Fourier Volume Rendering

TOM MALZBENDER
Hewlett-Packard Labs

In computer graphics we have traditionally rendered images of data sets specified spatially. Here, we present a volume rendering technique that operates on a frequency domain representation of the data set and that efficiently generates line integral projections of the spatial data it represents. The motivation for this approach is that the Fourier Projection-Slice Theorem allows us to compute 2-D projections of 3-D data sets using only a 2-D slice of the data in the frequency domain. In general, these "X-ray-like" images can be rendered at a significantly lower computational cost than images generated by current volume rendering techniques. Additionally, assurances of image accuracy can be made.

Categories and Subject Descriptors: F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems—*computation of transforms*; I.3.3 [**Computer Graphics**]: Picture/Image Generation—*display algorithms*; I.3.6 [**Computer Graphics**]: Methodology and Techniques—*graphics data structures and data types*; I.4.1 [**Image Processing**]: Digitization—*sampling*; I.4.10 [**Image Processing**]: Image Representation—*volumetric*

General Terms: Algorithms, Performance, Theory

Additional Key Words and Phrases: Fourier transform, Hartley transform, projection, sampling, scientific visualization, volume rendering

1. INTRODUCTION

Volume rendering entails the conversion of multidimensional data sets into two-dimensional (2-D) images without the use of intermediate geometrical primitives, such as polygons. Often, the data sets one wishes to view are point samples of some scalar quantity taken uniformly in three-dimensional (3-D) space, or that can at least be converted to such a representation. Current volume rendering techniques fall into two main classes: a screen space approach and an object space approach [11–13, 16]. In the screen space approach [14, 19], a ray is cast for each pixel, and preprocessed data are uniformly sampled and blended along the ray. In the object space approach [8, 20, 22], data are traversed either front to back or back to front, and each sample is progressively blended into the image.

In some sense, both of these approaches have complexity of $O(N^3)$ for an $N \times N \times N$ data array, since each sample point should be visited. Some

Author's address: Hewlett-Packard Labs, 1501 Page Mill Road, Palo Alto, CA 94304.

adaptive volume rendering techniques can avoid accessing every sample of the data set. For example, in ray cast volume rendering one can terminate front-to-back rays when ray opacity values are close to unity [15]. Also, hierarchical data structures can be used to avoid visiting volumes of empty space. Speed improvements gained by such techniques are highly data-set dependent, and complexity remains $O(N^3)$.

We present a technique that computes projection images of an N^3 scalar array with complexity $O(N^2 \log N)$ after a preprocessing step. In practice, the computational complexity is even better than this, since an $O(N^2)$ resampling cost dominates the computation for practical values of N. In fact, these projection images can typically be computed with one to two orders of magnitude fewer operations than either the screen space approach or the object space approach. 3-D spatial data are first transformed into the frequency domain with a one-time preprocessing operation. This is accomplished with either the 3-D Fast Fourier Transform (FFT) or the Fast Hartley Transform (FHT). Once this is done, projection images at arbitrary angles can be quickly generated by resampling along a plane oriented perpendicular to the viewing direction and by taking an inverse 2-D transform of the resultant array. A similar approach was independently developed in [10]. Here we also present filter design and spatial data preprocessing techniques that allow for artifact-free renderings.

2. NOTATION

For clarity, we refer to continuous functions in script and to discrete functions in roman type ($f(x, y, z)$, f(x, y, z)). Also, we use lowercase to denote the spatial domain and uppercase to denote the frequency domain ($f(x, y, z)$, $F(w_x, w_y, w_z)$). We also restrict our discussion to N × N × N cubic data arrays, where N is a power of 2. This simplification allows Radix-2 FFT, FHT procedures to be used.

3. TOMOGRAPHY

Volume rendering can be seen loosely as the inverse problem of tomographic reconstruction. In tomographic reconstruction, our goal is to compute the unknown distribution $f(x, y, z)$ that leads to a set of measured projections. On the other hand, in volume rendering we are given the distribution and are asked to compute projections of it. In CAT scan reconstruction, we start with a set of one-dimensional (1-D) projections of the 2-D density distribution we are trying to measure. These are usually collected by sweeping an X-ray emitter/detector pair at some angle, as shown in Figure 1, yielding a projection "shadow":

$$p_\theta(r) = -\log\left(\frac{I(r)}{I_0(r)}\right) = \int_{-\infty}^{\infty} f(x, y)ds,$$

where $I_0(r)$ and $I(r)$ are emitter and detector amplitudes, respectively.

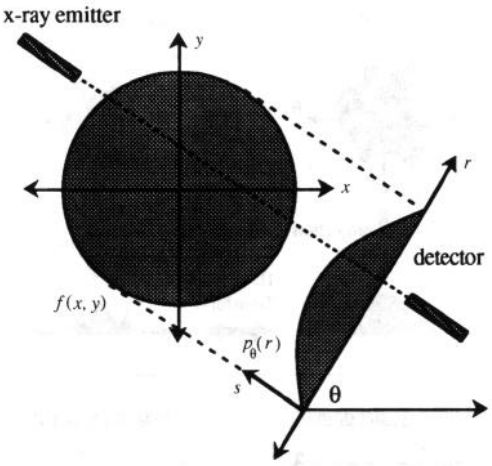


Fig. 1. CAT scan projection geometry.

This is repeated for several hundred angles and reconstruction techniques are used to compute the object's 2-D X-ray density distribution from this set of 1-D projections. One way to achieve this reconstruction is through the use of the Fourier Projection-Slice Theorem [4, 6, 9]. If $F(w_x, w_y)$ is the Fourier transform of $f(x, y)$ and $P_\theta(w_r)$ is the Fourier transform of $p_\theta(r)$, then this theorem states that

$$P_\theta(w_r) = F(w_r cos(\theta), w_r sin(\theta)).$$

Graphically, this is illustrated in Figure 2. In words, the 1-D Fourier transform of a projection of an object at some angle θ is a slice of the 2-D Fourier transform of the object at that same angle θ.

Figure 3 provides some intuitive explanation of why this might be true. Shown are a 2-D function and two of its frequency components, both in the spatial and frequency domains. Any point in the frequency domain corresponds to a sinusoid with some amplitude, phase, and orientation. If the sinusoid is not aligned with the projection direction, its projection will sum to zero. However, those components aligned with the projection direction sum to some finite value. This set of components with nonzero projections can be found in the frequency domain along a line perpendicular to the projection direction.

The Fourier Projection-Slice Theorem can be used for tomography in the following way: A set of 1-D projections of some object $f(x, y)$ are collected, each at a unique projection angle. If a 1-D Fourier transform is taken of each of these projections, they can be interpolated into a 2-D array $F(w_x, w_y)$, each at the angle that they were acquired at. When enough samples have been taken to represent this 2-D function sufficiently, a 2-D inverse transform of $F(w_x, w_y)$ can be taken to recover this original density function $f(x, y)$. There are difficulties with this simplified approach to tomography, namely, the interpolation stage. However, these details are beyond the scope of this paper [5, 17].

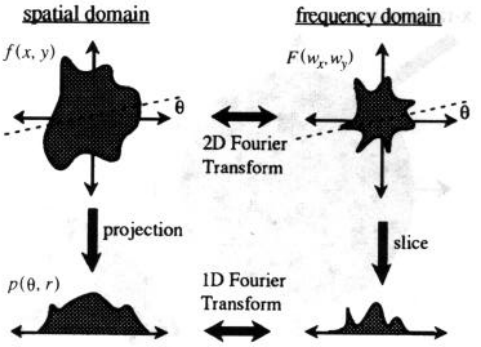


Fig. 2. Fourier Projection-Slice Theorem.

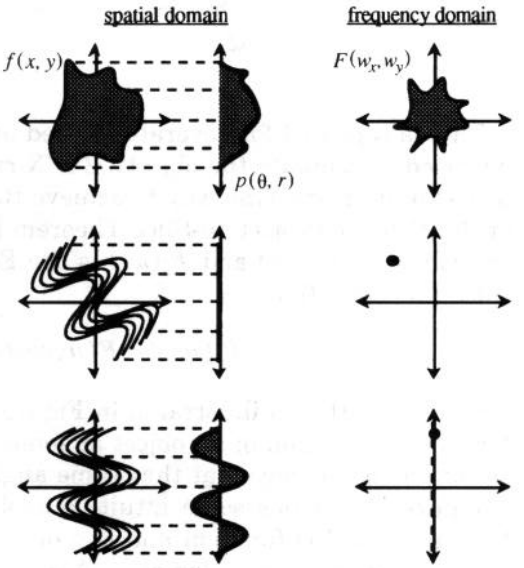


Fig. 3. Frequency components and their projections.

The Fourier Projection-Slice Theorem also holds in higher dimensions, where it is useful for volume rendering. Graphically, this is shown in Figure 4.

If we start with a 3-D continuous distribution $f(x, y, z)$ and its 3-D Fourier transform $F(w_x, w_y, w_z)$, given by

$$F(w_x, w_y, w_z) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x, y, z)e^{-2\pi i(xw_x + yw_y + zw_z)}dxdydz,$$

then a parallel projection of $f(x, y, z)$ can be computed by evaluating $F(w_x, w_y, w_z)$ along a plane defined by the arbitrary orthonormal vectors (Figure 4)

$$W_u = (w_{ux}, w_{uy}, w_{uz}),$$
$$W_v = (w_{vx}, w_{vy}, w_{vz}),$$

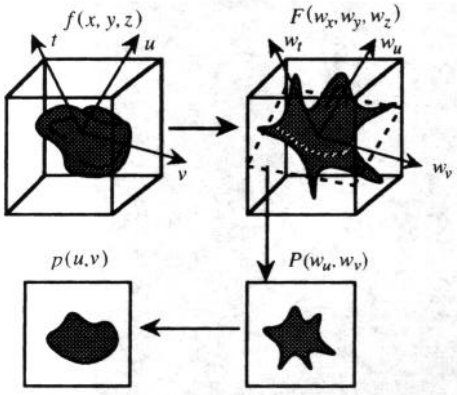


Fig. 4. Fourier volume rendering.

yielding

$$P(w_u, w_v) = F(w_{ux}w_u + w_{vx}w_v, w_{uy}w_u + w_{vy}w_v, w_{uz}w_u + w_{vz}w_v).$$

Taking its inverse 2-D Fourier transform yields the projection $p(u, v)$ along t:

$$p(u, v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} P(w_u, w_v)e^{2\pi i(uw_u + vw_v)}dw_u dw_v.$$

This technique is referred to as Fourier volume rendering (FVR). The primary motivation for this approach is that, once the forward 3-D transform is computed as a preprocessing operation, we can compute projections at arbitrary angles quickly by working with only 2-D manifolds of the data in the frequency domain.

4. WHAT FVR COMPUTES

FVR computes the function

$$p(u, v) = \int_{-\infty}^{\infty} f(t, u, v)dt.$$

Unfortunately, this type of linear projection is order independent along the line of projection (t), and therefore, hidden surface effects are not present. This limits us to transparent imagery, and the results often look like X-rays of the data set [23]. Although slower, current volume rendering techniques use opacity and color values per voxel to achieve either transparent or surfacelike effects. Some people, especially physicians, prefer X-ray style images due to their familiarity and lack of rendering effects. An image computed via FVR is shown in Figure 5, along with its frequency domain representation.

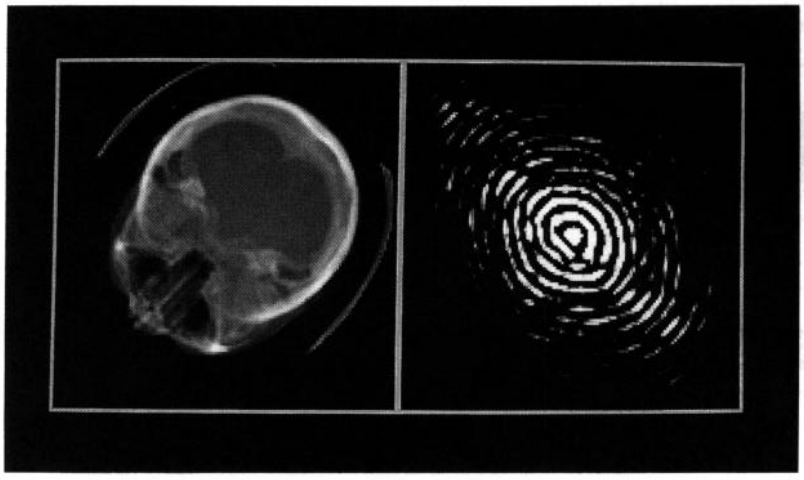


Fig. 5. FVR image of cranial CAT data set.

5. IMPLEMENTATION DETAILS

Up to now we have mainly discussed continuous functions and their continuous projection images. However, in volume rendering we are given discrete samples f(x, y, z) and are asked to compute a discrete image p(u, v). Various assumptions have been made in previous work about the nature of the continuous function, $f(x, y, z)$, that is being modeled by the given set of discrete data samples values, f(x, y, z). Traditionally, the voxel model assumed that $f(x, y, z)$ consisted of constant values within a volume element. Although attractive because of its computational simplicity, this assumption leads to discontinuous step functions between voxels. Assuming that $f(x, y, z)$ is given by trilinear interpolation of f(x, y, z) leads to an improved model, since it removes these discontinuities [14, 20]. However, discontinuities still exist in the first derivative of $f(x, y, z)$. Quadratic interpolation is, of course, possible, leading to discontinuities in the second derivative [21].

Our model of $f(x, y, z)$ is that of a band-limited function whose highest frequency content is given by the spatial sampling rate of f(x, y, z). In this case, not only is $f(x, y, z)$ infinitely differentiable, but it is also specified completely by f(x, y, z). Furthermore, $f(x, y, z)$ can be specified exactly by an arbitrary resampling rate greater than that given by f(x, y, z). This does not hold for lower-order interpolation techniques. The discreteness of our data set and desired image leads to additional considerations that will now be discussed.

5.1 Discrete Transforms

Either the FFT or the FHT can be used for the transformation both to and from the frequency domain required by FVR. The Fourier Projection-Slice Theorem holds for the FHT as well as for the FFT, the former being defined

in the forward 3-D discrete case as

$$H(k_x, k_y, k_z) = \frac{1}{N^3} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} \sum_{z=0}^{N-1} f(x, y, z) \left[\cos\left(2\pi \frac{xk_x + yk_y + zk_z}{N} \right) + \sin\left(2\pi \frac{xk_x + yk_y + zk_z}{N} \right) \right].$$

The inverse 2-D discrete FHT we need is given by

$$H^{-1}(u, v) = \sum_{k_u=0}^{N-1} \sum_{k_v=0}^{N-1} f(k_u, k_v) \left[\cos\left(2\pi \frac{uk_u + vk_v}{N} \right) + \sin\left(2\pi \frac{uk_u + vk_v}{N} \right) \right].$$

Refer to [2], [3], and [24] for fast implementations of these. We choose to use the FHT over the FFT since our input data set will most likely contain real data. Whereas the FFT of a real data set will, in general, yield a complex result, the FHT of a real data set is real. Since the FHT of a signal can be seen as taking the even part of the FFT and subtracting its odd part, we see that the two transform domains contain the same information about a signal:

$$H(w_x, w_y, w_z) = F(f_{even}(x, y, z)) - F(f_{odd}(x, y, z)).$$

For this reason, we mostly use the FFT in subsequent discussions about resampling filters due to its greater familiarity.

As with any application of FFTs or FHTs, attention must be paid to the format of the data arrays to be transformed. In particular, since zero Hertz and zero spatial position are typically held in the first location of the respective arrays, it may be necessary to shift both the spatial and frequency data logically by N/2 in each axis before and after any transformations are taken.

5.2 Resampling Considerations

After the initial preprocessing 3-D transformation step, two operations are performed to generate 2-D projections at some angle. First, one must resample the 3-D array along a plane, as shown schematically in 2-D in Figure 6. Next, we take an inverse transform of this array, yielding the resultant projection. The resampling step is usually the more computationally complex of the two and must be performed accurately to avoid aliasing artifacts.

Resampling in the frequency domain is similar to resampling in the spatial domain, involving the use of a reconstruction filter. A set of discrete samples in the frequency domain represents an infinitely periodic spatial signal [2]. In our case, the original data set is repeated over and over in each axis. Our original "space-limited" function can be retrieved by convolving our discrete

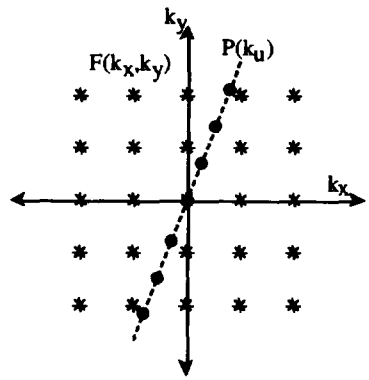


Fig. 6. 2-D frequency domain resampling.

data samples in the frequency domain by the *sinc*( ) function

$$sinc(w_x, w_y, w_z) = \frac{sin(\pi w_x)}{\pi w_x} \frac{sin(\pi w_y)}{\pi w_y} \frac{sin(\pi w_z)}{\pi w_z}.$$

The inverse transformation of this function is a cube of value 1.0 within the region occupied by the data set and 0.0 outside of it. We know from the Convolution Theorem that multiplication in the spatial domain is equivalent to convolution in the frequency domain. Therefore, since we can reconstruct our original nonperiodic spatial signal by multiplying this cube in the spatial domain, this amounts to convolving our discrete data set in the frequency domain by the *sinc*( ) function given above. This is shown graphically in Figure 7.

This forms the basis of using *sinc*( ) functions for *exact* reconstruction. Note that this convolution needs to be evaluated only on the 2-D lattice of points that we will pass to the 2-D inverse FHT. In practice, the *sinc*( ) function given above is an unacceptable reconstruction filter, since it is of infinite extent. We must find approximations that will introduce minimal error.

5.2.1 2*x Resampling*. The Fourier Projection-Slice Theorem tells us that the Fourier transform $P(w_u, w_v)$ of the continuous projection $p(u, v)$ of $f(x, y, z)$ lies along a plane of $F(w_x, w_y, w_z)$ that passes through the origin. $P(k_u, k_v)$ must sample $P(w_u, w_v)$ at a high enough rate to prevent aliasing. If the samples used for the inverse FHT are taken at the same spacing that the data are given in the frequency domain, then aliasing errors can inadvertently be introduced. This aliasing occurs in the spatial domain as a simple overlapping of copies of the original data set. The more familiar example of aliasing is overlapping in the frequency domain, as when a continuous spatial signal is not sampled at a high enough rate. However, here the overlap occurs in the spatial domain, since we are sampling in the frequency domain. If $F(w_x, w_y, w_z)$ is represented by samples at a spacing F_0 in each axis, then sampling at a spacing of less than $F_0/\sqrt{3}$, the diagonal of a cube, ensures that no overlap occurs in the spatial domain. As a practical matter, we

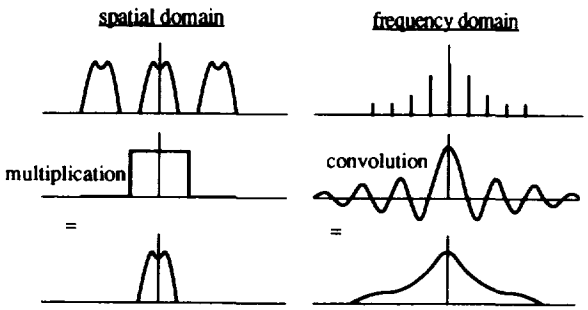


Fig. 7. 1-D continuous reconstruction from frequency domain samples.

sample at a slightly higher rate given by $F_0/2$ filling an array containing $(2N)^2$ entries that can readily be transformed by a radix-2 FHT or FFT.

5.2.2 *Band-Limited Resampling Filters.* As mentioned earlier, the *sinc*( ) function is an unacceptable reconstruction filter due to its infinite extent and needs to be replaced by some finite-extent approximation, $g(x, y, z)$. We have explored two techniques for generating $g(x, y, z)$, both leading to practical filters. The first of these involves windowing the *sinc*( ) function. Of course, one could clip our *sinc*( ) to some region by multiplying it with a cube of some extent. However, this amounts to convolving the spatial representation of the *sinc*( ), a cube, with the transform of the clipping cube, and we are left with a reconstruction filter that is finite in the frequency domain, but that has considerable "tails" in the spatial domain. A better choice for a clipping function in the frequency domain is the Hamming Window, which has the following form:

$$W(w_x, w_y, w_z) = \left[\alpha + (1 - \alpha)\cos\left(2\pi\frac{w_x}{F_1}\right)\right]\left[\alpha + (1 - \alpha)\cos\left(2\pi\frac{w_y}{F_1}\right)\right]$$
$$\left[\alpha + (1 - \alpha)\cos\left(2\pi\frac{w_z}{F_1}\right)\right],$$

where F_1 is the width of the window and $\alpha = 0.54348$. This leads to a lower side lobe height in the spatial domain than windowing with a cube.

5.2.3 *Projection on Convex Sets.* An approach we find more effective in designing a reconstruction filter is the use of the Projection on Convex Set (POCS) technique. POCS theory is omitted here due to its complexity, but can be found in [7] and [18]. In short, it allows constraints in both the frequency and spatial domains to be optimized or satisfied, if possible. The filter we are looking for has two constraints. First, we would like the filter function in the frequency domain, $G(w_x, w_y, w_z)$, to be of finite, in fact, small, extent, since larger filters lead to more operations required to evaluate each reconstruction point in the frequency domain. Second, we would like the function also to have limited extent in the spatial domain, since the role of the spatial

function is to limit our infinitely periodic spatial signal to a single cycle, corresponding to our data set. We would like $g(x, y, z)$ to approximate a cube the size of our data set. The two constraints can never be satisfied, since the Uncertainty Theorem tells us that a spatially limited function must have infinite extent in the Fourier domain and vice versa. However, we can use POCS to find functions that are truly limited in the frequency domain and that are of low amplitude outside some region in the spatial domain.

To use POCS for our application, we start with some approximation to the *sinc*( ) function, such as a Hamming windowed *sinc*( ), $G_0(w_x, w_y, w_z)$. We transform this filter into the spatial domain, yielding an infinite extent function with "tails." We now apply our spatial constraint by chopping off the tails of this function, yielding a space-limited function, $g_1(x, y, z)$. This is transformed back into the frequency domain, and the frequency domain constraint is applied by chopping off $G_1(w_x, w_y, w_z)$, yielding the band-limited $G_2(w_x, w_y, w_z)$. This concludes one POCS iteration. This procedure is iterated until the change from one iteration to the next is below some threshold. We stop this procedure with the truncation in the frequency domain, so that we are left with a filter that is truly limited in the frequency domain. This will limit the number of samples involved in the convolution.

5.2.4 *Spatial Premultiplication.* Figure 8 shows a 1-D reconstruction filter in both the frequency and spatial domains, along with the spatial function it is approximating. Although we design and employ much better filters, we have chosen a triangle function corresponding to linear interpolation, since difficulties with any filter are easy to see with this example. Although our reconstruction filters are 3-D, we show their 1-D analogs for clarity. Realizable filters have two problems. First, they are not of constant amplitude within the first cycle region, indicated by the dotted line. Since $w(x, y, z)$ operates by multiplying the spatial data, this leads to an overemphasis of the central region of the data set, compared with the perimeter area. This effect can be compensated for *exactly* by multiplying the original data set by $1/(g(x, y, z))$ over one cycle, as a preprocessing operation, before the forward 3-D transformation is taken. Note that this requires one to choose a reconstruction filter in advance. We term this operation *spatial premultiplication.* Figure 9 shows one axis of a separable spatial premultiplication function that would be used in conjunction with a trilinear interpolation resampling filter. The second limitation is given in the next subsection.

5.2.5 *Spatial Zero-Padding.* A second limitation of our reconstruction filter is that $g(x, y, z)$ is nonzero outside of the first cycle of the data set. Conceptually, we can think of $g(x, y, z)$ as multiplying our infinitely periodic data set *before* the projection is taken. This is because the convolution with $G(w_x, w_y, w_z)$ can be thought of as taking place over the entire frequency domain data set, which we then point sample along a plane. Of course, we never compute the results of the convolution except at the points we are interested in. If the convolution occurs before the projection, errors introduced by inexact reconstruction filters occur in all three dimensions to our original data set. This implies that we will be projecting a function that

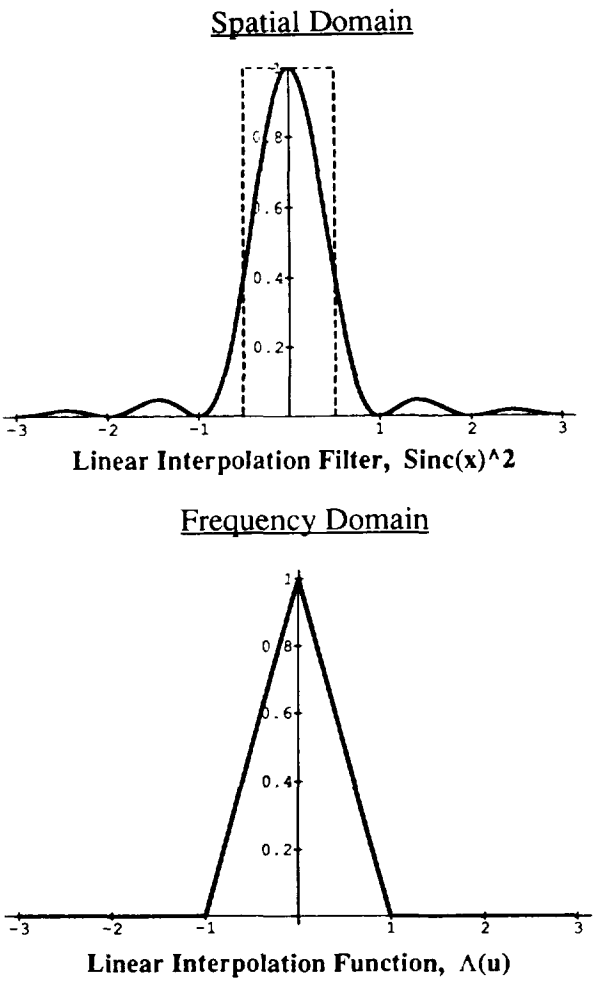
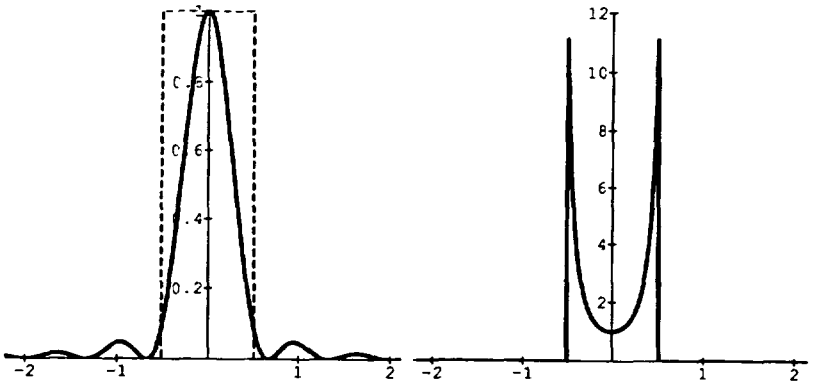


Fig. 8. Linear interpolation reconstruction filter.

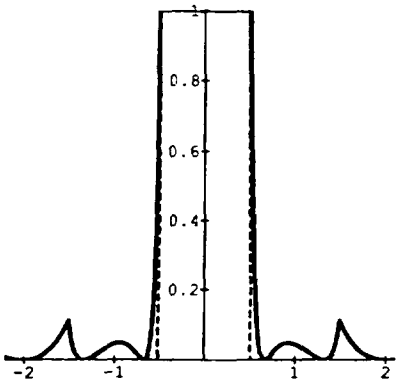
consists of our original data set periodic in each axis, albeit at a lower amplitude outside of the first cycle. At arbitrary projection angles, these copies will accumulate, leading to another aliasing artifact. We can only minimize, not eliminate, this error by designing a good reconstruction filter and by zero-padding the original data set by some small amount. The zero-padding allows the filter response, modified by data-set spatial premultiplication, as shown in Figure 9, to drop down to an acceptable level. Fortunately, for practical reconstruction filters with compact support, we can ensure that this aliasing error is less than one gray level for an 8-bit per color image.

Figure 10 shows the effect that reconstruction filter size ($m \times m \times m$) has on average aliasing performance for filters designed with the POCS method.



Spatial distortion introduced by linear interpolation resampling filter. (Inv. Fourier transform of a triangle function)

Spatial premultiplication function.



Resultant spatial distortion function.

Fig. 9. Spatial premultiplication.

The actual amount of aliasing is dependent on the content of the data set. Although we must use filters with an extent of at least $5 \times 5 \times 5$ samples to yield sub-gray-level errors, in practice, filters of size $3 \times 3 \times 3$ seem to be adequate to avoid visible artifacts. Figure 11 shows such a projection image generated with a $3 \times 3 \times 3$ POCS filter.

6. COMPLEXITY

In Figure 12 we compare the number of real multiplications and additions involved in three volume rendering techniques, namely, raytracing [14], object space compositing [22], and our FVR algorithm. The analysis given is rather simplistic and reflects only a first-order estimation. Unfortunately,

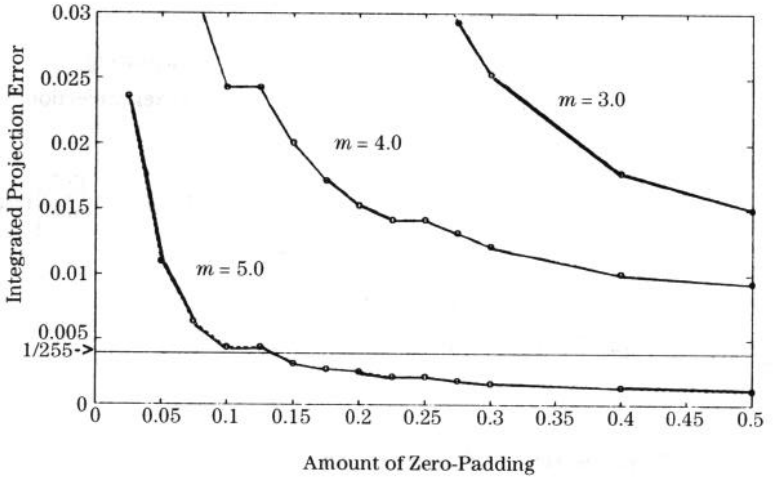


Fig. 10. Resampling filter performance.

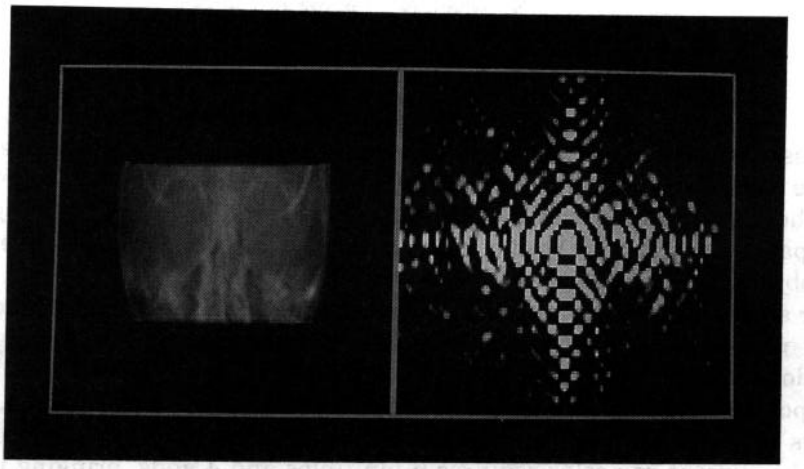


Fig. 11. FVR spatial and frequency images.

there are too many free parameters for us to show how complexity is affected by all of them. We choose to show the dependence on data-set size, and attempt reasonable assumptions for the rest. For example, image size is set to 1024×1024, and we assume that we have precomputed the color and alpha values for each voxel, in the case of raycasting and voxel space compositing. For the scene geometry, we assume a view roughly in line with one of the axes of the data set. This assumption is made solely to simplify the complexity analysis. We further assume that the data set fills the image.

Adaptive rendering techniques, such as terminating rays early, are not analyzed here, since they are highly data-set dependent. All three techniques

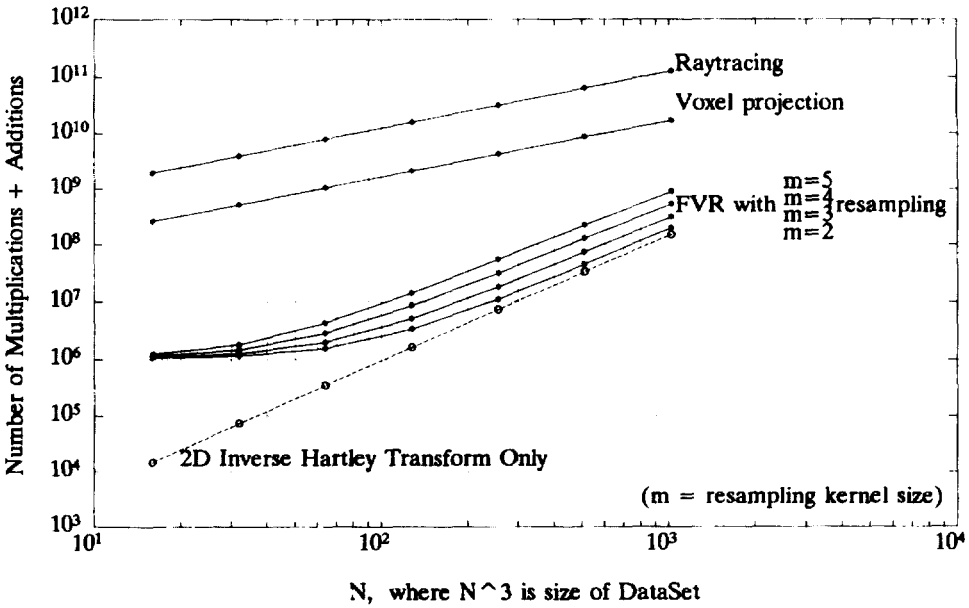


Fig. 12. Volume rendering complexity for 1K × 1K image.

discussed allow the use of such adaptive techniques. For FVR we are free to ignore those regions of frequency domain that have energy below some threshold during the resampling stage. This is similar to ignoring regions of the spatial domain with zero opacity, as can be done in both the screen space and object space approaches.

For simple raytracing we send off one ray per pixel and sample an average of N points along each ray. Trilinear interpolation requires roughly 13 additions and 14 multiplications per component.[1] Since we need to trilinearly interpolate both opacity and color values at each sample point (α, r, g, b), this brings us to 52 additions and 56 multiplications. Blending this sample point into the current ray color requires 6 multiples and 4 adds, bringing us to 56 additions and 62 multiplications per sample point. Thus, we have roughly 5.87×10^7N additions and 6.50×10^7N multiplications for a full-screen raycast image.

For voxel space compositing, we assume that all blending kernels are precomputed and that a parallel projection is taken. Each voxel is blended into the image with a 2-D kernel, assumed here to be rectangular for simplicity. To prevent holes in the composited image, blending kernels must be large enough to overlap somewhat; here we set the overlap to 10 percent in

[1]One subtraction each for generating Δx, Δy, Δz offsets, and one subtraction each for generating $(1 - \Delta x), (1 - \Delta y), (1 - \Delta z)$. We then generate four points that interpolate x, each having one addition and two multiplications. From these we generate two points that interpolate in y, and one final point that interpolates in z. This brings us to 13 additions and 14 multiplications.

each axis (x, y). For the assumed geometry, this implies a kernel size of $(1.1 \times 1024/N) \times (1.1 \times 1024/N)$ pixels. For each pixel of these kernels, 10 multiples and 4 adds are required.[2] Since we have N^3 voxels, this implies a total of $1.27 \times 10^7 N$ multiplies and $5.08 \times 10^6 N$ additions.

Finally, for FVR we have to resample the frequency domain along a plane and perform an inverse 2-D transformation. Notice that the shifting operations mentioned previously simply affect the order in which the arrays are filled and do not contribute to the overall complexity. The highest complexity arises from the resampling operation. For a stored ($m \times m \times m$) reconstruction kernel, the resampling stage incurs $3 \cdot 4m^3 (N(1 + Z_p))^2$ multiplications and $3 \cdot 4(m^3 - 1)(N(1 + Z_p))^2$ additions. Z_p is the percentage of zero-padding in each axis, set here to 10 percent. FVR can be performed independently for each color component, resulting in the factor of 3 above. The factor of 4 arises because we are resampling along the plane in the frequency domain with twice the number of samples that the data set contained in each axis. The FVR algorithm generates an image of size $2N \times 2N$. For a fair comparison of the techniques, we include the complexity of interpolating this into a larger 1024×1024 image.[3] This causes the roll-off in performance seen for small data sets in Figure 12. Note that the complexity of the forward 3-D preprocessing transform is not shown graphically, but is $6N^3 \log_2 N$ multiplies and $6N^3 \log_2 N$ additions [9].

In Figure 13 we perform a complexity comparison assuming an $N \times N$ image size, instead of holding this fixed at $1K \times 1K$, as done in Figure 12. The assumptions made are similar to those just discussed with two exceptions. First, for the voxel projection algorithm we hold the blending kernel size fixed at 3×3. Second, no screen space interpolation is now needed for FVR.

7. FREQUENCY DOMAIN DATA REPRESENTATION ADVANTAGES

There are a number of advantages to working with a frequency domain data representation, as opposed to a spatial representation. First, progressive refinement is a natural consequence of the representation. We are free to limit the samples taken to the low-frequency region of the slicing plane near the origin. This allows the complexity of both the resampling and inverse transformation stages to be reduced, resulting in a smaller image that can be interpolated up to arbitrary size.

Spatial filtering operations are also simple and computationally inexpensive to implement with a frequency domain representation. Spatial filtering

[2] Voxel colors and α's are weighted by the footprint kernel values. This requires 4 multiplies per kernel pixel. $(1 - \alpha_p)$ is also required by the over operation, which needs 2 multiplies and 1 add for each of the three color components. Therefore, we need 10 multiplies and 4 adds for each pixel of the blending kernel associated with each voxel.

[3] Screen space interpolation can be done in two passes, one in each axis. Between two points in y, we incur one subtraction, one shift, and $(1024/2N - 1)$ adds. There are roughly $4N^2$ pairs of such points. In x we incur the same number of operations per pair of points, but now there are $2048N$ such pairs. This yields a total of $(2N + 1024)^2$ operations.

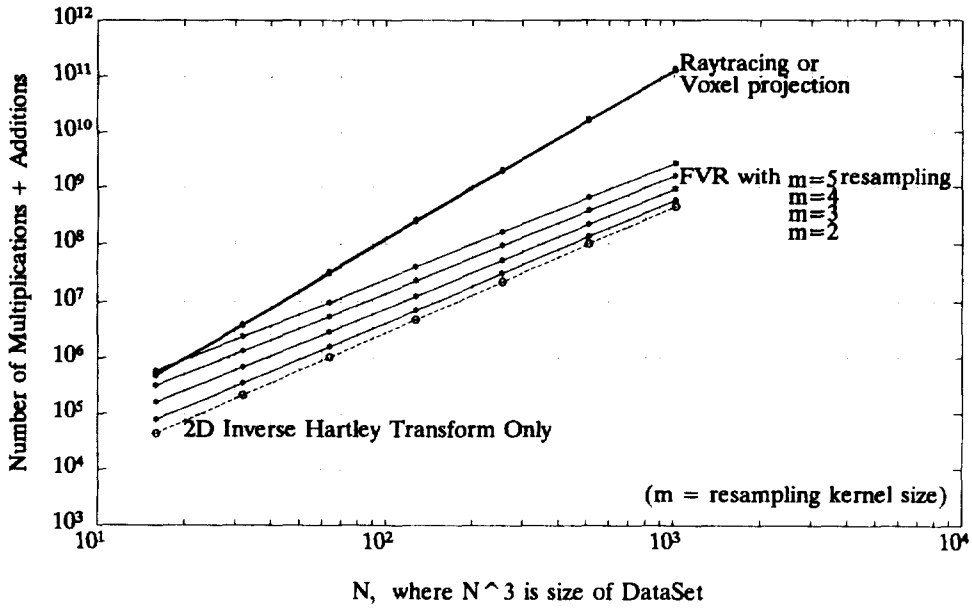


Fig. 13. Volume rendering complexity for N × N image.

is often applied in image processing, where mild high-boost filters can lead to the overall appearance of increased image sharpness. Arbitrary low, high, and bandpass filters can all be defined and implemented in the frequency domain as simple multiplications of each frequency value by a precomputed filter coefficient. This is to be contrasted with performing a more expensive convolution in the spatial domain.

Cross-correlations are useful for feature detection in image processing and can also be employed for 3-D scalar data sets [1]. If a feature $q(x, y, z)$, with transform $Q(w_x, w_y, w_z)$, is present in our data set, then it can be found by finding the maximum of the cross-correlation:

$$z(x,y,z) = \sum_{t_z=0}^{N-1} \sum_{t_y=0}^{N-1} \sum_{t_x=0}^{N-1} f(x,y,z)q(x + t_x, y + t_y, z + t_z).$$

For typical discrete 3-D data sets, however, this operation is expensive. For large data sets, this computation can be performed more efficiently in the frequency domain, since

$$Z(w_x, w_y, w_z) = F(w_x, w_y, w_z)Q^*(w_x, w_y, w_z),$$

where $Q^*()$ is the complex conjugate of $Q()$. Therefore, using the frequency domain representation, we only need to compute

$$z(x,y,z) = \mathrm{FFT}^{-1}\left[F(w_x, w_y, w_z)Q^*(w_x, w_y, w_z)\right]$$

and to find its maximum.

8. CONCLUSION AND FUTURE WORK

We have developed a technique for using the Fourier Projection-Slice Theorem in volume rendering. It allows one to compute the projection of a 3-D scalar function by evaluating only a 2-D slice of the function represented in the frequency domain. The technique is computationally less expensive than current approaches and should allow interactive display when implemented on current digital signal processing chips. The projection image computed is a strict line integral of the function, which gives it the quality of looking like an "X-ray" of the data set. The difficult problem of overcoming this limitation and of introducing hidden surface effects, perhaps through the use of alternative nonlinear transforms, is an unexplored issue.

ACKNOWLEDGMENTS

Ken Neighbors deserves thanks for the implementation of the Hartley Transform Code as well as for being instrumental in helping uncover the various aliasing mechanisms present in a naive prototype. Ron Pulleyblank is responsible for suggesting spatial premultiplication and was an effective "signal processing sounding board." The need for spatial data shifting was pointed out by John Jackson, of Stanford's MRI group. FFT code was provided for early prototypes by Vasudev Bhaskaran. The medical data set used in the figures is courtesy of I. Jackson, U. Bite, and G. Forbes at the Mayo Clinic.

REFERENCES

1. ANUTA, P. Spatial registration of multispectral and multitemporal digital imagery using fast Fourier transform techniques. *IEEE Trans. Geosci. Electron. GE-8*, 4 (Oct. 1970), 353–368.
2. BRACEWELL, R. *The Fourier Transform and Its Applications*. Rev. 2nd ed. McGraw-Hill, New York, 1986, pp. 385–410.
3. BRACEWELL, R. Assessing the Hartley transform. *IEEE Trans. Acoustics, Speech Signal Process. 38*, 12 (Dec. 1990), 2174–2176.
4. BRIGHAM, E. *The Fast Fourier Transform*. Prentice-Hall, Englewood Cliffs, N.J., 1974.
5. BROOKS, R. Computational principles of transmission CT. In *Medical Physics of CT and Ultrasound*. American Institute of Physics, New York, 1980, pp. 37–52.
6. CARTWRIGHT, M. *Fourier Methods for Mathematicians Scientists and Engineers*. Ellis Horwood, New York, 1990.
7. CIVANLAR, M. R., AND NOBAKHT, R. A. Optimal pulse shape design using projections onto convex sets. In *ICASSP 88 Proceedings*, vol. 3, (New York, Apr. 11–14). IEEE, New York, 1988, pp. 1874–1877.
8. DREBIN, R., CARPENTER, L., AND HANRAHAN, P. Volume rendering. *Comput. Graph.* 22, 4 (1988), 65–74.
9. DUDGEON, D., AND MERSEREAU, R. *Multidimensional Signal Processing*. Prentice-Hall, N.J., 1984, pp. 81, 82, 363–383.
10. DUNNE, S., NAPEL, S., AND RUTT, B. Fast reprojection of volume data. In *Proceedings of the 1st Conference on Visualization in Biomedical Computing*. 1990, pp. 11–18.
11. FUCHS, H., LEVOY, M., PIZER, S., AND ROSENMAN, J. In *NCGA '89 Conference Proceedings*, vol. 1. (Apr.). 1989, pp. 118–131.
12. HANRAHAN, P. Three-pass affine transforms for volume rendering. *Computer Graph. 24*, 5 (Nov. 1990), pp. 71–78.
13. KAUFMAN, A. *Volume Visualization*. IEEE Computer Society Press, Los Alamitos, Calif., 1991.

14. LEVOY, M. Display of surfaces from volume data. *IEEE Comput. Graph. Appl. 8*, 3 (May 1988), 29–37.
15. LEVOY, M. Display of surfaces from volume data. Doctoral dissertation, Dept. of Computer Science, Univ. of North Carolina at Chapel Hill, May 1990.
16. LEVOY, M. Viewing algorithms. In *Volume Visualization*. A. Kaufman, Ed., IEEE Computer Society Press, Los Alamitos, Calif., 1991, pp. 89–92.
17. MERSEREAU, R., AND OPPENHEIM, A. Digital reconstruction of multidimensional signals from their projections. *Proc. IEEE 62*, 10 (Oct. 1974), 1319–1338.
18. PAPOULIS, A. A new algorithm in spectral analysis and band limited extrapolation. *IEEE Trans. Circuits Syst. CAS-22*, 9 (Sept. 1975), 735–741.
19. SABELLA, P. A rendering algorithm for visualizing 3D scalar fields. *Comput. Graph. 22*, 4 (Aug. 1988), 51–55.
20. UPSON, C., AND KEELER, M. V-Buffer: Visible volume rendering. *Comput. Graph. 22*, 4 (Aug. 1988) 59–64.
21. WEBBER, R. Ray tracing voxel data via biquadratic local surface interpolation. *Visual Comput. 6*, 1 (Feb. 1990), 8–15.
22. WESTOVER, L. Footprint evaluation for volume rendering. *Comput. Graph. 24*, 4 (Aug. 1988), 367–376.
23. WHITTED, T. An improved illumination model for shaded display. *Commun. ACM 23*, 6 (June 1980), 343–349.
24. YANG, D. New fast algorithm to compute two-dimensional discrete Hartley transform. *Electron. Lett. 25*, 25 (Dec. 1989), 1705–1706.

Received May 1991; revised May 1991; accepted April 1992

Editor: Alain Fournier