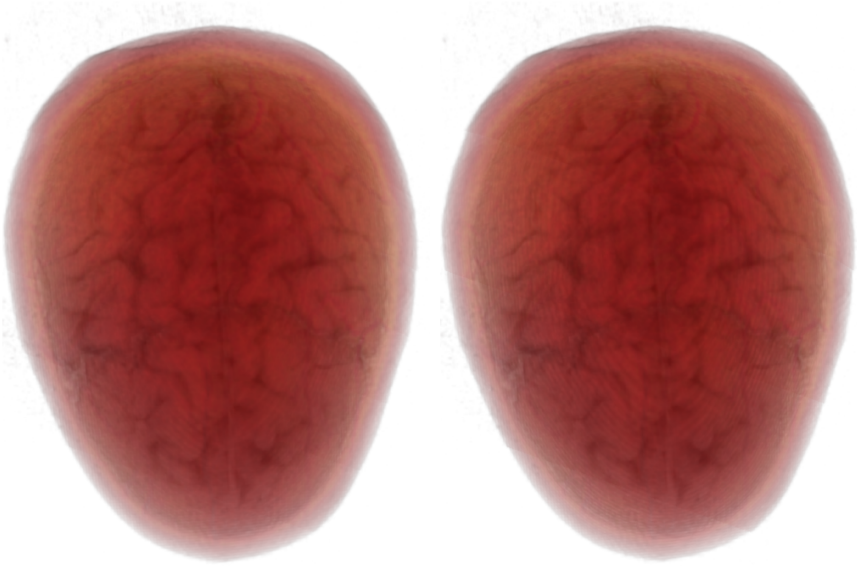
Figure 10.4 shows some examples of ray-casting for a portion of an MRI scan of a head. The Figure shows the difference between sampling at one step per voxel and sampling at two steps per voxel.

The advantage of ray-casting over other approaches is that it is easily implemented in parallel, it can be readily extended to a full ray-tracing for advanced effects, and it does not require any specialized hardware. For these reasons, ray-casting is still a common method for volume visualization even though it may be much slower techniques that take advantage of graphics hardware.

10.2.2 Object-Space Techniques

There are two common techniques for volume rendering unstructured grids in object-space: splatting and texture slicing. Both these methods project the volume in some form onto the image plane to get a rendering.

Splatting. *Splatting* is a method for volume rendering that operates on the voxels witin the volume [Westover 89,Westover 90]. Each scalar value within the volume is represented by a semi-transperent *kernel*, or footprint, that is projected onto the image plane and composited with other splats to create a final image. Depending



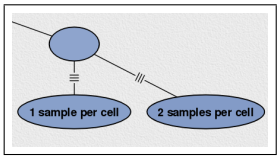
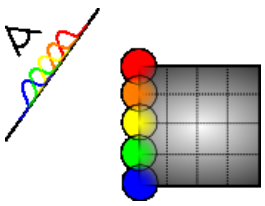


Figure 10.4: Ray-casting of an MRI of the top of a head using two samples per voxel (left), one sample per voxel (right). The right image suffers from ringing artifacts in the center and stair-casing artifacts near the bottom that are a consequence from undersampling.

on the optical model, these kernels need to be splatted in front-to-back or back-to-front order for correct compositing. The issues with splatting involve kernel selection, representation, and ordering.

For a grid, the choice of footprints is generally a circle, which is invariant to rotations and thus can be used for all view directions. To avoid aliasing, the opacity of the splat needs to be convolved with a smoothing kernel, typically a Gaussian function. The size of the splat is chosen to avoid holes in the volume. This requires a small overlap between neighboring splats. If this overlap is too large, the projections will be blurred. A diameter of 1.6 times the space between values has been found to be sufficient to avoid aliasing from splats while still preventing blurring [Crawfis and Max 93]. However, for perspective projections, this size needs to be adjusted according to the samples' distance from the viewpoint.



There are several ways to represent a footprint for splatting. The most efficient is to use a screen-aligned quadrilateral to represent each footprint [Crawfis and Max 93]. The quadrilateral is texture mapped using an opacity weighted texture that represents the classification of the scalar and rendered directly using graphics hardware.

To ensure that the splats are rendered in the correct order, an axis aligned *sheet plane* is used [Westover 90]. The idea is to choose the axis that is most

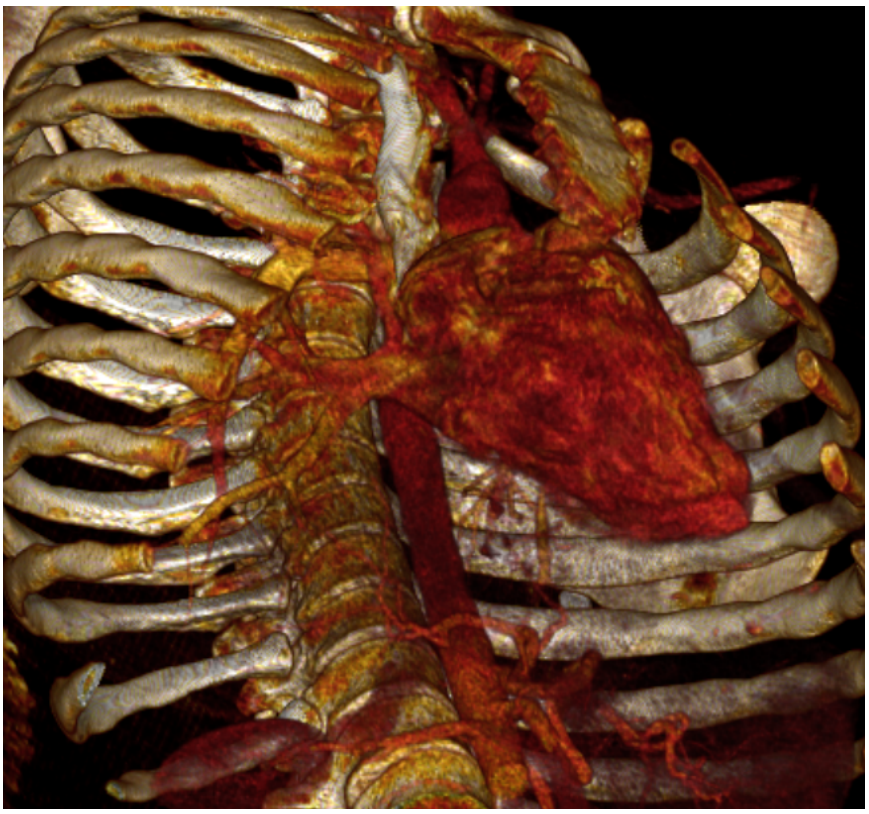
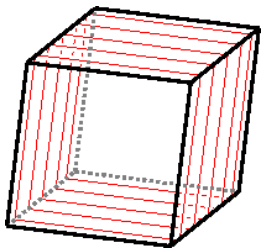


Figure 10.5: Direct volume rendering of a CT of a torso using texture slices.

aligned to the current viewpoint and render the data in layers along that axis. This traversal can be performed efficiently because of the regular access patterns of the data. However, it may result in artifacts while changing viewpoints. These include incorrect volume integration due to varying distances between splats and popping when switching between axes. To mitigate this problem, the sheet planes can be aligned to the image plane instead of an axis [Mueller and Crawfis 98]. This results is a less efficient but more correct rendering.

Texture Slicing. An object-space method that has become the most common due to modern graphics processors is *texture slicing* [Cullip and Neumann 93, Cabral et al. 94, Guan and Lipes 94, Wilson et al. 94]. This method maps the volume to 2D or 3D textures in graphics hardware. Parallel planes, or *proxy geometry*, are then used to slice the volume into discrete steps and sample the textures. These slices are finally rendered in front-to-back or back-to-front order and composited into a the resulting image.



Axis aligned texture slices

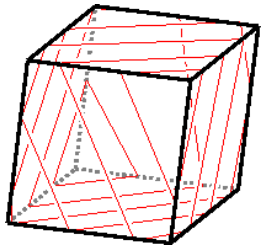
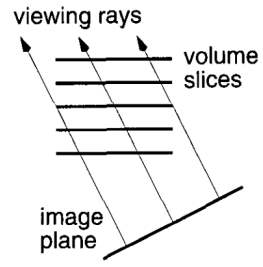


Image-plane aligned texture slices



Standard ray-casting

Early graphics boards only supported 2D textures in hardware, which makes volume rendering only possible by representing volumetric data as stacks of 2D images. The proxy geometry to sample these textures can then be performed using axis-aligned quadrilaterals. To improve efficiency, the data can be replicated in the three axis directions, which increases memory usage but allows the geoemtry for the axis most aligned with the viewpoint to be quickly constructed. Avoiding this extra storage requires on-the-fly slice reconstruction at the expense of performance overhead [Lefohn et al. 04]. Another problem that arises here is the dependency on the sampling rate used to produce the regular grid, which can be avoided by using extra slices and trilinear filtering between textures [Rezk-Salama et al. 00].

The advent of 3D texturing capabilities even further improved the ways that the graphics hardware can be used to perform volume rendering. Since the volume data is naturally stored as a 3D texture, it suffices to use proxy geometry consisting of a set of parallel polygons orthogonal to the viewing direction (aligned with the image-plane). The scalar values stored in 3D textures are processed in pairs of successive slices. For each pair of scalar values, the color and opacity contribution can be computed using the volume rendering integral. These results are then composited into the final image.

The advantage of object-space approaches are that they are fast. They represent the volume in ways that are efficient for modern graphics processors. Because of this, they are often the choice when interactivity is important. However, performing more advanced effects, such as shadows, may not be as easily achieved as with a ray-caster.

Figure 10.2.2 shows an example of a volume rendering using texture slices. The volume is ??x??x?? yet can still rendered interactively.

10.2.3 Hybrid Techniques

A popular method for structured grids that operates in both object- and image-space is *shear-warp* volume rendering [Cameron and Undrill 92, Yagel and Kaufman 92, Schröder and Stoll 92, Lacroute and Levoy 94]. Shear-warp is a highly optimized ray-casting approach that gets its speedup from a clever object-space shearing and warping of the volume.

The sampling step of a ray-caster is most efficient when the viewpoint is aligned with an axis orthogonal to the grid of the volume. This is because number of sampling steps required is less than at other viewpoints, the sample locations are easily determined, and the distances between samples is uniform. The idea behind shear-warp volume rendering is to use only send rays through the volume at axis aligned slices, but to shear and warp the volume depending on the viewpoint.

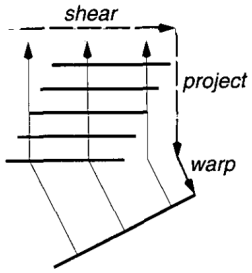
The view transformation V at any position can be factorized as follows:

$$V = P \cdot S \cdot W \tag{10.22}$$

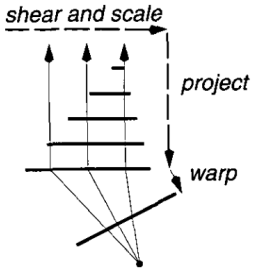
where P is a permuation matrix which transposes the coordinate system to the

principle axis, S is a shearing matrix, and W is a matrix that transforms the sheard object into image coordinates. For a perspective projection, an extra scaling of S is required. The volume is traversed in slice order, but the slices are warped to project correctly on the image plane. As a ray steps from slice to slice, the shearing ensures that the same warp can be used.

As with the other object-space methods that use an axis-aligned traversal, the data access for the three axes can be pre-computed and stored for efficiency. In addition, the computation can be preformed in parallel based on slices, instead of rays. The result is a substantial speedup over basic ray-casting. The disadvantage is that artifacts may occur from the sampling as the viewpoint approaches a 45° angle from any axis.



Shear-warp ray-casting for an orthogonal projection



Shear-warp ray-casting for a perspective projection

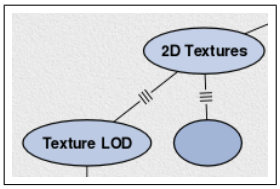
10.2.4 Acceleration Techniques

The majority of work in the area of volume rendering has been to improve upon these volume rendering approaches. Many of these are targeted at improving performance.

One obvious acceleration method is to avoid unnecessary computation within the volume. The two methods for doing this are early ray termination and empty space skipping. *Early ray termination* recognizes that if the data is being composited from front-to-back, and the resulting image is already opaque, there is no need to continue compositing. In the case of ray-casting, the ray can be terminated before it passes through the volume if the pixel it corresponds to is opaque [Levoy 90a]. A similar, though slightly more complex, test can be performed for splatting algorithms as well to end the splatting early if the region of the splat is opaque [Mueller et al. 99]. Generally a threshold like 95% opacity (5% transparency) is used because additional contributions are not noticeable.

Another method for avoiding unnecessary computation is *empty-space skipping* [Levoy 90a]. This acceleration technique is targeted towards avoiding the regions in the volume that are completely transparent. This can be done by computing a hierarchical representation of the volume that is more coarse in homogenous regions. The coarse regions can then be traversed with larger steps than the rest of the volume. A similar effect can be achieved by using distance field around objects of interest that adapt the step size [Zuiderveld et al. 92]. Isosurfacing can also be used as a pre-computation to determine the first hit location of the important regions in the mesh [Avila et al. 92] Similarly, methods have been established that send out scattered rays or use frame-to-frame coherence to find first hit locations before the full rendering pass occurs.

Interactivity can also be improved substantially using *level-of-detail* (LOD) methods. These methods change the resolution of the rendering during viewpoint interaction to maintain responsive frame rates. The most common LOD strategy is to use a coarser sampling of the volume. For ray-casting, this could mean using larger step sizes or using less rays [Levoy 90b]. For object-space techniques, fewer slices through the data can be used. A hierarchical representation of the



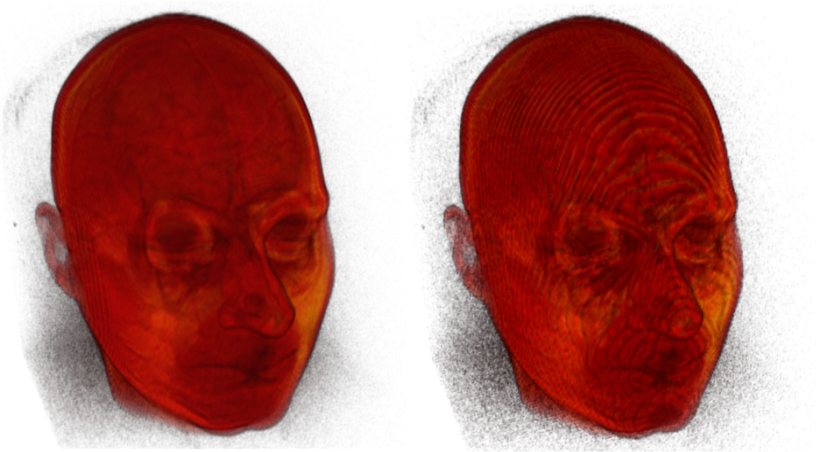


Figure 10.6: Direct volume rendering of an MRI of a head using texture slicing with 2D textures. The left image shows a full quality rendering using correct sampling and the right image shows a level-of-detail rendering using every fourth slice.

data, such as an octree, could also be used [Danskin and Hanrahan 92] that can be easily traversed for higher or lower resolutions of the data. The result is similar for both techniques. Figure 10.6 shows an example of texture slicing with 2D texture at full quality and with a reduced number of slices.

Probably the most substantial improvement to interactivity in volume rendering of unstructured grids has come with advances in graphics hardware. Graphics processing unit (GPU) speeds have been increasing at a rate faster than Moore's Law. They are very efficient for rendering large scenes of polygons. With the increasing amount of programmability, they have also become efficient for other types of parallel computation. Ray-casting techniques have been developed [Roettger et al. 02] and texture-based approaches have been improved substantially [Krüger and Westermann 03] due this programmability. This trend is likely to continue and interactivity will continue to increase for larger and larger volumes.