An Extension of Fourier-Wavelet Volume Rendering by View Interpolation

Michel A. Westenberg (michel@cs.rug.nl) and Jos B. T. M. Roerdink (roe@cs.rug.nl)
Institute for Mathematics and Computing Science,
University of Groningen,
P.O. Box 800, 9700 AV Groningen, The Netherlands

Abstract. This paper describes an extension to Fourier-wavelet volume rendering (FWVR), which is a Fourier domain implementation of the wavelet X-ray transform. This transform combines integration along the line of sight with a simultaneous 2-D wavelet transform in the view plane perpendicular to this line. During user interaction, only low resolution images are computed based on wavelet approximation coefficients. When user interaction ceases, the images are refined incrementally with the wavelet detail coefficients. The extension proposed in this paper is similar to a technique called view interpolation, which originates from the field of computer graphics. View interpolation is used to speed up rendering of complex scenes by precomputing images from a number of selected viewpoints. For intermediate viewpoints, rendering is performed by interpolating the precomputed images. In this paper, we show that for FWVR the speed of rendering low resolution images is increased by interpolation of precomputed sets of wavelet approximation coefficients in the Fourier domain. The differences with traditional view interpolation are that (i) interpolation is performed on the wavelet approximation coefficients in the Fourier domain and not on images, and (ii) interpolation is performed during user interaction only. When interaction ceases, ordinary FWVR progressively renders an image at high quality. Medical CT data are used to assess the accuracy and performance of the method. We use regular angular sampling of spherical coordinates which determine the viewing direction. The results show that angle increments as large as 10 degrees result in only a small degradation of image quality.

Keywords: Fourier domain volume rendering, wavelet X-ray transform, client-server visualization system, view interpolation

1. Introduction

Volume rendering is a technique for visualizing digital data representing large three-dimensional (3-D) volumes, arising from physical measurements (as in computerized tomography) or from computer simulations. Volume visualization techniques have been developed for viewing these data from different viewpoints, using advanced computer graphics techniques such as illumination, shading, and colour [11]. Due to their large size, the transmission and display of these data sets is time consuming. Therefore, multiresolution models are developed, which allow decomposition of the data into versions at different levels of resolution, so that the data can be visualized incrementally as they arrive ('progressive refinement'). Wavelets are a natural candidate for such a multiresolution approach [24].

Volume visualization methods are generally divided into two classes, i.e. *surface rendering*, where one reduces the volume to a number of surfaces representing the boundary between materials [10, 13], and *direct volume rendering* [4], which does not make use of intermediate graphical primitives, but tries to map the information in the 3-D data set directly onto the view plane. A standard method in direct volume rendering, called *X-ray volume rendering*, is to integrate the volume data along the line of sight. The method supports shading and depth-cueing [20], but no occlusion or perspective projection. Nevertheless, it turns out to be one of the preferred techniques for medical applications, because physicians are well-trained in interpreting X-ray like images for diagnosis. The corresponding mathematical concept is the *X-ray transform*, well-known from computerized tomography [17]. There exists an efficient way to compute this transform, called *Fourier volume rendering* (FVR), which makes use of frequency domain techniques [15, 16, 20], and is based upon the Fourier slice theorem [17]. Note that in contrast to computerized tomography, where one has to compute the inverse X-ray transform, in volume rendering one has to compute the forward X-ray transform. The function to be visualized by computing X-ray projections is known, albeit only at a digital sampling grid. Fourier volume rendering can be summarized as follows: After an initial 3-D Fourier transform of the data, a viewing direction $\boldsymbol{\theta}$ is chosen and the values of the Fourier transform in a plane, called the *slice plane*, through the origin in Fourier space and perpendicular to $\boldsymbol{\theta}$ are computed. Interpolation in frequency space is necessary to obtain the values of the Fourier transform of the function to be visualized at a regular grid in the slice plane. A subsequent inverse 2-D Fourier transform gives the desired image in the view plane. The time complexity of FVR is $\mathcal{O}(N^2 \log N)$ for a volume data set of size $N \times N \times N$.

Recently, we developed Fourier-wavelet volume rendering (FWVR) [19, 26] as a wavelet-based extension to Fourier volume rendering. FWVR is a Fourier domain implementation of the wavelet X-ray transform [19], which combines integration along the line of sight with a simultaneous 2-D wavelet transform in the view plane perpendicular to this line. We derived in [26] an efficient algorithm for computing the wavelet X-ray transform by using a frequency domain implementation of the wavelet transform. This is particularly efficient when the length of the wavelet decomposition and/or reconstruction filters is large, as is the case for some of the basic wavelets (e.g. B-spline wavelets [3, 22]) used below. This results in an algorithm whose initial step, i.e. computation of Fourier coefficients in a slice plane in frequency space, is identical to that of ordinary FVR. The additional step is a wavelet decomposition of the slice plane data in Fourier space to a given level of detail. Approximation images are then obtained by a partial wavelet reconstruction in Fourier space, followed by a 2-D inverse Fourier transform. Since wavelet detail coefficients are available in Fourier space, progressive refinement is straightforward. Progressive refinement is important for client-server based visualization systems, where the volume data are stored on a central server, while (part of) the rendering is performed on client systems. Not all of these clients will have a high-bandwidth connection, so a mechanism which visualizes data incrementally as they arrive can improve the response time of the system. FWVR enables us to implement such a client-server visualization system.

Another wavelet-based volume rendering method based on the X-ray transform is *wavelet splatting* [12]. This is a modification of the standard splatting algorithm [28] through the use of wavelets as reconstruction filters. Splatting is an object order method in which the voxels are represented by 3-D reconstruction kernels. Integration of these kernels along the line of sight results in building blocks called *footprints.* A mapping to the image plane by superposition of the footprints, weighted by the voxel values, forms the image in the view plane. Just as the original splatting method, the time complexity of wavelet splatting is $\mathcal{O}(N^3)$ for a volume data set of size $N \times N \times N$. In contrast, FWVR has the same time complexity as ordinary FVR, i.e. $\mathcal{O}(N^2 \log N)$. For a detailed comparison of FWVR and wavelet splatting with respect to time complexity and memory requirements, the reader is referred to [26]. Recently, Horbelt et al. have shown that the computation time of wavelet splatting can be reduced by adapting the resolution of the projection grid to the size of the B-spline wavelet basis functions [6]. The projection of the wavelet coefficients on this grid yields an approximation in dual B-spline space. An image in the view plane is then obtained by B-spline interpolation [21] of the projection grid to the size of the view plane. Another method to reduce the computation time is called two-stage splatting [27]. The method separates the splatting process in two stages: (i) coefficient projection and accumulation in a weight array W, and (ii) a 2-D convolution of W with the footprint of a wavelet basis function, which yields an image in the view plane.

A disadvantage of FWVR in the form presented in [26] is that it requires resampling of a slice in Fourier space at full resolution in order to perform a 2-D wavelet decomposition. The purpose of this paper is to extend FWVR with a technique similar to *view interpolation.* This is a method used in the field of computer graphics to speed up rendering of complex scenes [1], and consists in precomputing images for a number of viewing directions. Images for intermediate viewing directions are then obtained by interpolating the precomputed images. A similar technique was introduced in an image-based volume rendering method [2] that is based upon shear-warp factorization [9]. The extension proposed in this paper uses a set of precomputed sequences of wavelet approximation coefficients in the Fourier domain for different viewing directions. The approximation coefficients for intermediate viewing directions are then computed by interpolation. This decreases the computation time to obtain an approximation image substantially, allowing fast interaction with the data. When interaction ceases, ordinary FWVR is applied to refine the image incrementally to full resolution.

The organization of this paper is as follows. Section 2 summarizes standard Fourier volume rendering including interpolation and accuracy issues, introduces the basic wavelet concepts and describes Fourier-wavelet volume rendering as introduced in [26]. In Section 3, we describe the new method which introduces view interpolation in the Fourier-wavelet domain. Section 4 presents some experimental results, and we conclude with a discussion in Section 5.

2. Fourier-Wavelet Volume Rendering

We start by summarizing the main ideas of standard Fourier volume rendering, and briefly discuss interpolation and accuracy issues, showing that with a judicious combination of zero-padding of the data and good interpolation filters accurate renderings are obtained. Then we introduce a number of basic wavelet concepts, followed by a short description of Fourier-wavelet volume rendering.

2.1. Fourier Volume Rendering

Fourier domain volume rendering methods [15, 16] provide an implementation of X-ray volume rendering, where the volume data are integrated along the line of sight. That is, if $f(\boldsymbol{x})$, $\boldsymbol{x} = (x, y, z) \in \mathbb{R}^3$, is integrated along a direction vector $\boldsymbol{\theta}$, with $\boldsymbol{u}$ and $\boldsymbol{v}$ two mutually orthogonal vectors perpendicular to $\boldsymbol{\theta}$ (see Fig. 1), then the result, also called the X-ray transform of f, is given by

$$\mathcal{P}_{\boldsymbol{\theta}} f(u, v) = \int_{\mathbb{R}} f(u\boldsymbol{u} + v\boldsymbol{v} + t\boldsymbol{\theta})\ \mathrm{d}t.$$

The Fourier projection slice theorem [7] states that the 2-D Fourier transform of $\mathcal{P}_{\boldsymbol{\theta}} f$ equals the 3-D Fourier transform of f along a slice plane through the origin in Fourier space and perpendicular to $\boldsymbol{\theta}$. Denote the n-dimensional Fourier transform of a function $f \in L^2(\mathbb{R}^n)$ by $\mathcal{F}_n f$:

$$\mathcal{F}_n f(\boldsymbol{\omega}) = \int_{\mathbb{R}^n} e^{-2\pi i \boldsymbol{\omega}\cdot\boldsymbol{x}} f(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}, \qquad \boldsymbol{\omega} \in \mathbb{R}^n.$$

Then the Fourier projection slice theorem states that

$$\mathcal{F}_2 \mathcal{P}_{\boldsymbol{\theta}} f(\omega_u, \omega_v) = \mathcal{F}_3 f(\omega_u \boldsymbol{u} + \omega_v \boldsymbol{v}). \tag{1}$$

This theorem is the key to Fourier volume rendering. Given volume data sampled on a uniform grid, the FVR algorithm consists of the following steps:

ALGORITHM 1. **FVR**

- Preprocessing. *Compute the 3-D discrete Fourier transform of the volume data by FFT.*
- Actual volume rendering. *For each direction $\boldsymbol{\theta}$, do:*
  1. *Interpolate the Fourier transformed data and resample on a regular grid of points in the slice plane orthogonal to $\boldsymbol{\theta}$ ('slice extraction').*
  2. *Compute the 2-D inverse Fourier transform, again by FFT. This yields a discrete approximation to $\mathcal{P}_{\boldsymbol{\theta}} f$.*

The first step is just preprocessing: the 3-D Fourier transform is computed only once. The next two steps are repeated for each viewing direction, and must therefore be implemented as efficiently as possible. For a slice of size N by N, the complexity of the Fourier transform is $\mathcal{O}(N^2 \log N)$, and that of 3-D interpolation is $\mathcal{O}(K^3 N^2)$, where K is the linear size of the interpolation filter (K much smaller than N). Although the Fourier transform is asymptotically dominant, in practice most of the running time is spent on interpolation.

Since interpolation is the most critical step in FVR, accurate interpolation filters are necessary to avoid artefacts such as aliasing (due to insufficient sampling), and dishing, resulting in reduced intensities away from the center of the image. To reduce aliasing, one pads the data in the spatial domain with zeros before the initial 3-D Fourier transform. The price to pay is increased memory usage. Cubic interpolation [8] with 20% zero-padding has shown to offer a good compromise for FVR, resulting in small aliasing error and small dishing artefact [26]. Cubic B-spline interpolation [21] has turned out to reduce aliasing even more. Since the computational costs of cubic interpolation and cubic

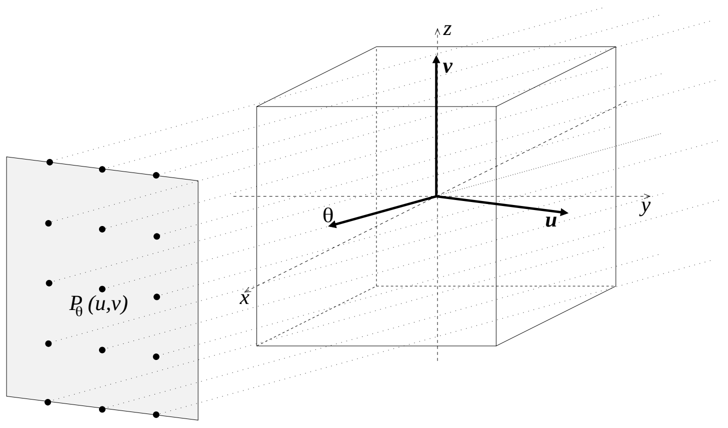


Figure 1. View plane perpendicular to the direction vector $\boldsymbol{\theta}$.

B-spline interpolation are comparable, we use the latter method for slice resampling.

The 3-D Fourier transform requires complex arithmetic and a floating point representation. Because the Fourier transform of a real signal is hermitian, a factor of two in the number of computations during slice extraction can be saved by dropping half of the Fourier transformed data, e.g. using a real-to-complex/complex-to-real FFT [5]. Also, one can reduce memory requirements by a factor of two by quantizing the floating point values to 2-byte shorts, without seriously affecting the accuracy [26].

2.2. Wavelet Representation

A wavelet decomposition of a signal is obtained by convolving the signal with an analysis filter, followed by downsampling. This results in a number of *approximation* coefficients giving the coarse features of the signal, and a set of *detail* coefficients giving the finer structure. The process can be repeated a number of times, say M; this number is called the *depth* or *level* of the decomposition. The signal can be reconstructed from the approximation and detail coefficients by upsampling, followed by convolution with a synthesis filter.

In our application, we will need two-dimensional wavelet decomposition and reconstruction filters, which are constructed from a one-dimensional biorthogonal wavelet basis. Such a basis derives from a *scaling function* ϕ with associated *basic wavelet* ψ, and dual scaling function $\tilde{\phi}$ with dual basic wavelet $\tilde{\psi}$. The corresponding basis functions are $\{\phi_{j,k}\}$ and $\{\psi_{j,k}\}$, $j,k \in \mathbb{Z}$, where $\phi_{j,k}(x) = 2^{-j/2}\phi(2^{-j}x - k)$ and $\psi_{j,k}(x) = 2^{-j/2}\psi(2^{-j}x - k)$; the dual basis functions are defined similarly. Here j and k denote scale and translation, respectively. From the 1-D basis, a 2-D separable wavelet basis is constructed with four basis functions, i.e. one scaling function $\Phi^0_{j,k,l}(x,y)$ and three wavelet basis functions $\Psi^\tau_{j,k,l}(x,y)$, $\tau \in T = \{1,2,3\}$:

$$\begin{aligned} \Phi^0_{j,k,l}(x,y) &= \phi_{j,k}(x)\phi_{j,l}(y) & \Psi^1_{j,k,l}(x,y) &= \phi_{j,k}(x)\psi_{j,l}(y) \\ \Psi^2_{j,k,l}(x,y) &= \psi_{j,k}(x)\phi_{j,l}(y) & \Psi^3_{j,k,l}(x,y) &= \psi_{j,k}(x)\psi_{j,l}(y) \end{aligned} \tag{2}$$

A similar definition holds for the dual scaling function $\widetilde{\Phi}^0_{j,k,l}(x,y)$ and wavelet basis functions $\widetilde{\Psi}^\tau_{j,k,l}(x,y)$. Then the M-level wavelet representation of a 2-D function f is given by

$$f(x,y) = \sum_{k,l} c^M_{k,l}\Phi^0_{M,k,l}(x,y) + \sum_{j=1}^{M}\sum_{\tau\in T}\sum_{k,l} d^{j,\tau}_{k,l}\Psi^\tau_{j,k,l}(x,y). \tag{3}$$

The *approximation* coefficients are $c^M_{k,l} = \langle f, \widetilde{\Phi}^0_{M,k,l}\rangle$ and the *detail* coefficients are $d^{j,\tau}_{k,l} = \langle f, \widetilde{\Psi}^\tau_{j,k,l}\rangle$, where $\langle\cdot,\cdot\rangle$ denotes the inner product in the space $L^2(\mathbb{R}^2)$ of square integrable functions on $\mathbb{R}^2$.

In the case of digital data, the *fast wavelet transform* and its inverse compute wavelet decomposition and reconstruction very efficiently by a subband filtering scheme called the pyramid algorithm [14]. The basis functions are represented by discrete filters $h = (h_n)_{n\in\mathbb{Z}}$, $g = (g_n)_{n\in\mathbb{Z}}$ for synthesis, and dual filters $\widetilde{h}$ and $\widetilde{g}$ for analysis. The 2-D basis (2) is represented by the four possible tensor products, hh, hg, gh and gg, of the 1-D filters h and g. (For example $(hh)_{k,l} = h_k\, h_l$.) Wavelet decomposition is performed recursively starting at level 0 by convolution followed by downsampling by a factor of two. Wavelet reconstruction is performed recursively starting at level M by upsampling by a factor of two followed by convolution.

2.3. Fourier-Wavelet Volume Rendering

The *wavelet X-ray transform* was introduced in [19], and an efficient implementation was derived in [26] by computing the wavelet transform in the frequency domain. The result is an algorithm that starts by computation of the Fourier transform in a slice plane, as in ordinary FVR, followed by a wavelet decomposition of the slice plane image in Fourier space. Here we summarize the main steps of this method.

2.3.1. *The wavelet X-ray transform*
The *wavelet X-ray transform* is defined by expanding the X-ray transform $\mathcal{P}_{\boldsymbol{\theta}} f$ of a function f in a 2-D wavelet series (cf. (3)):

$$\mathcal{P}_{\boldsymbol{\theta}} f(u,v) = \sum_{k,l} c^M_{k,l}(\boldsymbol{\theta}) \Phi^0_{M,k,l}(u,v) + \sum_{j=1}^{M} \sum_{\tau \in T} \sum_{k,l} d^{j,\tau}_{k,l}(\boldsymbol{\theta}) \Psi^\tau_{j,k,l}(u,v). \quad (4)$$

Note that the approximation coefficients $c^M_{k,l}$ and detail coefficients $d^{j,\tau}_{k,l}$, $\tau \in T = \{1,2,3\}$, now depend on the viewing direction $\boldsymbol{\theta}$. This transform is closely related to the wavelet X-ray transform defined in [25, 29], which combines integration over a line with a simultaneous 1-D wavelet transform along this line. The difference is, that we perform a 2-D wavelet transform in the plane perpendicular to the line.

Efficient computation of the wavelet coefficients in (4) depends on the following theorem, cf. [26] for a detailed derivation.

THEOREM 1. *The coefficients in the wavelet representation* (4) *for the X-ray transform of* $f \in L^2(\mathbb{R}^3)$ *are given by*

$$c^M_{k,l}(\boldsymbol{\theta}) = \mathcal{F}_2^{-1}\left(\mathcal{F}_2 \mathcal{P}_{\boldsymbol{\theta}} f \cdot \mathcal{F}_2 \widetilde{\Phi}'^0_M\right)(2^M k, 2^M l) \quad (5)$$

$$d^{j,\tau}_{k,l}(\boldsymbol{\theta}) = \mathcal{F}_2^{-1}\left(\mathcal{F}_2 \mathcal{P}_{\boldsymbol{\theta}} f \cdot \mathcal{F}_2 \widetilde{\Psi}'^\tau_j\right)(2^j k, 2^j l), \quad (6)$$

where

$$\widetilde{\Phi}'^0_M(u,v) = \overline{\widetilde{\Phi}^0_{M,0,0}(-u,-v)}, \quad \widetilde{\Psi}'^\tau_j(u,v) = \overline{\widetilde{\Psi}^\tau_{j,0,0}(-u,-v)},$$

and $\overline{z}$ *denotes the complex conjugate of* z.

By the Fourier slice theorem (1), $\mathcal{F}_2 \mathcal{P}_{\boldsymbol{\theta}} f(\omega_u, \omega_v) = \mathcal{F}_3 f(\omega_u \boldsymbol{u} + \omega_v \boldsymbol{v})$. Therefore, the wavelet coefficients at scale j in (4) can be computed by multiplying a slice of the 3-D Fourier transform of f by the 2-D Fourier transform of the scaling or wavelet function at scale j, followed by an inverse 2-D Fourier transform evaluated at the points of the form $(2^j k, 2^j l)$ in the view plane. We now turn to a description of the actual implementation.

2.3.2. *The wavelet transform in the Fourier domain*
The wavelet transform and its inverse consist of up- or downsampling and convolution.

2.3.2.1. *Up- and downsampling* Let $X_{k,l}$, $k = 0, \ldots, N_1 - 1$, $l = 0, \ldots, N_2 - 1$, denote the elements of the 2-D discrete Fourier transform (DFT) of a 2-D signal x of length N_1 by N_2, with N_1 and N_2 both even.

Downsampling corresponds to taking the samples with even index in both dimensions. By applying a *biphase decomposition* [18, 23], one obtains that the values $X^{\text{down}}_{k,l}$, $k = 0, \ldots, N_1/2-1$, $l = 0, \ldots, N_2/2-1$ of the 2-D DFT of the downsampled signal are given by

$$X^{\text{down}}_{k,l} = \frac{1}{4}(X_{k,l} + X_{k-\frac{N_1}{2},l} + X_{k,l-\frac{N_2}{2}} + X_{k-\frac{N_1}{2},l-\frac{N_2}{2}}). \tag{7}$$

Let X be the matrix whose elements are the Fourier coefficients $X_{k,l}$ of x, and X^{down} the corresponding matrix for the downsampled signal. Then (7) can be written in matrix notation as

$$\mathsf{X}^{\text{down}} = \frac{1}{4}(\mathsf{X}_a + \mathsf{X}_b + \mathsf{X}_c + \mathsf{X}_d) \quad \text{when} \quad \mathsf{X} = \begin{pmatrix} \mathsf{X}_a & \mathsf{X}_b \\ \mathsf{X}_c & \mathsf{X}_d \end{pmatrix}, \tag{8}$$

where $\mathsf{X}_a, \mathsf{X}_b, \mathsf{X}_c, \mathsf{X}_d$ are the four submatrices obtained by equally dividing X into two along the row and column direction.

Conversely, upsampling by a factor of two in the spatial domain means inserting zeros between the samples in both dimensions. One easily derives the following relation between the matrix X of Fourier coefficients of the original signal, and the corresponding matrix X^{up} of the upsampled signal [26]:

$$\mathsf{X}^{\text{up}} = \begin{pmatrix} \mathsf{X} & \mathsf{X} \\ \mathsf{X} & \mathsf{X} \end{pmatrix} \tag{9}$$

So, the DFT matrix X^{up} of the upsampled signal is obtained by replicating the matrix X in both dimensions.

2.3.2.2. *Wavelet decomposition and reconstruction* Let the input of the wavelet transform be a finite 2-D input sequence, represented by an array c^0 of size $N_1 \times N_2$. Let c^j and $d^{j,\tau}$ denote 2-D sequences of approximation coefficients $c^j_{k,l}$ and detail coefficients $d^{j,\tau}_{k,l}$, respectively, cf. (2). Denote by C^j and $D^{j,\tau}$ the corresponding matrices of Fourier coefficients, obtained by applying a 2-D DFT to c^j and $d^{j,\tau}$, respectively.

Define 2-D filter matrices H^j and $\mathsf{G}^{j,\tau}$, $\tau = 1, 2, 3$, by

$$(\mathsf{H}^j)_{k,l} = H^j_k\, H^j_l, \quad (\mathsf{G}^{j,1})_{k,l} = H^j_k\, G^j_l,$$
$$(\mathsf{G}^{j,2})_{k,l} = G^j_k\, H^j_l, \quad (\mathsf{G}^{j,3})_{k,l} = G^j_k\, G^j_l.$$

Here H^j_k and G^j_k are the DFT values of the 1-D synthesis filters h and g. For example, if the signal length in a given spatial direction is $2^{-j}N$ (assumed to be larger than the length L of the filter h), then

$$H^j_k = \sum_{n=0}^{L-1} h_n e^{-\frac{2\pi i n k 2^j}{N}} = H^0_{k2^j}, \quad k = 0, 1, \ldots, 2^{-j}N - 1. \tag{10}$$

Dual filter matrices $\widetilde{\mathsf{H}}^j$ and $\widetilde{\mathsf{G}}^j$ are defined in a similar way in terms of the dual filters $\widetilde{h}$ and $\widetilde{g}$. Note from (10) that it is sufficient to compute the filters H^0 and G^0. The filters for the other scales are obtained by downsampling the filters for the finest scale $j = 0$. A similar remark holds for the analysis filters.

In the frequency domain, the wavelet decomposition has the matrix representation [26]

$$C^{j+1} = [\widetilde{\mathsf{H}}^j \bullet C^j]^{\mathrm{down}}, \quad D^{j+1,\tau} = [\widetilde{\mathsf{G}}^{j,\tau} \bullet C^j]^{\mathrm{down}}, \tag{11}$$

where $A \bullet B$ denotes elementwise multiplication of matrices A and B, and $[\ldots]^{\mathrm{down}}$ is defined as in (8).

Wavelet reconstruction has the following matrix representation in the frequency domain [26]:

$$C^j = \mathsf{H}^j \bullet [C^{j+1}]^{\mathrm{up}} + \sum_{\tau=1}^{3} \mathsf{G}^{j,\tau} \bullet [D^{j+1,\tau}]^{\mathrm{up}}, \tag{12}$$

where $[\ldots]^{\mathrm{up}}$ is defined as in (9).

We will refer to (11) and (12) as *Fourier-wavelet decomposition* (FWD) and *Fourier-wavelet reconstruction* (FWR), respectively. The result of an M-level decomposition yields an approximation array C^M of size $2^{-M}N_1 \times 2^{-M}N_2$, and detail arrays $D^{j,\tau}$, $j = M, M-1, \ldots, 1$, $\tau = 1, 2, 3$, of size $2^{-j}N_1 \times 2^{-j}N_2$. A reconstruction at a desired level K is first computed in the Fourier domain by (12) and the resulting approximation C^K is then inversely Fourier transformed to give the desired approximation c^K in the spatial domain.

As shown in [26], the complexity of the Fourier domain implementation of the 2-D wavelet transform (or its inverse) is $\mathcal{O}(N^2 \log_2 N)$, when $N_1 = N_2 = N$.

2.3.3. *The Fourier-wavelet volume rendering algorithm*

The wavelet extension of FVR requires only a small modification of the standard algorithm. The resulting algorithm, referred to as Fourier-wavelet volume rendering (FWVR), is summarized as follows.

ALGORITHM 2. **FWVR**

- Preprocessing. *Compute the 3-D FFT of the volume data (size N^3).*
- Actual volume rendering. *For each direction $\boldsymbol{\theta}$, do:*
  1. *Interpolate the Fourier transform on a regular grid of size $(2N)^2$ in the slice plane orthogonal to $\boldsymbol{\theta}$. This yields the array C^0 to be used for initializing the wavelet transform.*

2. *Perform a 2-D Fourier-wavelet decomposition (FWD) of depth M, yielding approximation coefficients $C^M_{k,l}$ and detail coefficients $D^{j,\tau}_{k,l}$, where $j = M, M-1, \ldots, 1$, respectively.*

3. *Perform a partial Fourier-wavelet reconstruction (FWR) from $C^M_{k,l}$, by putting all detail signals $D^{j,\tau}_{k,l}$ equal to zero, followed by a 2-D inverse Fourier transform, yielding an initial approximation (size $(2N)^2$) in the spatial domain.*

4. *Refine the approximation by partial FWR using the detail signals $D^{j,\tau}_{k,l}$ with $K < j \leq M$, followed by a 2-D inverse Fourier transform to obtain an approximation (size $(2N)^2$) at a finer scale K in the spatial domain.*

In order to prevent aliasing, the sampling step size of the slice plane should be sufficiently small. If the original step size of the 3-D Fourier transform of the volume data is F_0, resampling should be done with a step size of at most $F_0/\sqrt{3}$ [15]. The scale factor $\sqrt{3}$ originates from the length of the diagonal of a unit cube. In practice, one usually takes $F_0/2$, giving rise to a resampling grid of size $(2N)^2$.

The choice of the decomposition depth M depends on the desired level of detail for the low resolution images, the size of the data, and the length of the wavelet filters. Typically, we take $M = 2$ or $M = 3$ for datasets of size $128 \times 128 \times 128$ or $256 \times 256 \times 256$. Larger M blurs the low resolution images too much, making interpretation difficult.

The approach taken in Algorithm 2 is well-suited to implement a *client-server* visualization system. The server performs the initial 3-D Fourier transform, slicing, and FWD at each view angle (steps 1 and 2), and sends the required approximation/detail coefficients to the client. The client performs the FWR and inverse Fourier transform to obtain an approximation image (steps 3 and 4). During user interaction, only the Fourier domain approximation coefficients $C^M_{k,l}$ are used. When user interaction ceases, the Fourier domain detail coefficients $D^{j,\tau}_{k,l}$ are taken into account, so that the client can obtain reconstructions at higher levels of detail. This progressive refinement can be implemented most efficiently by a so-called *non-pyramidal* reconstruction scheme, which involves upsampling of the wavelet coefficients to full resolution, followed by application of a precomputed filter which combines the effect of all intermediate resolution filters into a single one [26]. The coefficients can be quantized to shorts (2 bytes), with a quantization error in the order of 10^{-8}, without introducing visible artefacts. The progressive refinement inherent in the algorithm can improve interaction with the data, since the response time of the system drops significantly.

3. View Interpolation in the Fourier-Wavelet Domain

In this section, we introduce a technique similar to view interpolation [1] for Fourier-wavelet volume rendering. The differences are that (i) interpolation is performed on the wavelet approximation coefficients in the frequency domain and not in the image domain, and (ii) interpolation is performed during user interaction only. By user interaction, we mean rotation of the view vector $\boldsymbol{\theta}$. When user interaction ceases, ordinary FWVR is applied to render an accurate image. This relaxes the accuracy requirements imposed on the interpolation method used during interaction.

A view vector $\boldsymbol{\theta}$ is determined by spherical coordinates (θ, ϕ), where $0 \leq \theta \leq \pi$ and $0 \leq \phi < 2\pi$. For the X-ray transform, we can restrict θ to $0 \leq \theta \leq \frac{\pi}{2}$, since we can obtain images for $\frac{\pi}{2} < \theta \leq \pi$ by mirroring the respective images. View interpolation requires an appropriate sampling of the parameters θ and ϕ. Here, a compromise has to be found between memory requirements and image quality. A large number of precomputed viewing directions produces high quality images for intermediate viewing directions, but suffers from high memory costs. On the other hand, too few precomputed views result in low quality images which suffer from extreme blurring. Regular angular sampling of θ and ϕ is sub-optimal for traditional view interpolation, since the sampling densities are not uniformly spread. For instance, with θ close to zero, the sampling density is very high, i.e. the neighbouring views are close to each other, whereas the density is much lower for $\theta = \frac{\pi}{2}$. For FWVR, the situation is different. A view corresponds to a slice passing through the origin in Fourier space, which means that the sampling density decreases away from the origin. Since interpolation is performed in the Fourier domain, this means that low frequencies are sampled at a higher rate than high frequencies. Therefore, the only effect of regular angular sampling is loss of detail if the sampling rate is too low. This is not really a problem, because interpolation is performed on the wavelet approximation coefficients, which contain mainly low frequencies. Moreover, view interpolation will be performed only during user interaction, when low resolution views are taken from the data by rotating $\boldsymbol{\theta}$, and the human eye is not very sensitive for loss of detail when motion is involved.

In the following, we take a number of N_θ values for θ and N_ϕ values for ϕ, respectively, resulting in a total of $N_\theta \times N_\phi$ precomputed slices. The view vector $\boldsymbol{\theta}$ is denoted by (θ_i, ϕ_j), where

$$\theta_i = \frac{i\pi}{2(N_\theta - 1)} \text{ and } \phi_j = \frac{j2\pi}{N_\phi}, \qquad 0 \leq i < N_\theta,\ 0 \leq j < N_\phi.$$

Recall that $\boldsymbol{u}$ and $\boldsymbol{v}$ defining the view plane are two mutually orthogonal vectors perpendicular to $\boldsymbol{\theta}$. We require $\boldsymbol{u}$ to be in the x-y plane, so that it depends on ϕ only. The vector $\boldsymbol{v}$ is then fixed by taking $\boldsymbol{v} = \boldsymbol{u} \times \boldsymbol{\theta}$, resulting in a right-handed coordinate system. For our application, view interpolation is performed on the wavelet approximation coefficients only. These precomputed coefficients are denoted by $\mathsf{C}^M_{k,l}(\theta_i, \phi_j)$.

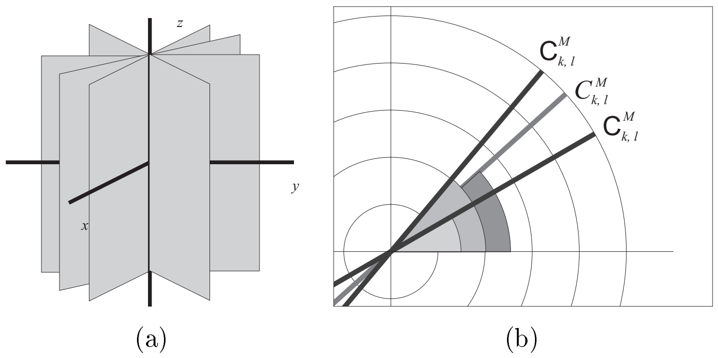


Figure 2. (a) Precomputed slices in Fourier space for fixed $\theta = \frac{\pi}{2}$ and $N_\phi = 8$. (b) Interpolation of the precomputed slices $\mathsf{C}^M_{k,l}(\theta, \phi_1)$ and $\mathsf{C}^M_{k,l}(\theta, \phi_2)$. The resulting approximation coefficients are $C^M_{k,l}(\theta, \phi)$.

For a chosen viewing direction (θ, ϕ), $0 \leq \theta \leq \frac{\pi}{2}$, and $0 \leq \phi < 2\pi$, slice interpolation is performed as follows. Find i and j, such that $\theta_i \leq \theta < \theta_{i+1}$ and $\phi_j \leq \phi < \phi_{j+1}$. Then, compute the interpolated approximation coefficients $C^M_{k,l}(\theta, \phi)$ by bilinear interpolation:

$$\begin{aligned} C^M_{k,l}(\theta,\phi) \;&=\; (1-\alpha)\,(1-\beta)\,\mathsf{C}^M_{k,l}(\theta_i,\phi_j) \;+\; (1-\alpha)\,\beta\,\mathsf{C}^M_{k,l}(\theta_i,\phi_{j+1}) \\ &\quad+\; \alpha\,(1-\beta)\,\mathsf{C}^M_{k,l}(\theta_{i+1},\phi_j) \;+\; \alpha\,\beta\,\mathsf{C}^M_{k,l}(\theta_{i+1},\phi_{j+1}), \end{aligned}$$

where

$$\alpha = \frac{\theta - \theta_i}{\theta_{i+1} - \theta_i}, \text{ and } \beta = \frac{\phi - \phi_j}{\phi_{j+1} - \phi_j}.$$

Viewing directions for which $\frac{\pi}{2} < \theta < \pi$ require mirroring of the final images. Figure 2 illustrates the meaning of the parameters for the special case that $\boldsymbol{\theta}$ lies in the x-y plane and is rotating around the z-axis. In this case, $\theta = \frac{\pi}{2} = \theta_{N_\theta - 1}$ and, therefore, $\alpha = 0$. Figure 2(a) shows the situation in 3-D. The vertical planes represent the precomputed slices for $N_\phi = 8$. Figure 2(b) shows the scene projected on the ω_x-ω_y plane. Two precomputed slices are shown in black, and the interpolated slice is shown in grey.

The extension of Fourier-wavelet volume rendering (Algorithm 2) with view interpolation is now as follows.

ALGORITHM 3. **FWVR with view interpolation**

- Preprocessing. *Compute the 3-D Fourier transform of the volume data, and compute a set* $\mathsf{C}^M_{k,l}(\theta_i, \phi_j)$ *of wavelet approximation coefficients in frequency space for a number of different viewing directions* (θ_i, ϕ_j), $0 \le i < N_\theta$, $0 \le j < N_\phi$.

- Actual volume rendering. *For each direction* $\boldsymbol{\theta}$ *do:*

  1. *Interpolate the precomputed coefficients* $\mathsf{C}^M_{k,l}(\theta_i, \phi_j)$ *in the slice plane orthogonal to* $\boldsymbol{\theta}$*. This yields the array* $C^M_{k,l}(\boldsymbol{\theta})$.

  2. *Perform a partial Fourier-wavelet reconstruction from* $C^M_{k,l}(\boldsymbol{\theta})$*, followed by a 2-D inverse Fourier transform to obtain an approximation in the spatial domain.*

4. Experimental Results

Experiments with two CT data sets were carried out to assess quality and performance of the proposed algorithm. We used a small CT data set of size 128^3 and a large CT data set of size 256^3. A fourth-order B-spline wavelet was used as the basic wavelet. An important property of FWVR is that other wavelets give only marginally different timing results [26]. For the small data set, we used two decomposition levels and for the large data set three decomposition levels, so that the size of the precomputed sequence of approximation coefficients is the same for both data sets. The decomposition depth M cannot be set larger for the fourth-order B-spline wavelets, because the size of the downsampled data should always be larger than the length of the filters used for the wavelet decomposition (41 coefficients). Cubic B-spline interpolation [21] with 20% zero-padding was applied for resampling slices in Fourier space.

Figure 3 shows plots of relative error norms of the difference between an approximation image obtained by view interpolation (Algorithm 3) and by direct computation by FWVR (Algorithm 2) for the small CT data set. Angle increments of 5 degrees and 10 degrees were used for the view vector $\boldsymbol{\theta}$ rotating around the z-axis, i.e. $\theta = \frac{\pi}{2}$. Figure 3(a) shows a plot of the L_∞ norm (absolute difference), and Fig. 3(b) shows a plot of the L_2 norm (mean squared difference). The plots show that the relative L_2 norm is very small for both 5 degree and 10 degree angle increments. On the whole, it is less than one grey value. On average, the relative L_2 norm for 5 degree angle increments is a factor of 9.5 smaller than for 10 degree angle increments. The L_∞ norm may be large at certain view angles when using 10 degree angle increments. For example, expressed in grey values, it can be as large as 32. On average, the L_∞ norm for 5 degree angle increments is a factor of 2.9 smaller than for 10 degree angle increments.

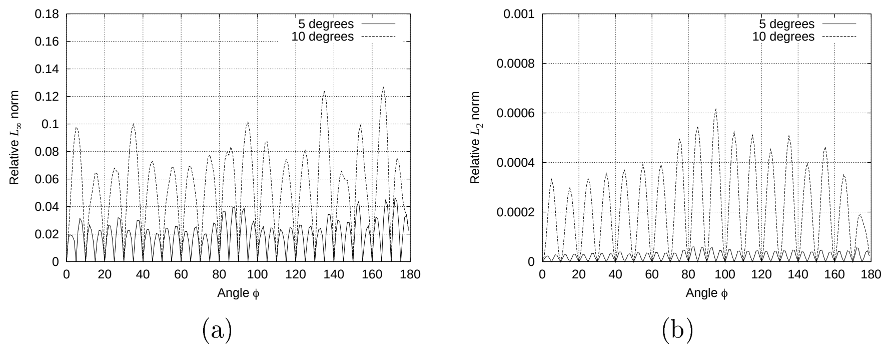


Figure 3. Plots of relative L_∞ (a) and L_2 (b) norms of the difference between an approximation image obtained by view interpolation and by direct computation. Angle increments of 5 degrees and 10 degrees were used.

Although the L_∞ norm may be large for 10 degree angle increments, the approximation images still look acceptable. The problem with the L_∞ norm is that it takes the maximum absolute difference over the whole image. To obtain a better impression of the *amount* of pixels that deviate from the exact value, we can look at the histograms of the absolute differences between images obtained by view interpolation and by direct computation. This is done for the viewing direction $\boldsymbol{\theta} = (\frac{\pi}{2}, \frac{166\pi}{180})$, for which the L_∞ error is maximal. The cumulative histograms are shown in Fig. 4(a) and Fig. 4(b) for 5 degree angle increments and 10 degree angle increments, respectively. The histograms show that, in spite of a large L_∞ error at 10 degree angle increments, 93% of the pixels are within an error margin of 5 grey values. For 5 degree angle increments this number is 98%. A difference within a range of 5 grey values is so small that the human eye cannot distinguish it, especially not in bright areas.

Figure 5(a) shows an exact image obtained by direct computation, and Fig. 5(b)-(c) show difference images obtained by subtracting the exact image from images obtained by view interpolation using 5 degree angle increments and 10 degree angle increments, respectively. The grey values of the difference images were scaled to show better contrast, where white corresponds to a positive difference and black to a negative difference. The view vector is $(\frac{\pi}{2}, \frac{7\pi}{180})$. This viewing direction was chosen because the L_∞ norms are large for both 5 degree angle increments

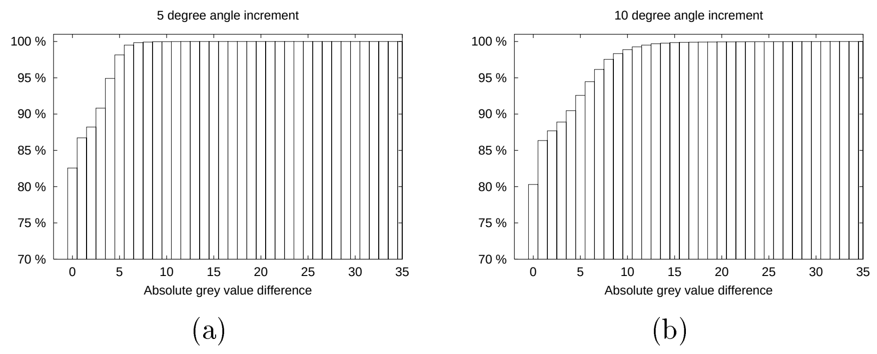


Figure 4. Cumulative histograms of the absolute difference between images obtained by direction computation and view interpolation. (a) 5 degree angle increments. (b) 10 degree angle increments.

and 10 degree angle increments. The images in Fig. 5(b)-(c) show that differences are small and distributed uniformly over the image, resulting in a slight blurring. This effect is only visible in still images, and we want to emphasize that these approximation images are shown *only* during user interaction, when a user chooses new viewing directions several times per second. Since the human eye is less sensitive for loss of detail in images involving motion, the blurring is not a problem.

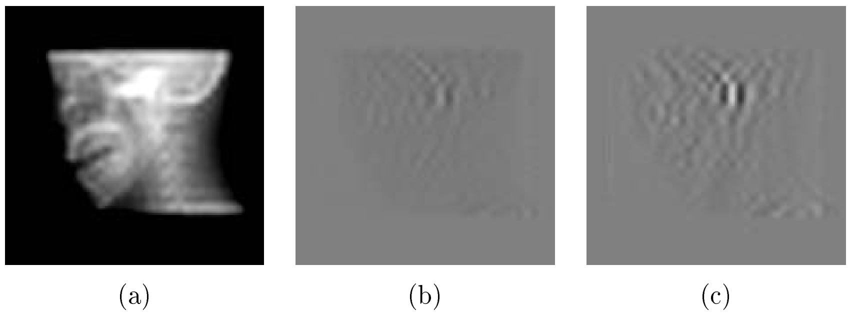
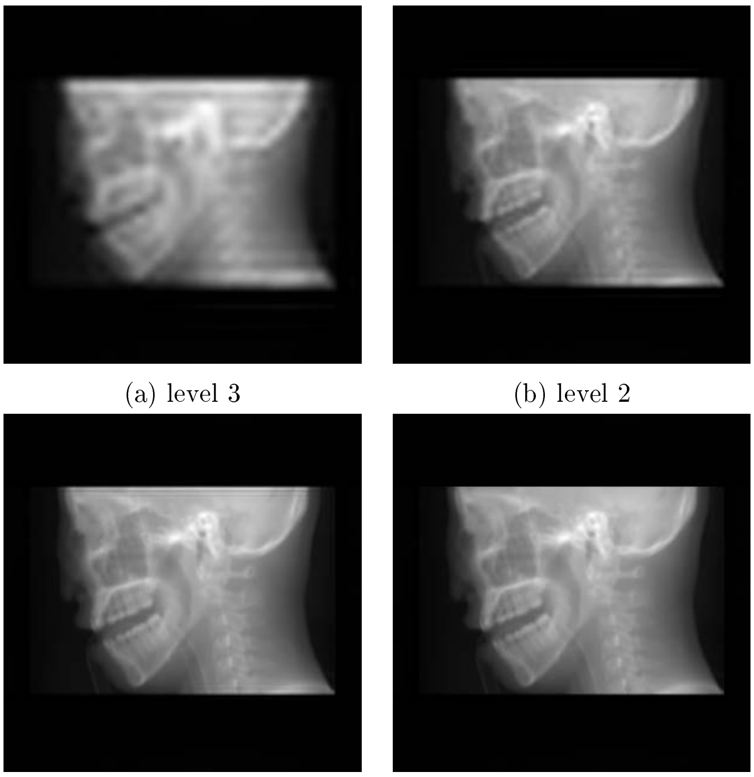


Figure 5. (a) Exact approximation image for the view vector $(\frac{\pi}{2}, \frac{7\pi}{180})$ obtained by direct computation. (b)-(c) Difference images obtained by subtracting the exact image from images obtained by view interpolation with 5 degree angle increments (b) and 10 degree angle increments (c).

When user interaction ceases, we apply Algorithm 2 to refine the images incrementally to full resolution as shown in Fig. 6 for the large CT data set. The level 1 approximation (Fig. 6(c)) uses only 25% of the wavelet coefficients, yet differences with the full reconstruction

(Fig. 6(d)) are hardly distinguishable, providing an extra motivation for the use of wavelets.


(a) level 3 (b) level 2

(c) level 1 (d) level 0

Figure 6. FWVR (Algorithm 2) rendering by a three-level fourth-order B-spline wavelet decomposition of the large CT data set.

Table I shows rendering times of FWVR with view interpolation (Algorithm 3) and cumulative rendering times of ordinary FWVR (Algorithm 2). Timings were performed on a Pentium III 500 MHz processor. All results include the time used by the inverse 2-D FFT. While a user is interacting with the data, FWVR with view interpolation (Algorithm 3) is performed. The results show that this allows for fast interaction; for a volume of size 256^3 the method renders at 5.6 frames per second, whereas ordinary FWVR manages only 0.7 frames per second (computed from the table entry corresponding to the time to obtain a level 3 approximation). When interaction ceases, a slice is extracted from the 3-D Fourier transform of the data, in order to render an exact image for that viewing direction, which is obtained after slightly more than 2 seconds by ordinary FWVR (Algorithm 2).

Table I. Rendering times (in seconds) of FWVR extended by view interpolation (Alg. 3) and ordinary FWVR (Alg. 2). During user interaction, low resolution images are computed by FWVR with view interpolation. When interaction ceases, an exact image is computed by ordinary FWVR. The table entries for Alg. 2 are cumulative.

	CT head (128^3)	CT head (256^3)
User interaction — (Alg. 3)		
FWVR with view interpolation	0.04	0.18
User interaction ceased — (Alg. 2)		
Slice extraction	0.25	1.05
Fourier-wavelet decomposition	0.32	1.35
Level 3 approximation		1.45
Level 2 approximation	0.34	1.63
Level 1 approximation	0.37	1.44
Full reconstruction	0.42	2.14

5. Discussion

Fourier-wavelet volume rendering is a computationally efficient method to visualize data at progressively higher levels of detail, which is useful in client-server systems. In this paper, we have overcome one of the disadvantages of FWVR, i.e. the need to interpolate a slice in Fourier space at full resolution in order to perform a 2-D wavelet decomposition. This was accomplished by precomputing sets of wavelet approximation coefficients in the Fourier domain for a set of selected fixed viewing directions. The new algorithm computes images for intermediate viewing directions by interpolation of the precomputed coefficients. The main differences between ordinary view interpolation (as used in computer graphics) and view interpolation in Fourier-wavelet space are that (i) interpolation is performed on the wavelet approximation coefficients in the frequency domain and not in the image domain, and (ii) interpolation is performed during user interaction only.

We have used simple bilinear interpolation for view interpolation. This was done for two reasons: (i) it is computationally more efficient than higher order interpolation methods, and (ii) higher order interpolation methods give only marginally different results. Since view interpolation is applied *only* during user interaction, we consider speed to be more important than accuracy. Furthermore, the results show that bilinear interpolation gives acceptable errors, and angle increments as large as 10 degrees result in only a small degradation of image quality.

The computational cost of view interpolation is independent of the angle increments, and changing these only affects the precomputation stage. If the angle increments are made smaller, it takes more time to precompute the approximation coefficients, and also storage space requirements increase. However, rendering always takes the same amount of time, but image quality increases when smaller angle increments are used.

References

1. Chen, S. E. and L. Williams: 1992, 'View Interpolation for Image Synthesis'. In: *Computer Graphics (Proceedings of SIGGRAPH 92)*. pp. 279–288.
2. Choi, J. and Y. G. Shin: 1999, 'Efficient Image-Based Rendering of Volume Data'. *Journal of KISS* **26**(3), 261–270.
3. Chui, C. K.: 1992, *An Introduction to Wavelets*. Academic Press.
4. Drebin, R. A., L. Carpenter, and P. Hanrahan: 1988, 'Volume Rendering'. *Computer Graphics (SIGGRAPH '88 proceedings)* **22**(4), 65–74.
5. Frigo, M. and S. G. Johnson: 1998, 'FFTW: An Adaptive Software Architecture for the FFT'. In: *Proc. ICASSP*, Vol. 3. p. 1381.
6. Horbelt, S., M. Unser, and M. Vetterli: 1999, 'Wavelet Projections for Volume Rendering'. In: M. A. Alberti, G. Gallo, and I. Jelinek (eds.): *Eurographics '99*. pp. 56–59.
7. Kak, A. C. and M. Slaney: 1988, *Principles of Computerized Tomographic Imaging*. New York: IEEE Press.
8. Keys, R. G.: 1981, 'Cubic Convolution Interpolation for Digital Image Processing'. *IEEE Transactions on Acoustics, Speech, and Signal Processing* **ASSP-29**(6), 1153–1160.
9. Lacroute, P. and M. Levoy: 1994, 'Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transformation'. In: A. Glassner (ed.): *Proceedings of SIGGRAPH '94*. pp. 451–458.
10. Levoy, M.: 1988, 'Volume Rendering: Display of surfaces from volume data'. *IEEE Computer Graphics and Applications* **8**(3), 29–37.
11. Lichtenbelt, B., R. Crane, and S. Naqvi: 1998, *Introduction to Volume Rendering*. Hewlett-Packard Professional Books, Prentice-Hall.
12. Lippert, L., M. H. Gross, and C. Kurmann: 1997, 'Compression Domain Volume Rendering for Distributed Environments'. *Computer Graphics Forum* **16**(3), 95–107.
13. Lorensen, W. E. and H. Cline: 1987, 'Marching Cubes: A High Resolution 3D Surface Construction Algorithm'. *Computer Graphics* **21**(4), 163–169.
14. Mallat, S. G.: 1989, 'A theory for multiresolution signal decomposition: the wavelet representation'. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **11**, 674–693.
15. Malzbender, T.: 1993, 'Fourier Volume Rendering'. *ACM Transactions on Graphics* **12**(3), 233–250.
16. Napel, S., S. Dunne, and B. K. Rutt: 1991, 'Fast Fourier Projection for MR Angiography'. *Magnetic Resonance in Medicine* **19**, 393–405.

17. Natterer, F.: 1986, *The Mathematics of Computerized Tomography.* B. G. Teubner & J. Wiley.
18. Rioul, O. and P. Duhamel: 1992, 'Fast Algorithms for Discrete and Continuous Wavelet Transforms'. *IEEE Transactions on Information Theory* **38**(2), 569–586.
19. Roerdink, J. B. T. M. and M. A. Westenberg: 1999, 'Wavelet-Based Volume Visualization'. *Nieuw Archief voor Wiskunde* **17**(2), 149–158.
20. Totsuka, T. and M. Levoy: 1993, 'Frequency Domain Volume Rendering'. In: J. Kajiya (ed.): *Computer Graphics (SIGGRAPH '93 Proceedings)*, Vol. 27. pp. 271–278.
21. Unser, M.: 1999, 'Splines: a Perfect Fit for Signal and Image Processing'. *IEEE Signal Processing Magazine* **16**(6), 22–28.
22. Unser, M., A. Aldroubi, and M. Eden: 1993, 'A Family of Polynomial Spline Wavelet Transforms'. *Signal Processing* **30**, 141–162.
23. Vetterli, M. and C. Herley: 1992, 'Wavelets and Filter Banks: Theory and Design'. *IEEE Transactions on Signal Processing* **40**(9), 2207–2232.
24. Vetterli, M. and J. Kovačević: 1995, *Wavelets and Subband Coding.* Prentice-Hall.
25. Warrick, A. L. and P. A. Delaney: 1995, 'A wavelet localized Radon transform'. In: *Proc. SPIE Vol. 2569, Wavelet Applications in Signal and Image Processing III, Andrew F. Laine; Michael A. Unser; Mladen V. Wickerhauser; Eds.* pp. 632–643.
26. Westenberg, M. A. and J. B. T. M. Roerdink: 2000a, 'Frequency Domain Volume Rendering by the Wavelet X-Ray Transform'. *IEEE Transactions on Image Processing* **9**(7), 1249–1261.
27. Westenberg, M. A. and J. B. T. M. Roerdink: 2000b, 'X-Ray Volume Rendering Through Two-Stage Splatting'. *Machine Graphics & Vision* **9**(1/2), 307–314.
28. Westover, L. A.: 1990, 'Footprint Evaluation for Volume Rendering'. *Computer Graphics* **24**(4), 367–376.
29. Zuidwijk, R. A.: 1997, 'The Wavelet X-ray Transform'. Technical Report PNA-R9703, Centre for Mathematics and Computer Science, Amsterdam.