Frequency Domain Volume Rendering

Takashi Totsuka[*] *Marc Levoy*[†]

[*]SONY Corporation
[†]Computer Science Department, Stanford University

Abstract

The Fourier projection-slice theorem allows projections of volume data to be generated in $O(n^2 \log n)$ time for a volume of size n^3. The method operates by extracting and inverse Fourier transforming 2D slices from a 3D frequency domain representation of the volume. Unfortunately, these projections do not exhibit the occlusion that is characteristic of conventional volume renderings. We present a new frequency domain volume rendering algorithm that replaces much of the missing depth and shape cues by performing shading calculations in the frequency domain during slice extraction. In particular, we demonstrate frequency domain methods for computing linear or nonlinear depth cueing and directional diffuse reflection. The resulting images can be generated an order of magnitude faster than volume renderings and may be more useful for many applications.

CR Categories: I.3.7 [Computer Graphics]: Three-dimensional Graphics and Realism.; I.3.3 [Computer Graphics]: Picture/Image Generation; Display Algorithms.

Additional Keywords: Volume rendering, Fourier transform, Shading models, Scientific visualization, Medical imaging, Digital signal processing.

1 Introduction

Volume rendering is an important tool for visualizing 3D scalar fields. Most existing algorithms operate in the spatial domain. They can be classified as either image space algorithms (e.g. [7]) or object space algorithms (e.g. [4], [15]) depending on the order in which the data is traversed: along each ray cast from the image plane or along X, Y, and Z axis of the volume data. The complexity of these algorithms is $O(n^3)$ since all voxels must be visited to render an image. This high cost limits the use of these algorithms in interactive environments. Although efficient algorithms exist for sparse data sets [8], [14],[16], such optimization is data dependent.

In an effort to drastically reduce rendering costs, frequency domain algorithms based on the Fourier projection slice theorem have been proposed [5], [10]. It is well known that the integral of a 1D signal is equal to the value of its spectrum at the origin. The Fourier projection slice theorem extends this notion to higher dimensions.

* Sony Corporation. 6-7-35 Kitashinagawa, Shinagawa Tokyo 141, Japan (totsuka@av.crl.sony.co.jp)
† Center for Integrated Systems, Stanford University Stanford, CA 94305 (levoy@cs.stanford.edu)

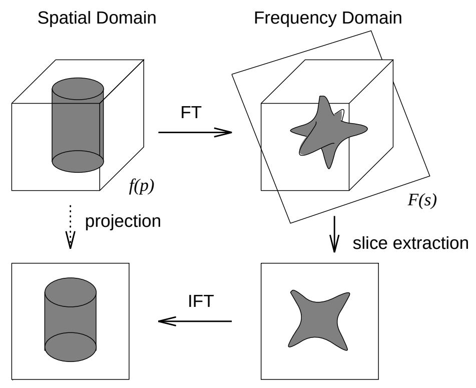

Figure 1: Volume rendering using Fourier projection slice theorem

For a 3D volume, the theorem states that the following two are a Fourier transform pair:

- The 2D image obtained by taking line integrals of the volume along rays perpendicular to the image plane.
- The 2D spectrum obtained by extracting a slice from the Fourier transform of the volume along a plane which includes the origin and is parallel to the image plane.

Using this theorem, once a volume data is Fourier transformed, an (orthographic) image for any viewing direction can be obtained by extracting a 2D slice of the 3D spectrum at the appropriate orientation and then inverse Fourier transforming it (figure 1). The cost of this approach is dominated by the 2D inverse fast Fourier transform (IFFT) which is $O(n^2 \log n)$. Hence, the overall cost is also $O(n^2 \log n)$. Since $\log n$ grows slowly, the advantage of this approach over spatial domain algorithms is greater at large data sizes.

Despite their theoretical speed advantage, frequency domain volume rendering algorithms suffer from several well-known problems:

High interpolation cost: Because the sample points of the 3D spectrum and those of the 2D slice do not coincide except at the origin, the 3D spectrum must be interpolated and then resampled in order to extract a 2D slice. Since this interpolation is imperfect, replicas of the volume data are not fully suppressed, causing ghosts to appear on the projection image. Because any filter that provides a sharp cutoff in the spatial domain also has wide support, high-quality interpolation is expensive. As the interpolation is $O(n^2)$, the FFT is still asymptotically dominant. However, due to a large constant factor associated with the interpolation, current implementations spend the majority of their running time in interpolation, making the algorithm not attractive for practical data sizes (128^3 or 256^3).

Memory cost: Due to the wide dynamic range and complex arithmetic associated with Fourier transforms, a pair of floating point numbers is required for each voxel. Assuming a 64-bit double precision representation, 16 bytes are required per voxel. By contrast, only 1 byte per voxel is necessary in spatial domain algorithms.

Lack of depth information: The projection obtained by the Fourier projection slice theorem is a line integral normal to the direction of view. Voxels on a viewing ray contribute equally to the image regardless of their distance from the eye. The image therefore lacks occlusion, an important visual cue. While some users (diagnostic radiologists in particular) prefer integral projections since nothing is hidden from view, this characteristic would be considered a drawback in most applications.

The first two problems listed above are technical in nature, and several promising solutions are proposed later in this paper. The lack of occlusion is fundamental, however, in so far as no projection-slice theorem is known that mimics the integro-differential equation ([6]) approximated by volume rendering algorithms. Fortunately, occlusion is only one of many cues employed by the human visual system to determine the shape and spatial relationships of objects. Other available cues include perspective, shading, texture, shadows, atmospheric attenuation, stereopsis, ocular accommodation, head motion parallax, and the kinetic depth effect.

It is possible, of course, to apply any shading technique in the spatial domain before the volume is Fourier transformed. However, such a naive approach would require recomputation of the volume followed by an expensive 3D forward FFT each time the view or the lighting condition is changed.

In an earlier paper [9], we instead showed that for a limited class of shading models, the dependence on viewing direction and lighting direction could be factored out of the projection integral, yielding equations of the form

$$I = \sum_{i=0}^{n} w_i \left(\int_{-\infty}^{+\infty} f_i\big(x(t), y(t), z(t)\big)\, dt \right). \qquad (1)$$

Here, effects of viewing and lighting direction are solely expressed by weights w_i while the volumes f_i are independent of them. The indicated integration can be evaluated efficiently using the projection slice theorem. For example, linear depth cueing can be computed as the weighted sum of projections through three volumes that are depth cued before 3D forward FFT along X, Y, and Z directions, respectively.

The obvious disadvantage of this hybrid spatial-frequency domain approach is that it requires multiple copies of the volume. While still asymptotically faster than conventional spatial domain volume rendering, implementation considerations (problems one and two above) make it barely superior in practice.

In the present paper, we describe methods for rendering volumes with depth cueing and directional shading that operate entirely within the frequency domain. They are based on two well-known properties of the Fourier transform.

- Multiplication by a linear ramp in the spatial domain is equivalent to differentiation in the Fourier domain.
- Differentiation in the spatial domain is equivalent to multiplication by a linear ramp in the Fourier domain.

Using these properties, depth cueing implemented in [9] as spatial domain multiplication, is implemented in the present paper using frequency domain differentiation. Similarly, directional shading, implemented in [9] using spatial domain differentiation, is implemented in the present paper using frequency domain multiplication.

The remainder of the paper is organized as follows. Section 2 reviews the previous works. Section 3 presents our new frequency domain shape cueing techniques. Sections 4 and 5 refer to solutions to the interpolation and the memory cost problems, respectively. Section 6 shows results from our implementation, and section 7 gives conclusions and possible future directions.

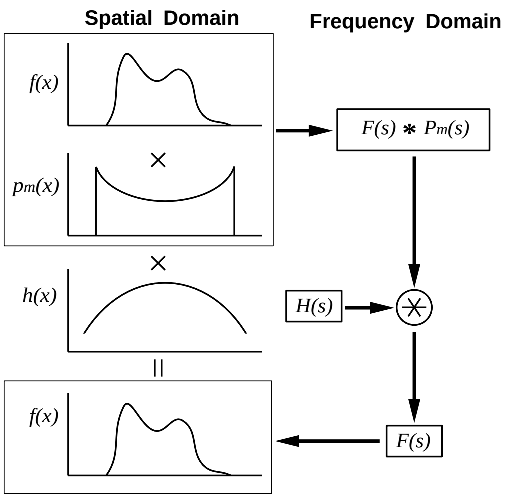

Figure 2: Premultiplication of the volume data

2 Base Algorithm

We begin by briefly reviewing current frequency domain volume rendering algorithms. In the following discussion, small letters ($f, g, \ldots$) represent data in the spatial domain and capital letters ($F, G, \ldots$) represent data in the frequency domain. We also assume that the transform between the two domains is the Fourier transform which is denoted by $\mathcal{F}$.

Let $f(\boldsymbol{x})$ be a volume and $F(\boldsymbol{s})$ be its Fourier transform. $\boldsymbol{x}$ and $\boldsymbol{s}$ are 3D vectors in the spatial and frequency domain, respectively. Given $f(\boldsymbol{x})$, the algorithm first transforms it into the frequency domain to yield $F(\boldsymbol{s})$. This is done only once. For each view, the discrete spectrum $F(\boldsymbol{s})$ is interpolated along the extraction plane (parallel to the image plane and passing through the origin) using a filter $H(\boldsymbol{s})$. The interpolated spectrum is resampled to obtain a 2D spectrum which is then inverse transformed to obtain a spatial domain projection.

By the convolution theorem, interpolation $F(\boldsymbol{s}) * H(\boldsymbol{s})$ corresponds to $f(\boldsymbol{x}) \cdot h(\boldsymbol{x})$ in the spatial domain. Here, $h(\boldsymbol{x})$ is the response of the filter. Unless $H(\boldsymbol{s})$ is an ideal lowpass filter, its response has a smooth shoulder. Thus, the periphery of the volume and consequently the periphery of the projected image is attenuated. To cope with this "vignetting" problem, the volume data $f(\boldsymbol{x})$ can be premultiplied by the reciprocal of the response, $p_m(\boldsymbol{x}) = \frac{1}{h(\boldsymbol{x})}$ before its forward transformation [10]. As H and P_m cancel during interpolation, we obtain a correct slice of F (figure 2). We have implemented this method using filters obtained from Malzbender and have obtained excellent results, as documented in section 4 and 6.

3 Shape Cueing Techniques

3.1 Depth Cueing

Depth cueing is obtained by weighting voxels according to their distance from the observer. Let $d(\boldsymbol{x})$ be the weighting function or depth cueing function for a given eye position. Then, a depth-cued volume is expressed as $f(\boldsymbol{x}) \cdot d(\boldsymbol{x})$. By transforming it to the frequency domain and extracting a slice, we obtain a depth cued projection. As stated earlier, this straightforward approach requires an expensive 3D FFT ($n^3 \log n$) for each view. There is, however, an elegant and inexpensive equivalent operation in frequency domain.

Including the compensation $p_m(\boldsymbol{x})$ for the filter response, spatial domain depth cueing can be expressed as $f(\boldsymbol{x}) \cdot d(\boldsymbol{x}) \cdot p_m(\boldsymbol{x})$. By transforming and interpolating, this corresponds to $\mathcal{F}\{f(\boldsymbol{x}) \cdot d(\boldsymbol{x}) \cdot p_m(\boldsymbol{x})\} * H(\boldsymbol{s})$ at sample points on the slice in the frequency domain. Using the convolution theorem, this expression can be rewritten as follows:

$$\begin{aligned} & \mathcal{F}\{f(\boldsymbol{x})d(\boldsymbol{x})p_m(\boldsymbol{x})\} * H(\boldsymbol{s}) \\ = \; & (F(\boldsymbol{s}) * D(\boldsymbol{s}) * P_m(\boldsymbol{s})) * H(\boldsymbol{s}) \\ = \; & (F(\boldsymbol{s}) * P_m(\boldsymbol{s})) * (H(\boldsymbol{s}) * D(\boldsymbol{s})) \\ = \; & \mathcal{F}\{f(\boldsymbol{x})p_m(\boldsymbol{x})\} * H'(\boldsymbol{s}) \end{aligned} \quad (2)$$

where $H'(\boldsymbol{s}) = H(\boldsymbol{s}) * D(\boldsymbol{s})$.

Thus, merely by replacing the interpolation filter H with H', we have obtained depth cueing. Note that the above expression operates entirely in the frequency domain, and moreover is evaluated only on the plane of the slice being extracted. Hence, it is a 2D operation. Note also that because $\mathcal{F}\{f(\boldsymbol{x}) \cdot p_m(\boldsymbol{x})\}$ is independent of the eye position, the 3D forward transform is performed only once.

Although H' must be computed for each view, the cost of recomputation is small because the support of filter H is small ($3^3 \sim 5^3$) and $D(\boldsymbol{s})$ is usually a simple expression. In practice, the recomputation is negligible compared with the cost of interpolation itself.

This frequency domain depth cueing method applies to any depth cueing function $d(\boldsymbol{x})$. Indeed, the method can be designed to highlight the middle portion of the volume while attenuating the front and back portions.

By way of example, we first consider simple linear depth cueing, $d_l(\boldsymbol{x})$. Let the view vector be $\boldsymbol{V}$. The signed depth measured from the origin of the volume is thus given by $(\boldsymbol{V} \cdot \boldsymbol{x})$, and $d_l(\boldsymbol{x})$ can be written as

$$d_l(\boldsymbol{x}) = C_{cue}(\boldsymbol{V} \cdot \boldsymbol{x}) + C_{avg} \quad (3)$$

where C_{cue} is the strength of the depth cueing effect and C_{avg} is a constant (see figure 3). Taking Fourier transforms, we obtain

$$D_l(\boldsymbol{s}) = -\frac{C_{cue}}{i2\pi}(\boldsymbol{V} \cdot \boldsymbol{\Delta}) + C_{avg}\,\delta(\boldsymbol{s}) \quad (4)$$

where $\boldsymbol{\Delta} = [\Delta_x, \Delta_y, \Delta_z]$ is the differential operator of convolution ($\Delta_x * f = \frac{\partial}{\partial x} f$). Substituting the interpolation filter with depth cueing (H') yields

$$\begin{aligned} H'(\boldsymbol{s}) &= H(\boldsymbol{s}) * D_l(\boldsymbol{s}) \\ &= -\frac{C_{cue}}{i2\pi}(\boldsymbol{V} \cdot \nabla H(\boldsymbol{s})) + C_{avg}\,H(\boldsymbol{s}) \end{aligned} \quad (5)$$

The first term exhibits the depth cueing effect. Since ∇H can be precomputed and stored in a table, computation of H' is of insignificant cost. An example of frequency domain linear depth cueing and projection is shown in figure 6(b). As a reference, the same volume rendered without depth cueing is shown in figure 6(a).

Although any function can be used for D, finding one that has a simple form reduces the cost of computing H'. The size of H' is also a consideration, since it directly impacts rendering time. To illustrate this important issue, let us employ a half period of

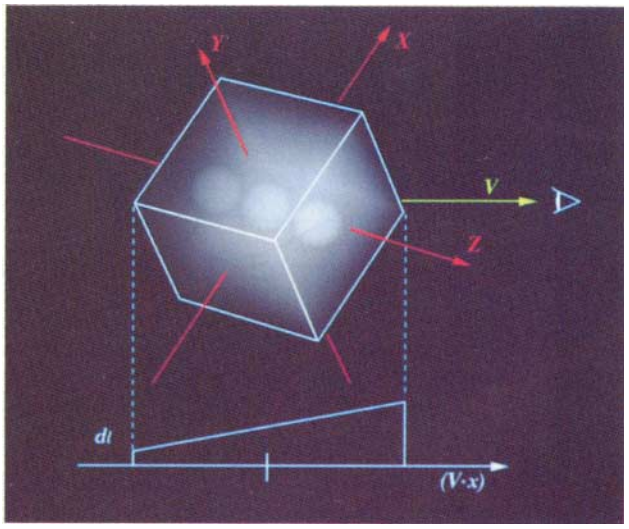

Figure 3: Linear depth cueing

a sine wave as $d(\boldsymbol{x})$. Since the transform of a sine function is two impulses, H' can be computed by shifting H and adding three copies[1] with complex weights. Note that this considerably increases the size of the filter kernel. By adjusting the origin, amplitude, and period such that the value is zero at the farthest voxel and unity at the closest voxel, we eliminate the need for a DC term. D now has the form $C_1\,\delta(\boldsymbol{s} - \boldsymbol{s}_w) + C_2\,\delta(\boldsymbol{s} + \boldsymbol{s}_w)$ where C_1 and C_2 are complex constants determined by the amplitude and the shift of the wave and $\boldsymbol{s}_w$ is determined by the period of the wave. The period is typically made long enough so that the depth cueing appears almost linear. We can further remove one of the impulses by doubling the weight of the remaining impulse. By removing one of the impulses, the projection image is no longer a real[2]. However, the real part of the result still contains the correct projection image. With this technique, depth cueing is implemented by an interpolation with a shifted H, which is practically free.

The notion of a shifted H gives us an alternative way to look at the process. Extracting a slice from a spectrum at a position translated from the origin by a distance d in a direction $\boldsymbol{V}$ corresponds to phase-shifting the spatial domain projection by $e^{i2\pi d t}$ at distance t in the same direction $\boldsymbol{V}$. The real part of such a phase-shifted projection appears to fade in and out as a function of position in direction $\boldsymbol{V}$ and, for appropriate values of d, the visual effect is that of depth cueing.

3.2 Directional Shading

In a scene composed of surfaces, directional shading using the well-known Lambertian reflection model is given by

$$C_{amb}\,O_c L_{amb} + C_{dif}\,O_c L_{dif}\,\mathrm{MAX}\big(0, (\boldsymbol{N} \cdot \boldsymbol{L})\big) \quad (6)$$

where C_{amb} and C_{dif} are constants defining the strength of ambient and directional shading terms, O_c is an object color, L_{amb} and L_{dif} are constants defining the color of ambient and directional lights, and $\boldsymbol{N}$ and $\boldsymbol{L}$ are unit surface normal and light vectors, respectively.

[1]Two for the impulses of the sine wave term and one for the constant term of $d(\boldsymbol{x})$.

[2]The imaginary part is a cosine wave since we are using the analytic signal of the depth cueing function. See the discussion on the Hilbert transform in [1].

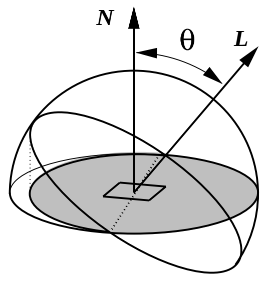

Figure 4: Hemispherical light source

Ignoring the attenuation of light inside the volume, the ambient term can be approximated using

$$C_{amb}\, L_{amb}\, f(\boldsymbol{x}) \tag{7}$$

The diffuse term, however, must be handled carefully because the nonlinear function MAX does not have a simple frequency domain representation. Note that the frequently used alternative, $|\boldsymbol{N}\cdot\boldsymbol{L}|$, which shades surfaces as if they are two-sided rather than the bounding surface of a solid, is also nonlinear and cannot be handled directly in the frequency domain.

To avoid this problem, we employ a hemispherical light source [12], [9]. The irradiance E_i on a surface having normal vector $\boldsymbol{N}$ illuminated by a hemisphere whose pole points in direction $\boldsymbol{L}$ as shown in figure 4 is proportional by Nusselt's analog (as described in [3]) to the projection of the visible portion of the hemisphere down onto the plane containing the surface, or

$$E_i = L_{dif}\, \frac{1}{2}(1 + \cos\theta) = L_{dif}\, \frac{1}{2}\big(1 + (\boldsymbol{N}\cdot\boldsymbol{L})\big) \tag{8}$$

With this shading model, the diffuse term in a surface model is expressed as

$$C_{dif}\, O_c L_{dif}\, \frac{1}{2}\big(1 + (\boldsymbol{N}\cdot\boldsymbol{L})\big) \tag{9}$$

For volumes, we have

$$\begin{aligned} & C_{dif} L_{dif}\, \frac{1}{2}\, |\nabla f(\boldsymbol{x})| \left(1 + \frac{(\nabla f(\boldsymbol{x})\cdot\boldsymbol{L})}{|\nabla f(\boldsymbol{x})|}\right) \\ = \;& C_{dif} L_{dif}\, \frac{1}{2}\big(|\nabla f(\boldsymbol{x})| + \big(\nabla f(\boldsymbol{x})\cdot\boldsymbol{L}\big)\big) \end{aligned} \tag{10}$$

Since volume datasets do not have explicitly defined surfaces, $\nabla f(\boldsymbol{x})$ is used as the normal vector at each location. The strength of directional shading in volume rendering algorithms is commonly made proportional to the gradient magnitude as a simulation of the *surface-ness* of the volume [4],[7]. Locales having high gradient magnitudes (i.e., steep jumps in density) reflect more light.

Equation (10) can be computed entirely in the frequency domain. By the derivative theorem, the gradient in one domain is the first moment in the other domain. Thus, the shading computation can be performed as a moment computation in the frequency domain. This useful property of linear shading can also be exploited in image understanding algorithms. For example, [13] uses the moment to estimate the orientation of surfaces assuming that the reflectance function is linear with respect to the slope of the surfaces.

Transforming equations (7) and (10) to the frequency domain and including compensation for the filter response, we obtain

$$\begin{aligned} & \mathcal{F}\Big\{C_{amb}\, L_{amb}\, f(\boldsymbol{x}) \\ & \quad + C_{dif} L_{dif}\, \frac{1}{2}\big(|\nabla f(\boldsymbol{x})| + \big(\nabla f(\boldsymbol{x})\cdot\boldsymbol{L}\big)\big)\Big\} \\ = \;& \big(C_{amb}\, L_{amb} + i\pi C_{dif}\, L_{dif}\, (\boldsymbol{s}\cdot\boldsymbol{L})\big) \\ & \quad \times\big(\mathcal{F}\{f(\boldsymbol{x})\, p_m(\boldsymbol{x})\} * H(\boldsymbol{s})\big) \\ & \quad + \frac{1}{2} C_{dif}\, L_{dif}\, \big(\mathcal{F}\{|\nabla f(\boldsymbol{x})|\, p_m(\boldsymbol{x})\} * H(\boldsymbol{s})\big) \end{aligned} \tag{11}$$

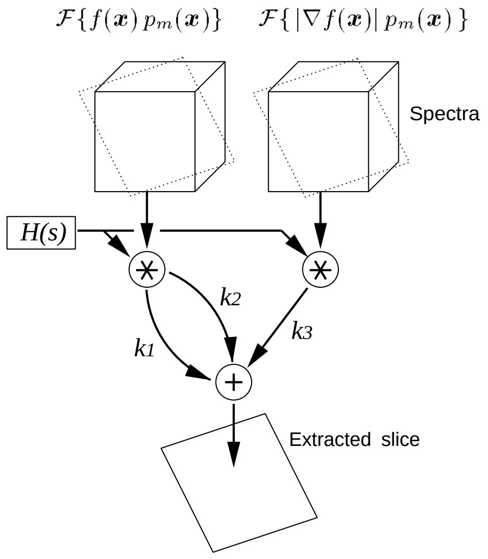
$\mathcal{F}\{f(\boldsymbol{x})\, p_m(\boldsymbol{x})\}$
$\mathcal{F}\{|\nabla f(\boldsymbol{x})|\, p_m(\boldsymbol{x})\}$
Spectra
H(s)
k2
k3
k1
Extracted slice

Figure 5: Shading computation in frequency domain. $k_1 = C_{amb}\, L_{amb}$ (ambient term), $k_2 = i\pi C_{dif}\, L_{dif}\, (\boldsymbol{s}\cdot\boldsymbol{L})$ (shading term), $k_3 = \frac{1}{2} C_{dif}\, L_{dif}$ (constant term).

The first term corresponds to the ambient term and the $(\boldsymbol{N}\cdot\boldsymbol{L})$ part of equation (9) while the second term corresponds to the accompanying constant 1. Once $f(\boldsymbol{x})\, p_m(\boldsymbol{x})$ and $|\nabla f(\boldsymbol{x})|\, p_m(\boldsymbol{x})$ are Fourier transformed, the shading computation can be performed during slice extraction (figure 5). Note that the interpolation filter H is applied first in order to reconstruct the pure spectrum of $f(\boldsymbol{x})$ from the premultiplied volume. Then, the first moment of the spectrum is computed to apply the directional shading.

Although computing a moment incurs a few additional floating point operations per sample on the slice, the additional expense is small relative to the number of operations that are required to evaluate the convolution at the sample point. It should also be noted that equation (11) can be easily extended to multiple light sources. In this case, we only have to add the moment terms for additional light sources. The increase in the computation cost is minor.

Figure 6(c) shows a projection shaded using this technique. As before, the method operates entirely in the frequency domain and requires computations only on the plane of the slice being extracted

The major drawback of this shading model is that it requires a second spectrum, $\mathcal{F}\{|\nabla f(\boldsymbol{x})|\, p_m(\boldsymbol{x})\}$ since there is no simple way to compute a gradient magnitude in the frequency domain. Hence, two slices must be extracted from two volumes. A linear shading equation such as $C_{amb}\, L_{amb}\, f(\boldsymbol{x}) + C_{dif} L_{dif}\, \nabla f(\boldsymbol{x})$ that requires only one volume can be derived under an appropriate interpretation. However, the upper bound of C_{dif} is restricted in order not to generate negative values and consequently the shading effect is restricted.

3.3 Combining Depth Cueing and Shading

It is possible to combine the depth cueing and directional shading techniques described in the foregoing section. When the two techniques are used together, the shading must be applied first. Otherwise, distortion by the depth cueing would result in incorrect gradient vector by which the shading effect is computed. However, this order of operation requires two convolutions: one performed before the shading computation to recover F by interpolation filter H and one performed after shading in order to apply the depth cueing function. This approach makes depth cueing no longer an inexpensive operation since we can't use the composite filter H'.

We can work around this problem by reversing the order of shading and depth cueing and then adjusting the result to get the desired effect. Using this ordering, we employ the composite filter H' to perform the interpolation and the depth cueing at once. As we will see, for practical settings, even this adjustment is not necessary.

Here, we will examine the effect of reversed order operation in spatial domain. We focus on the gradient term of the shading equation (second term of equation (10)) since other terms are not affected by the order. Applying depth cueing function $d(\boldsymbol{x})$ to equation (10), we obtain the shaded and depth cued term. Omitting the coefficient $\frac{1}{2}C_{dif}\,L_{dif}$, the gradient term is $(\nabla f(\boldsymbol{x})\cdot\boldsymbol{L}\,)\,d(\boldsymbol{x})$. Reversing the order of computation, we get

$$\begin{aligned}&\big(\nabla[f(\boldsymbol{x})\,d(\boldsymbol{x})]\cdot\boldsymbol{L}\,\big)\\ =\;&(\nabla f(\boldsymbol{x})\cdot\boldsymbol{L}\,)\,d(\boldsymbol{x})\;+\;f(\boldsymbol{x})\,(\nabla d(\boldsymbol{x})\cdot\boldsymbol{L}\,)\end{aligned}\qquad(12)$$

The second term is the difference from the correct value. Since $d(\boldsymbol{x})$ is a function of depth $(\boldsymbol{V}\cdot\boldsymbol{x})$, the difference can be rewritten as

$$\begin{aligned}&f(\boldsymbol{x})\,(\nabla[d_{1D}\,\big(\boldsymbol{V}\cdot\boldsymbol{x}\big)]\cdot\boldsymbol{L}\,)\\ =\;&f(\boldsymbol{x})\,d'_{1D}\,\big(\boldsymbol{V}\cdot\boldsymbol{x}\big)\;(\,\boldsymbol{V}\cdot\boldsymbol{L}\,)\end{aligned}\qquad(13)$$

where $d_{1D}(t)$ is a 1D depth cueing function. To maximize the shading effect, $\boldsymbol{L}$ is usually set perpendicular to $\boldsymbol{V}$ (i.e., the scene is illuminated from the side). In this case, the difference term becomes zero and the adjustment is not necessary. An example of this common special case is shown in figure 6(d).

If $(\boldsymbol{V}\cdot\boldsymbol{x})$ is non-zero, we need an adjustment. For linear depth cueing, the difference term including all the coefficients is

$$\frac{1}{2}C_{cue}C_{dif}\,L_{dif}\;f(\boldsymbol{x})\,(\boldsymbol{V}\cdot\boldsymbol{L})\qquad(14)$$

which we can compute during slice extraction without convolution. For a more complex depth cueing function, a convolution is necessary.

4 Reducing Rendering Time

Although the interpolation required in order to extract an arbitrarily oriented slice from the 3D spectrum is $O(n^2)$, it consumes most of the running time. As might be expected, the cost of this interpolation step is almost entirely determined by the size of the filter. For the $3\times3\times3$ filter we employ, 27 input samples contribute to each output sample. If we instead employed a $1\times1\times1$ filter, only one input sample would contribute to each output sample, a great saving in time. Because a smaller filter has less sharp cut off in spatial domain, the resulting image would contain strong ghosts if it were used uniformly over the entire interpolation process. However, by adaptively changing the filter size, we can reduce rendering time while maintaining high image quality.

Most of the energy in a spectrum usually resides in a small number of low frequency components, while the vast majority of high frequency components are nearly zero. We have observed that usually 99% of the energy is contained by about 10% of the frequency components.

This property makes an adaptive scheme which selects an inexpensive filter for weak frequency components very attractive. For simplicity, let us consider interpolation of a 1D spectrum F by two filters; a larger filter H_1 and a smaller filter H_2. Each input sample component is filtered or scattered by either H_1 or H_2 according to its strength. Let F_1 be the set of those samples that are filtered by H_1 and F_2 be those filtered by H_2. Obviously, $F_1+F_2=F$. The correct result we want is $F*H_1$ or in the spatial domain, $f\,h_1$. The adaptive scheme can thus be written as follows:

$$\begin{aligned}&\mathcal{F}^{-1}\{\,F_1*H_1\;+\;F_2*H_2\,\}\\ =\;&\mathcal{F}^{-1}\{\,F*H_1\;+\;F_2*(H_2-H_1)\,\}\\ =\;&f\,h_1+f_2\,(h_2-h_1)\end{aligned}\qquad(15)$$

The term $f_2\,(h_2-h_1)$ denotes the difference between the adaptively filtered image and the correct image. The mean square error is given by integrating the power of this error term. Using Rayleigh's theorem, its upper bound is given in the frequency domain as follows.

$$\begin{aligned}&\frac{1}{L}\int_{-\infty}^{+\infty}|\,f_2\,(h_2-h_1)\,|^2\,dx\\ \le\;&\frac{1}{L}\,h^2_{d\text{-}max}\int_{-\infty}^{+\infty}|\,f_2\,|^2\,dx\\ =\;&\frac{1}{L}\,h^2_{d\text{-}max}\int_{-\infty}^{+\infty}|\,F_2\,|^2\,ds\end{aligned}\qquad(16)$$

where L is the length of the non-zero region of f and $h_{d\text{-}max}$ is the maximum of $|h_2-h_1|$. This upper bound allows us to select input samples to be filtered by H_2 such that the mean square error of the rendered image is below a user defined tolerance. Similar analysis provides an upper bound for the mean square error when more than 2 filters are employed. The idea extends straightforwardly to 3D discrete signals.

This adaptive scheme is incorporated to the slice extraction as follows. First, each sample in the 3D spectrum is examined, and those whose magnitude is small enough to satisfy equation (16) are marked. This process is done only once after a volume data is transformed to the frequency domain. During slice extraction, each sample point on the slice plane is visited. If for a given sample point all of the 3D spectrum voxels that fall within the support of the larger filter are marked, the smaller filter is employed instead.

It is possible to improve this scheme further. To avoid testing all voxels falling within the support of the larger filter, we modify the preprocess to mark only those voxels that themselves satisfy equation (16) and for which all neighboring voxels lying within a distance from them equal to one-half of the support of the larger filter satisfy the equation. Given this more conservative marking, it is sufficient during slice extraction to test the spectrum voxel closest to the slice sample position. If that voxel is marked, we know without visiting any other voxels that it is safe to employ the smaller filter.

5 Reducing Memory Cost

Because the 3D spectrum is complex and requires a floating point representation due to its large dynamic range, a straightforward implementation using a double precision format consumes 16 times more memory than a spatial domain algorithm[3]. This explosion in memory cost can be controlled by using the Hartley transform [10] and a shorter number representation.

The Hartley transform is a direct relative of the Fourier transform [2]. The transform is defined as follows:

$$\mathcal{H}\{f(x)\}\;=\;F_H(s)\;=\;\int_{-\infty}^{+\infty}f(x)\,\mathrm{cas}2\pi sx\,dx\qquad(17)$$

[3]Assuming each voxel is represented by one byte in the spatial domain algorithm. With shading, spatial domain algorithms require more memory.

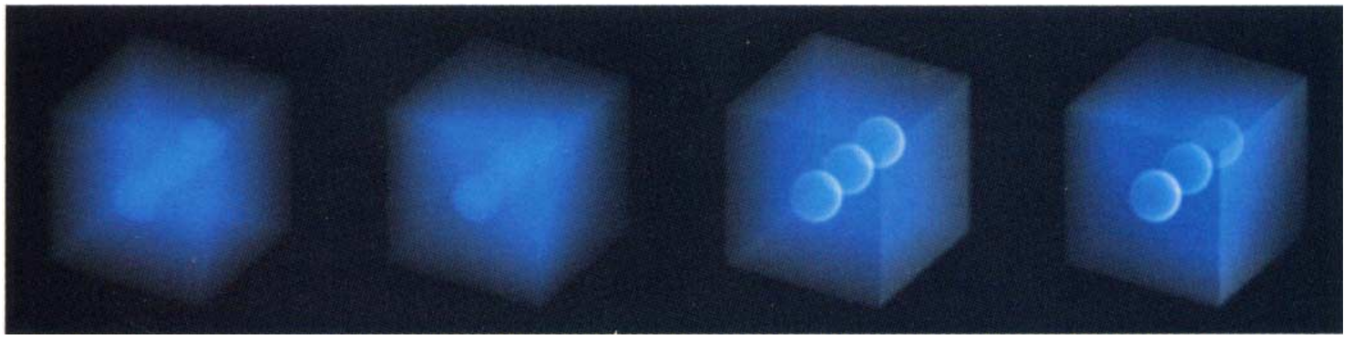

Figure 6: Examples of frequency domain depth cueing and shading. (a) projection without depth cueing, (b) linear depth cueing, (c) directional shading without depth cueing, (d) directional shading with depth cueing.

where cas$2\pi sx = \cos 2\pi sx + \sin 2\pi sx$. Since the kernel is a real function, this transform maps a real function $f(x)$ to a real spectrum $F_H(s)$. Use of the Hartley transform, therefore, eliminates the need for a complex number. Since the Fourier spectrum of a real signal is hermitian[4], the same amount of memory saving is possible with the Fourier transform by dropping half of the spectrum (e.g., store only the positive coefficients along the S_x axis). However, such implementation would unnecessarily complicate the slice extraction process.

Due to wide dynamic range of spectra, a floating point format is necessary. Considering the necessity of premultiplying the volume before transforming, a 64-bit double precision format is a safe choice to represent a spectrum of a 256^3 volume. However, even using the Hartley transform, this occupies 8 times more memory than the original volume. This problem can be minimized by using a shorter floating point format. We have defined and used a 16-bit floating point format which reduces the memory cost factor to two.

6 Results

Figures 7-9 show images rendered using the algorithms we have described. The shading, depth cueing, adaptive filtering, the Hartley transform, and the 16-bit floating point format are all used in rendering these three images.

Figure 7 shows a human skull mounted in a lucite head cast. The data was acquired using computed tomography (CT). Zeros are padded to the original data (106^3) and resulting 128^3 volume data was rendered. The volume is shaded by a hemispherical light source located to the right and is also linearly depth cued with respect to the observer's position.

The use of multiple light sources is shown in figure 8. A polygonalization of the Utah teapot has been 3D scan-converted into a 256^3 volume data which is then shaded by a red, a green, and a blue light located perpendicular to the observer and 120 degrees apart. The resulting color on the surface provides some intuition for the orientation of the gradient vector.

Figures 9 and 10 compare the frequency domain rendering technique with a conventional spatial domain volume rendering. These images were generated using identical shading and depth cueing. There is no visible difference between the two images.

The adaptive filtering scheme described in section 4 was implemented using a $3 \times 3 \times 3$ and a $1 \times 1 \times 1$ filter with the maximum

[4]A signal whose real part is even and whose imaginary part is odd, i.e. $f(x) = f^*(-x)$.

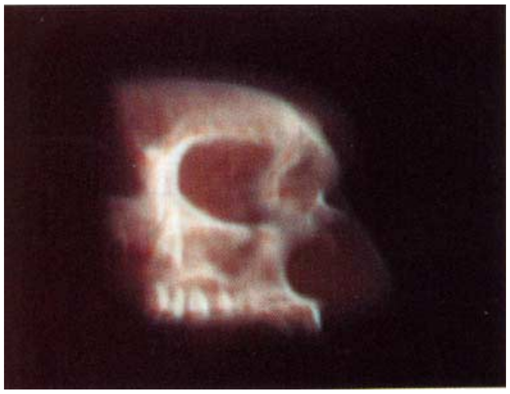

Figure 7: Human head. Frequency domain volume rendering. Data courtesy of North Carolina Memorial Hospital.

difference in response set to ($h_{d\text{-}max}$) 0.3. Figures 7-9 were generated using this scheme. As shown in table 1, the scheme reduced the cost of interpolation to about 15% of the non-adaptive case. Relative error was always below 40dB, a level at which image differences are not visible.

Table 1 also shows rendering times to generate figures 7-9. Rendering times by a spatial domain renderer are also shown for comparison. These times include all necessary operations to create a 2D projection. For the frequency domain rendering technique, it consists of slice extraction (interpolation and resampling), inverse Harteley transform, and format conversion to and from the 16-bit floating point format and the machine's native format. Times were measured on an IRIS Crimson with a 50Mhz R4000 processor using non-optimized code. As the table shows, the running time of the frequency domain method grows much slower than the spatial domain method, which grows at $O(n^3)$.

The effect of round off error caused by the 16-bit floating format was very small. Relative difference from images generated using a 64-bit double precision representation were below 50dB. Figures 7-9 were generated using this format.

7 Conclusions

The use of the Fourier projection slice theorem allows us to replace the $O(n^3)$ spatial domain projection computation that arises

Volume data	Size	Adaptive filtering			Rendering time	
		Non adaptive	Adaptive			
		Num. ops.†	Num. ops.†	(Ratio)	Freq. domain	Spatial domain
Head	128^3	5.92×10^5	1.01×10^5	(17.1%)	0.54 sec	3.15 sec
Teapot	256^3	1.81×10^6	2.33×10^5	(12.9%)	1.77	24.29
Turbine	256^3	1.85×10^6	3.00×10^5	(16.2%)	2.03	24.38

†A filtering operation consists of a filter table look up, a reference to a voxel, a multiplication, and an addition.

Table 1: Effect of adaptive filtering

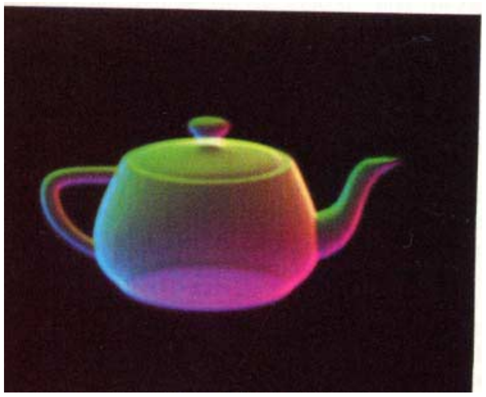

Figure 8: Utah teapot. Frequency domain volume rendering. The pot is lit by a red light (right), a green light (upper left), and a blue light (lower left).

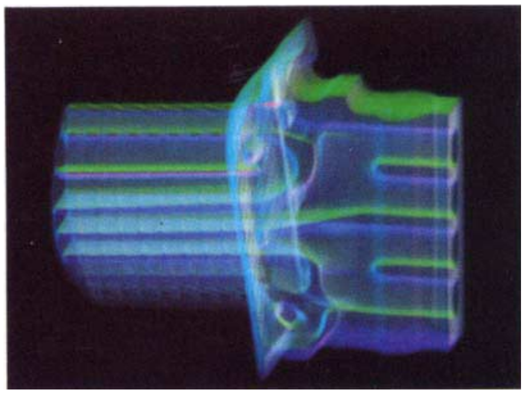

Figure 9: Turbine blade. Frequency domain volume rendering. The blade is lit by a green light (top), a blue light (bottom), and a dim red light (right). Data courtesy of General Electric.

in volume rendering with an $O(n^2 \log n)$ frequency domain computation, although the frequency domain projection operator is non-occluding, resulting in a loss of realism. In this paper, we have shown that other $O(n^3)$ spatial domain rendering computations that arise in volume rendering (i.e., shading and depth cueing) can be replaced with $O(n^2)$ frequency domain methods, and we propose that a judicious selection of these methods can restore much of the realism lost by using a non-occluding projection.

The speed advantage of our algorithm over volume rendering is considerable. As our experiments show, a 128^3 volume can be rendered in a fraction of a second on a conventional workstation. Further optimization of the code should achieve interactive rendering without specialized hardware.

Besides its speed advantage, the frequency domain approach lends itself to simple and elegant speed-accuracy tradeoffs. By extracting only the central portion of the 3D spectrum present on a slice, a renderer could provide a low resolution image quickly while the user is rotating the volume, to be replaced with a higher quality image when the mouse button or joystick is released.

Since the core computations of the algorithm are convolution and the FFT, an implementation using digital signal processors (DSPs) obviously suggests itself. With the growth of multimedia applications involving video and sound encoding and decoding, such processors are becoming a standard part of most graphics workstations. It should also be noted that these computations exhibit high data level parallelism and can be parallelized in any one of several ways.

With regard to limitations and improvements, further effort should be made to relax the limitations imposed by the linear nature of the Fourier/Hartley transform. The algorithm currently does not allow non-linear attenuation.

Acknowledgements

The authors wish to thank Tom Malzbender for helpful suggestions and his interpolation filter coefficients and Ronald Bracewell for useful hints on the use of the Hartley transform. The notion that shading could be factored with respect to digital compositing, an idea that inspired the present work, was suggested by Brice Tebbs. Discussions with Adam Levinthal were useful in the early stages of this project. Hide Hirase's volume modeling toolkit helped us creating test datasets.

This research was supported by the National Science Foundation (NSF), the National Aeronautics and Space Administration (NASA), and the sponsoring companies of the Stanford Center for Integrated Systems (CIS).

References

[1] Bracewell, Ronald, *The Fourier Transform and its Applications, revised second edition*, McGraw-Hill, 1986.

[2] Bracewell, Ronald, *The Hartley Transform*, Oxford University Press, 1986.

[3] Cohen, Michael and Greenberg, Donald, "The Hemicube: A Radiosity Solution for Complex Environments", *Computer Graphics*, Vol.19, No.3, pp.31-40, 1985.

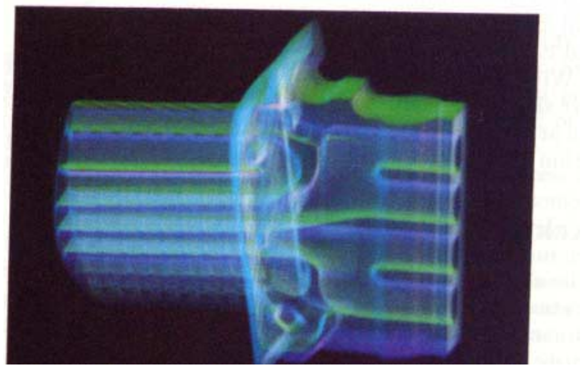

Figure 10: Same dataset as figure 9. Rendered with identical shading and depth cueing but using a spatial domain volume renderer.

[4] Drebin, Robert, Carpenter, Loren, and Hanrahan, Pat, "Volume Rendering", *Computer Graphics*, Vol.22, No.4, pp.65-74, 1988.

[5] Dunne, Shane, Napel, Sandy, and Rutt, Brian, "Fast Reprojection of Volume Data", *Proceedings of the First Conference on Visualization in Biochemical Computing*, IEEE Computer Society Press, pp.11-18, 1990.

[6] Hottel, Hoyt, and Sarofim, Adel, "Radiative Transfer", McGraw-Hill, 1967.

[7] Levoy, Marc, "Display of Surfaces from Volume Data", *IEEE Computer Graphics and Applications*, Vol.8, No.3, pp.29-37, 1988.

[8] Levoy, Marc, "Efficient Ray Tracing of Volume Data", *ACM Transactions on Graphics*, Vol.9, No.3, pp.245-261, 1990.

[9] Levoy, Marc, "Volume Rendering using the Fourier Projection-Slice Theorem", *Proceedings of Graphics Interface '92*, Canadian Information Processing Society, pp.61-69, 1992.

[10] Malzbender, Tom, "Fourier Volume Rendering", *ACM Transactions on Graphics*, Vol.12, No.3, July 1993.

[11] Napel, Sandy, Dunne, Shane, and Rutt, Brian, "Fast Fourier Projection for MR Angiography", *Magnetic Resonance in Medicine*, Vol.19, pp.393-405, 1991.

[12] Nishita, Tomoyuki and Nakamae, Eihachiro, "Continuous Tone Representation of Three-Dimensional Objects", *Computer Graphics*, Vol.20, No.4, pp.125-132, 1986.

[13] Pentland, Alex, "Linear Shape from Shading", *International Journal of Computer Vision*, Vol.4, pp.l53-162, 1990.

[14] Subramanian, K.R. and Fussel, Donald, "Applying space subdivision techniques to volume rendering", *Proceedings of the First IEEE Conference on Visualization. (Visualization '90)*, IEEE Computer Society Press, pp.150-159, 1990.

[15] Westover, Lee, "Footprint Evaluation for Volume Rendering", *Computer Graphics*, Vol.24, No.4, pp.367-376, 1990.

[16] Zuiderveld, Karel, Koning, Anton, and Viergever, Max, "Acceleration of ray-casting using 3D distance transforms", *Proceedings of the SPIE – Visualization in Biomedical Computing 1992*, Vol.1808, pp.324-335, 1992.