Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

Institute for Information Systems

Computer Graphics Research Group

Internal Report # 247

Two Methods for Wavelet-Based Volume Rendering

M. H. Gross, L. Lippert, R. Dittrich, S. Häring

Computer Science Department
ETH-Zürich, Switzerland
E-Mail: grossm@inf.ethz.ch, lippert@inf.ethz.ch
http://www.inf.ethz.ch/department/IS/cg/

Two Methods for Wavelet-Based Volume Rendering

Comp. Sc. Dept. internal report no. 247

M. H. Gross, L. Lippert, R. Dittrich, S. Häring
Computer Science Department
ETH-Zürich, Switzerland
E-Mail: grossm@inf.ethz.ch, lippert@inf.ethz.ch

Abstract - This paper describes two methods for wavelet domain rendering of volume data sets, which base on entire different paradigms for the approximation of the underlying volume rendering equation. The first approach comprises a ray-tracing algorithm for cardinal B-spline wavelet representations of both color and opacity values and carries out high quality images. We especially propose elaborated integration and evaluation schemes, where the computation of shading is also performed in wavelet space. The second approach proposes a fast intensity integration method based on wavelet splats. This scheme takes advantage from the self-similarity of the basis functions and computes the required splats by Fourier projection slicing. At this point, it unifies the advantages coming along with Fourier domain and with wavelet domain rendering and it avoids some of the major artifacts the first one is prone to. Due to the computational efficiency, the splatting method allows to compute coarse approximations at interactive rates and refines the picture progressively.

1 Introduction

Volume Rendering is still a challenging and important subfield in scientific visualization, since volume data sets of any type arise in many different applications. A general formulation of the volume rendering problem was provided by [7] or [14], who describe the resulting image essentially as a solution of an integral equation, the so-called *volume rendering equation*. In most cases, however, low-albedo approximations of the solution are sufficient and can be computed using one of the various ray-tracing schemes introduced in the past, such as [3] or [18]. Unfortunately, these types of image order methods are still computationally expensive and, hence, are not very well shaped for real-time and interactive applications.

In contrast to the image order approaches, object order algorithms, such as splatting [9], [11] try to accelerate the rendering process by harvesting the spatial structure of volume data sets. Recent approaches also include hybrid rendering methods [8], which link both paradigms. Other fundamental techniques are given with various types of transform domain volume rendering, where Fourier spaces have been explored succesfully [13], [19]. Unfortunately, beside of some computational advantages especially Fourier based methods are prone to major shortcomings regarding sampling, progressivity, filtering, local level of detail or transfer functions.

Hence, the wavelet transform has been stressed for volume rendering, since it seems to be a very promising tool to overcome some of the major problems. But in spite of different approaches, such as [6], [15], [20], there have not yet been fundamental quantum leaps in computational efficiency.

Our research was motivated by providing an overall framework for wavelet domain volume rendering to carry out approximations of the volume rendering equation at different levels of accuracy. This enables one to harvest all advantages coming along with wavelet representations, such as error bounds, localization, progressive refinement, smooth approximations, level-of-detail, feature enhancement etc. Therefore, we propose two essentially different methods for wavelet domain volume rendering. The first one, an image order algorithm, ray-traces volumes entirely in wavelet space including exponential transfer functions and illumination. It features high quality images and can be considered as a reference method. The second approach introduced in this paper encompasses an object oriented splatting algorithm. Some of its initial ideas were scetched in [11] and especially employ Fourier descriptions of wavelet basis functions to efficiently compute line integrals. As a consequence, the method performs rendering at nearly interactive rates including all advantageous features of the WT. In particular, this approach is well shaped for networked and remote rendering. Both methods base on cardinal B-spline wavelets, but are not restricted to them.

The paper is structured as follows: Due to the rich literature on wavelets in computer graphics, we preassume the reader to be familiar with the fundamentals of orthogonal and semi-orthogonal WTs and restrict our description in section 2 to the approximations of the volume rendering equation we base on. Section 3 presents the ray-tracing approach, where particular emphasis is given to the computation of color and alpha supported by efficient data structures. In Section 4 we explain the second algorithm, a wavelet based splatting and its essential mathematical and algorithmic prerequisities. Finally, Section 5 given various examples for both methods.

2 Volume Rendering in Wavelet Domain

The objective of the following section is to briefly summarize the mathematical foundations of volume rendering and of the wavelet approximations of the underlying integral equation.

2.1 The Volume Rendering Integral

The most general representation of the volume rendering problem is provided by following equation [7], which is derived from particle physics and regulates the transport of light in participating media. Here, the intensity $I(\boldsymbol{x},\boldsymbol{s})$ at a location $\boldsymbol{x}$ transported into direction $\boldsymbol{s}$ is computed as:

$$I(\boldsymbol{x},\boldsymbol{s}) = I_{in}e^{-\int_0^R \sigma(\boldsymbol{x}_{in} + R'\boldsymbol{s})dR'} + \int_0^R e^{-\int_{R'}^R \sigma(\boldsymbol{x}_{in} + R''\boldsymbol{s})dR''} Q(\boldsymbol{x}_{in} + R'\boldsymbol{s},\boldsymbol{s})dR' \tag{1}$$

where $Q(\boldsymbol{x},\boldsymbol{s})$ denotes the generalized source term, which itself can be decomposed into:

$$Q(\boldsymbol{x},\boldsymbol{s}) = q(\boldsymbol{x}) + \sigma_s(\boldsymbol{x})\int p(\boldsymbol{x},\boldsymbol{s}' \rightarrow \boldsymbol{s})I(\boldsymbol{x},\boldsymbol{s})d\boldsymbol{s}' \tag{2}$$

Here σ denotes a *total interaction coefficient* that represents the probability that a particle will suffer absorbtion $(\sigma_a(\boldsymbol{x}))$ or scattering$(\sigma_s(\boldsymbol{x}))$, per unit distance traveled:

$$\sigma := \sigma_a + \sigma_s \tag{3}$$

The source term $q(\boldsymbol{x})$ represents the number of particles created per unit volume. We can associate this term for instance with a local illumination model defined as:

$$q(\boldsymbol{x}) := q_{ambient}(\boldsymbol{x}) + q_{diffuse}(\boldsymbol{x})\sum_i max(\boldsymbol{n}(\boldsymbol{x}) \cdot \boldsymbol{Light_i}, 0) + q_{specular}(\boldsymbol{x})\sum_i max(\boldsymbol{n}(\boldsymbol{x}) \cdot \boldsymbol{Halfway_i}, 0)^{specularity} \tag{4}$$

where $\boldsymbol{n}(\boldsymbol{x})$ denotes the normal vector at $\boldsymbol{x}$. The scattering phase function $p(\boldsymbol{x},\boldsymbol{s}' \rightarrow \boldsymbol{s})$ describes the flow of particles at a given point $\boldsymbol{x}$ from a direction $\boldsymbol{s}'$ into $\boldsymbol{s}$. In many classical volume rendering methods [10], only low albedo approximations of the upper integral equation are computed, which are based on:

$$I(t_0, t_L) = \int_{t_0}^{t_L} q(\boldsymbol{x}_{in} + t \cdot \boldsymbol{s})e^{-\int_{t_0}^{t} \alpha(u)du} dt \tag{5}$$

For the opacity term we set

$$\alpha(u) = \sigma(\boldsymbol{x}_{in} + u \cdot \boldsymbol{s}) \tag{6}$$

If the opacity term α is set to 0, equation (5) collapses to a line integration procedure, which computes an image of the volume intensity function q along a given ray $\boldsymbol{x}_{in} + t \cdot \boldsymbol{s}$.

$$I(t_0, t_L) = \int_{t_0}^{t_L} q(\boldsymbol{x}_{in} + t \cdot \boldsymbol{s})dt \tag{7}$$

It is clear, that eq. (7) can be computed much more efficiently but due to the restriction of the transfer function the resulting image quality is X-ray like and inferior. The two methods proposed in the following sections target at numeric approximations of eq. (5) and of eq. (7), respectively.

2.2 Wavelet Approximations of Volume Data

For a discrete volume data set, sampled on an equally spaced 3D grid, we denote by $c^{q_{RGB}}_{m_0pqr}$ the volume color samples and by $c^{\alpha}_{m_0pqr}$ the opacity samples. Let furthermore ψ_m represent a wavelet basis function at iteration m and let ϕ_m be the corresponding scaling function, the framework of the wavelet transform (WT) allows us to construct an initial continuous approximation of our samples using scaling functions at iteration $m = m_0$:

$$\alpha^3(x,y,z) \quad = \quad \sum_{p,q,r\in Z} c^{\alpha}_{m_0pqr}\phi^3_{m_0pqr}(x,y,z) \tag{8}$$

$$q^3_{RGB}(x,y,z) \;=\; \sum_{p,q,r\in Z} c^{q_{RGB}}_{m_0pqr}\phi^3_{m_0pqr}(x,y,z) \tag{9}$$

where

$$\phi^3_{mpqr}(x,y,z)=\phi_{mp}(x)\cdot\phi_{mq}(y)\cdot\phi_{mr}(z) \tag{10}$$

and

$$\phi_{mn}(x)=2^{\frac{-m}{2}}\phi(2^{-m}x-n) \tag{11}$$

If, for instance, the cardinal B-spline wavelets of [2] and [17] are employed for further processing, the upper equation exactly corresponds to a B-spline approximation of the volume data.

Let‘s now preassume a discrete 3D tensor product wavelet transform, such as in [6] or [15], which decomposes separately each RGB and α component according to fig. 1.

$$\begin{tikzcd}
c_{m_0pqr} \arrow[r] \arrow[dr] \arrow[dddr] & c_{m_0+1,\,pqr} \arrow[r] \arrow[dr] \arrow[dddr] & \cdots \arrow[r] \arrow[dr] \arrow[dddr] & c_{Mpqr} \\
 & d^1_{m_0+1,\,pqr} & & d^1_{Mpqr} \\
 & \vdots \quad \vdots & \vdots & \vdots \quad \vdots \\
 & d^7_{m_0+1pqr} & & d^7_{Mpqr}
\end{tikzcd}$$

Fig. 1. Initial 3D wavelet transform to decompose both volume color and opacity samples.

The resulting coefficients represent the underlying coordinates of the decomposed functions in the functional space of wavelets and scaling functions. As a consequence we end up with a hierarchical approximation of the initial samples. A multiresolution expansion of the color and opacity functions can be provided as follows:

$$\alpha^3(x,y,z) = \sum_{m=m_0+1}^{M}\sum_{type=1}^{7}\sum_{p,q,r\in Z} d^{\alpha,\,type}_{mpqr}\psi^{3,\,type}_{mpqr} + \sum_{p,q,r\in Z} c^{\alpha}_{Mpqr}\phi^3_{Mpqr} \tag{12}$$

$$q^3_{RGB}(x,y,z) \;=\; \sum_{m=m_0+1}^{M}\sum_{type=1}^{7}\sum_{p,q,r\in Z} d^{q_{RGB},\,type}_{mpqr}\psi^{3,\,type}_{mpqr} + \sum_{p,q,r\in Z} c^{q_{RGB}}_{Mpqr}\phi^3_{Mpqr} \tag{13}$$

Based on the upper expansion, it is straightforward to compute the opacity gradient $\nabla\alpha$, which is fundamental to any illumination and shading algorithm. The normal $\boldsymbol{n}(x(t), y(t), z(t))$ defined for any spatial position along a viewing ray is given by

$$\boldsymbol{n}(t) \;=\; \frac{\nabla\alpha(t)}{|\nabla\alpha(t)|} \tag{14}$$

Since, according to eq. (12) α is defined by a linear combination of wavelets and scaling functions, the partial derivatives are computed by

$$\partial\alpha^3(x(t),y(t),z(t))/\partial x \;=\; \partial \sum_{m=m_0+1}^{M}\sum_{type=1}^{7}\sum_{p,q,r\in Z} d^{\alpha,\,type}_{mpqr}\psi^{3,\,type}_{mpqr} \tag{15}$$

$$/\partial x \;+\partial \sum_{p,q,r\in Z} c^{\alpha}_{Mpqr}\phi^3_{Mpqr}$$

$$/\partial x$$

Similar computations are necessary for $\partial\alpha^3/\partial y$ and $\partial\alpha^3/\partial z$. Because of the tensorproduct nature of the basis functions ϕ^3_{Mpqr} and $\psi^{3,\,type}_{mpqr}$, the upper expression can be computed analytically. The scaling function satisfies:

$$\partial\phi^3_{mpqr}/\partial x \;=\; (d\phi_{mp}(x)/dx)\cdot\phi_{mq}(y)\cdot\phi_{mr}(z) \tag{16}$$

For the wavelets we recieve similar expressions.

If the bases are given in closed form, as for instance in the case of B-splines, the normal can be computed analytically.

Note, that the upper expansions enable one to harvest all of the advanced features of the wavelet transform, such as

- progressive refinement
- hierarchical approximations

- local support
- linear time complexity
- smooth approximations
- error bounds

Recall, furthermore, that both global and local thresholding of single coefficients finally govern the compression ratio and overall level of detail of the approximation [5].

3 Ray-Tracing in Wavelet Space

This section introduces a ray-tracer in wavelet domain, which essentially computes an approximate solution of eq. (5). There have been various attempts, such as in [6], [15], [20] which however feature high computational costs. This motivated us to come up with alternative solution. When designing this algorithm, particular emphasis was given on efficient data structures and fast algorithms for the local recovery of the color and opacity function.

3.1 Numeric Approximation

Following eq. (5) the color value $I_{RGB}(t_0, t_L)$ is computed for each pixel in the image plane by line integration through the illuminated volume, weighted exponentionally with the accumulated opacity function along the ray $\boldsymbol{t}$. The outer integral typically runs from an entry point t_0 up to a leaving point t_L. For computational reasons we rewrite eq. (5) into a recurrence relation as:

$$I_{RGB}(t_0,t_i) = I_{RGB}(t_0,t_{i-1}) + e^{-A(t_0,\,t_{i-1})}\int_{t_{i-1}}^{t_i} q_{RGB}(t)e^{-A(t_{i-1},\,t)}dt \tag{17}$$

with

$$A(t_k, t_l) := \int_{t_k}^{t_l}\alpha(s)ds \tag{18}$$

This relation implies a discretization of the ray parameter t. Based on the approximations of α and q from eqns. (12) and (13), we have to find a numeric quadrature method, because eq. (17) cannot be solved analytically. Figure 2 illustrates the process of direct volume rendering in wavelet space. After an inital WT of color and opacity, a filtering step enables to control the precision of the approximation. The local reconstruction step composes the attenuation and source term, where in the rightmost part of the pipeline $A(t_0, t)$ is computed. Obviously, it is not necessary to perform a global inverse wavelet transform. Moreover, due to the local support of the bases, it is sufficient to collect a limited amount of wavelets to evaluate the required functions α and q at any spatial position. The major problem is to find a computationally efficient method for the extraction of the relevant basis functions.

Fig. 2. Pipeline for wavelet domain volume ray-tracing

The numeric techniques proposed for solving the volume rendering integral are based on the Newton-Cotes quadrature formulae. It enables us to describe the error-bounds analytically. For different segments of the volume data different step sizes along the ray ($\delta t_i = t_i - t_{i-1}$) are applied to ensure that the approximation error due to the integration scheme is less then the upper error bound, defined by the user. For the trapezoid rule the approximation error is given by:

$$Error_{I_{trapezoid}}(t_0, t_{i-1}, t_i) = (\Delta t)^2\frac{t_i - t_{i-1}}{12}e^{-A(t_0,\,t_{i-1})}(q(\xi)\alpha(\xi)e^{-A(t_{i-1},\,\xi)})^{(2)} \qquad \xi \in (t_{i-1}, t_i) \tag{19}$$

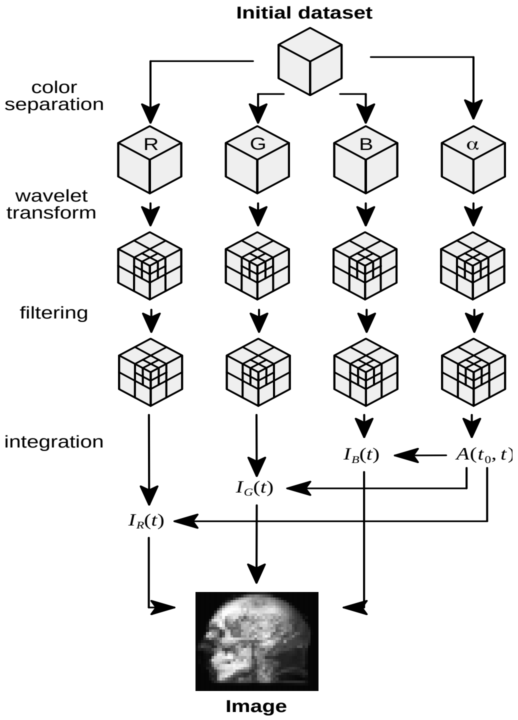
where t_{i-1} and t_i describes the intersection parameter of the viewing ray and the support region of the wavelet, Δt denotes the stepsize of the N+1 evenly spaced parameters between t_{i-1} and t_i and (n) denotes the n th. derivative. The error bound that is obtained by Simpson’s rule can be expressed as:

$$Error_{I_{Simpson}}(t_0, t_{i-1}, t_i) = (\Delta t)^4\frac{t_i - t_{i-1}}{180}e^{-A(t_0,\,t_{i-1})}(q(\xi)\alpha(\xi)e^{-A(t_{i-1},\,\xi)})^{(4)} \qquad \xi \in (t_{i-1}, t_i) \tag{20}$$

Details can be found in [12]. Note, that within a multiresolution setup the support regions of different wavelets and scaling functions can overlap significantly.

3.2 Computation of Relevant Basis Functions

As stated earier, the major procedure, which finally determines the computational efficiency is the local reconstruction of the initial signal from the wavelet transform pyramid. Therefore, it is necessary to recover all basis functions at a given spatial position ($x(t)$, y(t), $z(t)$) on the viewing ray. Due to the local support of the bases, *relevance* can be defined as the set of all functions, that do not vanish, e.g.



$$rel(f, t) = \{(m, p, q, r) \in \mathbf{Z}^4 | ((x(t), y(t), z(t)) \in supp(f^3_{mpqr}))\} \tag{21}$$

A fast and accurate detection of these parameters is essential to restrict the summations in eqns. (12) and (13). In 1D we define the support of a wavelet ψ and of a scaling function ϕ as:

$$\begin{aligned} [l_\phi, r_\phi] &:= supp(\phi) \\ [l_\psi, r_\psi] &:= supp(\psi) \end{aligned} \tag{22}$$

This expression generalizes to an arbitrary dilation and translation parameter *m* and *p* into the following form

$$\begin{aligned} supp(f_{mp}) &= [(l_f + p)2^m, (r_f + p)2^m] \\ & f \in \{\phi, \psi\}, m, n \in \mathbf{Z} \end{aligned} \tag{23}$$

Hence, for each resolution parameter *m* and for a given position $(x, y, z)^T \in \mathbf{R}^3$ the corresponding *p*,*q* and *r* have to satisfy:

$$\begin{aligned} \lfloor x/2^m - r_f \rfloor \le p \le \lceil x/2^m - l_f \rceil \\ \lfloor y/2^m - r_f \rfloor \le q \le \lceil y/2^m - l_f \rceil \\ \lfloor z/2^m - r_f \rfloor \le r \le \lceil z/2^m - l_f \rceil \end{aligned} \tag{24}$$

3.3 Data Structures and Implementation

Due to the multiple overlap of the support regions of wavelets and scaling functions, sophisticated data handling and list management is essential to obtain a sufficient outperformance. In our implementation, we generate a two-level list data structure for each ray, which keeps all basis functions contributing to the approximation of the actual spatial location. It is based on separate chaining. The first level of the list is organized with respect to the different levels of decomposition and connected via pointers. In the second level of the list, we keep all nonvanishing coefficients in RGBα, whose individual leaving parameter t_A is greater than the current ray parameter *t*. They are arranged with respect to t_A. Figure 3 illustrates the principal data organization we propose.

Once the current integration parameter *t* exceeds the leaving parameter t_A of a single list element (wavelet coefficient, index, t_A) is removed from the list. Conversely, any new element is added if *t* enters the wavelet‘s region of support.

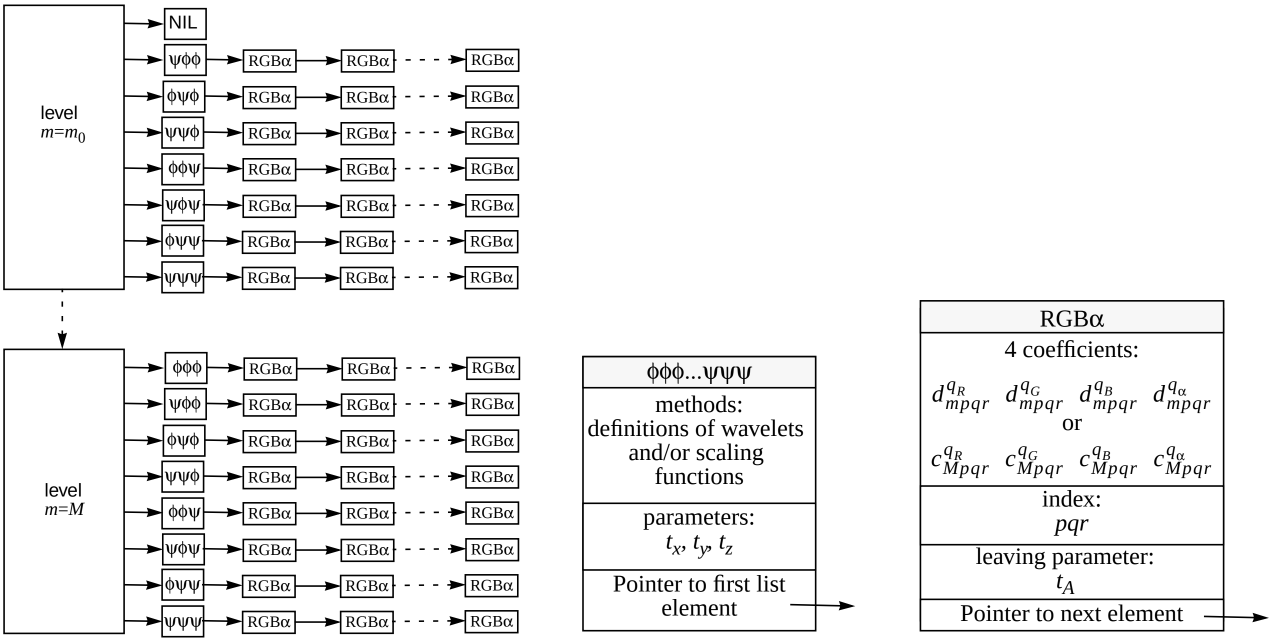

level
$m=m_0$
NIL
ψφφ
φψφ
ψψφ
φφψ
ψφψ
φψψ
ψψψ
RGBα
level
$m=M$
φφφ
φφφ...ψψψ
methods: definitions of wavelets and/or scaling functions
parameters: t_x, t_y, t_z
Pointer to first list element
RGBα
4 coefficients:
$d^{q_R}_{mpqr}$ $d^{q_G}_{mpqr}$ $d^{q_B}_{mpqr}$ $d^{q_\alpha}_{mpqr}$
or
$c^{q_R}_{Mpqr}$ $c^{q_G}_{Mpqr}$ $c^{q_B}_{Mpqr}$ $c^{q_\alpha}_{Mpqr}$
index: pqr
leaving parameter: t_A
Pointer to next element

Fig. 3. Data organization for wavelet based volume ray tracing

In order to compute the leaving parameter t_A efficiently, we harvest the dilation and translation properties of the wavelet bases. For a specific wavelet type and iteration step m the support regions do not differ in size and are aligned to the principal axes of the coordinate system. That is, all intersection parameters can be derived from an initial computation using three edges of the prototype function f_{m000}. Let a ray be defined by its origin ($\boldsymbol{ray}_{initial}$) and by a direction ($\boldsymbol{ray}_{direction}$). The intersection parameters of the ray with respect to the three „*leaving edges*“ are obtained by parametric clipping:

$$t_{leaving,i} = \begin{cases} \dfrac{b_i - ray_{initial,i}}{ray_{direction,i}} & \text{if } ray_{direction,i} > 0 \\ \infty & \text{if } ray_{direction,i} = 0 \\ \dfrac{a_i - ray_{initial,i}}{ray_{direction,i}} & \text{if } ray_{direction,i} < 0 \end{cases} \tag{25}$$

$$dt_i = \begin{cases} \dfrac{b_i - a_i}{ray_{direction,i}} & \text{if } ray_{direction,i} \neq 0 \\ 0 & \text{if } ray_{direction,i} = 0 \end{cases}$$

where $\boldsymbol{a}$ and $\boldsymbol{b}$ specify the bounding box of the basis and $i \in \{x, y, z\}$.

Each basis, translated from the prototype by p, q, r yields the following intersection parameters.

$$\begin{aligned} t_x &= t_{leaving,x} + p \cdot dt_x \\ t_y &= t_{leaving,y} + q \cdot dt_y \\ t_z &= t_{leaving,z} + r \cdot dt_z \end{aligned} \tag{26}$$

The required t_A is selected as the minimum of:

$$t_A = min(t_x, t_y, t_z) \tag{27}$$

This scheme is further explained by fig. 4, in which a 2D setup is presented for illustration.

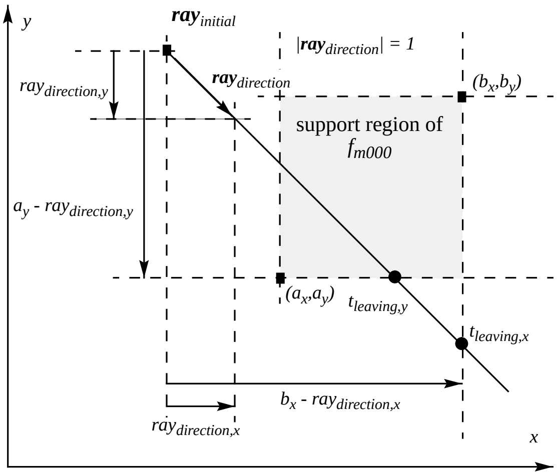


Fig. 4. Computation of the parameter t_A for a single basis function.

From the various examples presented in section 5 it can be fixed easily, that the method presented here enables one to carry out highly realistic images. The approximation order is governed by the type of wavelet underlying the transform. We recommend cardinal B-spline wavelets, because they yield a natural multiresolution extension to the standard B-spline approximation scheme [2].

4 Wavelet Based Volume Splatting

The method introduced above can be considered as a reference, since it renders high quality images including opacity, shading and exponential transfer. However, the algorithms to accomplish this, are still computationally expensive. Obviously, in order to achieve real time and interactive performance we have to find a different approach to the problem. Moreover, a trade-off has to be found between computational costs and image quality. Therefore, we developed an object oriented wavelet splatting method, which was initially scetched in [11] and is extended subsequently.

4.1 Wavelet Splats

Designing this method, we were inspired by both Fourier projection slicing volume rendering, such as in [13] or [19] and by splatting methods [9]. It is founded on the observation, that for many applications, as in medical imaging, X-ray like images are even superior for interactive data exploration and analysis. Recalling eq. (7) these pictures are computed by a line integral through the volume data. Consequently, the color $I_{\mathrm{RGB}}(\mu,\nu)$ at a particular spatial position (μ,ν) of the image plane is defined as an integral over the parameter domain t of the ray:

$$I_{RGB}(\mu, \nu) = \int_{t_0}^{t_L} q(x(t, \mu, \nu), y(t, \mu, \nu), z(t, \mu, \nu))dt \tag{28}$$

Inserting a wavelet approximation of q results in

$$\begin{aligned} I_{RGB}(\mu, \nu) = & \sum_{p,q,r} c_{M,pqr}^{q_{RGB}} \int_{-\infty}^{\infty} \phi_{M,pqr}^{3}(x(t, \mu, \nu), y(t, \mu, \nu), z(t, \mu, \nu))dt \\ & + \sum_{m = m_0 + 1}^{M} \sum_{type = 1}^{7} \sum_{p,q,r} d_{m,pqr}^{q_{RGB}, type} \cdot \int_{-\infty}^{\infty} \psi_{m,pqr}^{3,type}(x(t, \mu, \nu), y(t, \mu, \nu), z(t, \mu, \nu))dt \end{aligned} \tag{29}$$

Equation (29) can be interpreted as a weighted accumulation of integrals of the type $\int \phi_{M,pqr}^{3} dt$ representing the scaling functions and of integrals like $\int \psi_{m,pqr}^{3,type} dt$ for the various wavelets. Due to the local support of our bases, the lower and upper bounds of the integration can be set to $\pm\infty$ even for finite volume data sets. More precisely, these integrals finally turn out to be footprints of splats, in terms of RGBα textures weighted by the associated coefficients. The self-similarity of the wavelets enables us to compute the integrals once for each of the 8 prototype functions $\phi^3_{M,\,000}$ and $\psi^{3,\,type}_{M,\,000}$. All other textures are derived from it by scaling and translating them in the image plane. Note, that R, G and B components of the resulting textures have the same structure and are splatted by weighting with the coefficients of the transform. In short, our method takes the volume rendering as an accumulation of scaled and translated versions of weighted RGB textures, as shown in fig. 5.

Note furthermore, that wavelet based splatting comes along with all advantagous properties, listed previously. Our problem can be reduced to finding a method to compute efficiently the required line integrals (footprint textures) for the prototype functions.

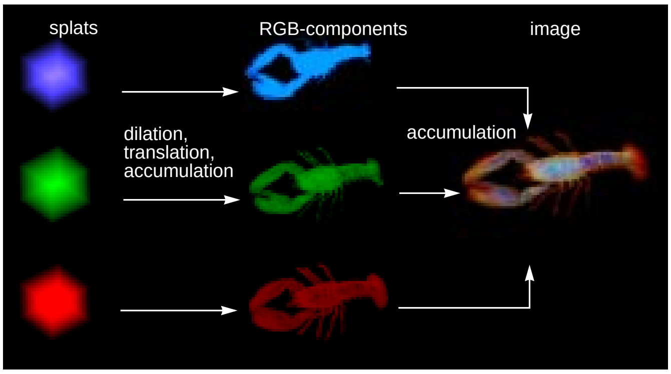


Fig. 5. Volume rendering as a splatting process of wavelet textures.

4.2 Splat Computation via Fourier Projection Slicing

The computation of the line integrals can be accomplished by Fourier projection slicing. As a fundamental theorem of multidimensional Fourier transforms [4], it essentially states, that the Fourier transform (FT) of a bunch of line integrals of a function *f*(*x*,*y*,*z*) is given by slicing its Fourier transform $F(\omega_1, \omega_2, \omega_3)$ with a plane perpendicular to those lines and intersecting the origin. Widely used in imaging, this method allows to compute the line integrals of any multidimensional band-limited tensor product type function. Since many wavelet types, such as B-splines [2], come along with closed form representations in frequency domain, it is straightforeward to apply this theorem to get the required splats. Figure 6 depicts the setting, where an inverse FFT processes the slices to obtain the wavelet splat.

Formally the method can be described as follows:

Let $F(\omega_1, \omega_2, \omega_3)$ be the 3D FT of a volume function $f(x, y, z)$ as

$$F(\omega_1, \omega_2, \omega_3) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x, y, z) e^{-i(x\omega_1 + y\omega_2 + z\omega_3)} dxdydz \tag{30}$$

The intersection plane spanned by ***u***, ***v*** defines the 2D Fourier transform $I(u, v) = F(\omega_1(u, v), \omega_2(u, v), \omega_3(u, v))$ of our texture splats. The definition of the viewing parameters is figured out in spherical coordinates (α,β), where the following relations hold:

$$\boldsymbol{n} = \begin{bmatrix} \cos\alpha\cos\beta \\ \sin\alpha\cos\beta \\ \sin\beta \end{bmatrix} \quad \boldsymbol{u} = \begin{bmatrix} -\sin\alpha \\ \cos\alpha \\ 0 \end{bmatrix} \quad \boldsymbol{v} = \begin{bmatrix} -\cos\alpha\sin\beta \\ -\sin\alpha\sin\beta \\ \cos\beta \end{bmatrix} \tag{31}$$

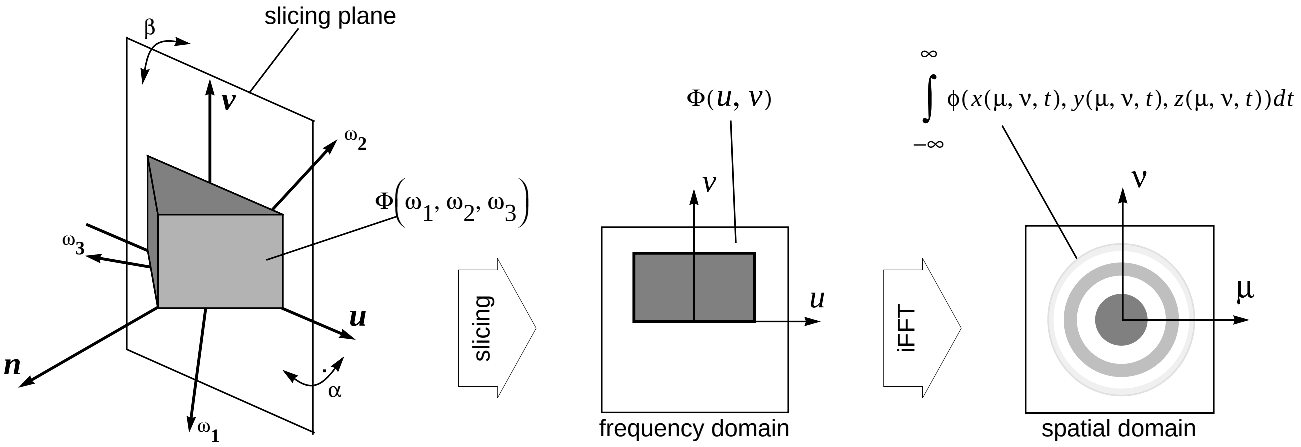

slicing plane
β
v
ω2
$\Phi(\omega_1, \omega_2, \omega_3)$
ω3
u
n
α
ω1
slicing
$\Phi(u, v)$
v
u
frequency domain
iFFT
$\int_{-\infty}^{\infty} \phi(x(\mu, \nu, t), y(\mu, \nu, t), z(\mu, \nu, t))dt$
ν
μ
spatial domain

Fig. 6. Illustration of the Fourier projection slicing theorem in 3D for an idealized Shannon wavelet

Note, that the slicing of the FT depends strongly on the type of wavelet and can be accomplished analytically. As opposed to Fourier domain rendering, our method does not require expensive interpolation filtering in these cases. The footprint of the splats are carried out using iFFT methods. A set of splats for cardinal B-spline scaling functions of different order are presented in fig. 7.

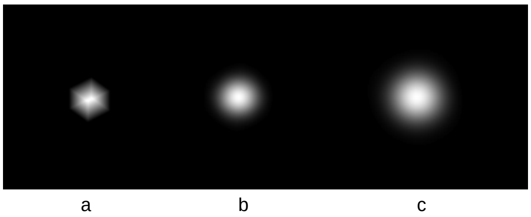


Fig. 7. Footprints of splats for cardinal B-spline functions of different order. a) order: 1, b) order = 2, c) order = 4

Note furthermore, that the footprints can both be positive and negative valued for the wavelets, respectively.

4.3 Determination of the Sampling Rate

Although the method generally does not require expensive filtering in frequency domain, yet we have to specify the correct sampling rate in order to discretize the FTs of our wavelets and scaling functions.

The sampling rate is finally derived by Shannons‘ sampling theorem [1]. The following considerations focus on B-spline wavelets, but are not restricted to them. Any other type of wavelet, whose support can be computed with the standard second order moment formulae [2] performs as well.

In order to derive the theoretically optimal sampling rate as a function of the scaling parameter *m*, we start from the support of the basis in spatial domain, which is given for cardinal B-Spline wavelets of order *j* on the real line with [0,2*j*-1]. Consequently, the optimal sampling rate $\Delta\omega_m$ in frequency domain is obtained by inverse application of Shannon‘s theorem, which states:

$$\Delta\omega_m < \frac{2\pi}{supp(f)} \tag{32}$$

In 3D however, the maximum length of the support is defined by the spatial diagonal giving rise to an additional factor of $\sqrt{3}$. We yield for the wavelet that correponds to a scaling function of order *j*:

$$\Delta\omega_m = \frac{2\pi}{\sqrt{3}2^m(2j-1)} \tag{33}$$

The relations between wavelets of different scaling parameters *m* in spatial and frequency domain are regulated by the Fourier scaling theorem:

$$FT(2^{-m/2}\psi(2^{-m}x)) = 2^{m/2}\Psi(2^m\omega) \tag{34}$$

Obviously, sampling the Fourier transform of a wavelet ψ at positions $i \cdot \Delta\omega_m$ is invariant to scaling except of a factor $2^{m/2}$.

Let ω_s bound the frequency interval, which computes as a function of *m* as $[-2^{-m}\omega_s, 2^{-m}\omega_s]$. That is the sampling rate increases according to eq. (33) but the interval drops appropriately. Hence, at a first glance the total number of required samples in frequency space would be constant. We have to note, however that the initial sampling rate $\Delta\omega_0$ repeats the continuous footprint in spatial domain. In order to compute the splat in spatial domain we have to sample them with the pixelrate of the framebuffer. This operation enforces conversely a periodic spread of the spectrum, which is essentially independent of the iteration *m*. Since the bandwidth drops but the support increases with *m*, the overall number of samples grows with *m*. The scaling properties of the WT allow one to interpret the bases of different scaling parameters *m* to be sampled at a constant rate $\Delta\omega_0$, while scaling the bounds of the sampling interval with

$$[-2^m\omega_s, 2^m\omega_s] \tag{35}$$

The final number of samples at a given iteration *m* required, according to Shannon, yields to:

$$N(m) = \left\lceil \frac{2^{m+1}\omega_s}{\Delta\omega_0} + 1 \right\rceil \tag{36}$$

Similar expressions can be derived for the 2D case, where we define a rectangular sampling region in frequency domain bounded by (*w*,*h*). This region scales according to

$$([-2^m w, 2^m w], [-2^m h, 2^m h]) \tag{37}$$

The number of required samples in (***u***,***v***) direction is obtained accordingly:

$$\begin{aligned} NW(m) &= \lceil 2^{m+1} w/\Delta\omega_0 + 1 \rceil \\ NH(m) &= \lceil 2^{m+1} h/\Delta\omega_0 + 1 \rceil \end{aligned} \tag{38}$$

This equation finally determines the spatial resolution of the footprint texture as a function of the scaling parameter *m*. Figure 8 illustrates the relationship for a Haar scaling function and presents the corresponding sampling grids. It is easy to see, that scaling the wavelet requires scaling the bounds of the sampling region appropriately while keeping the sampling rate according to the framebuffer.

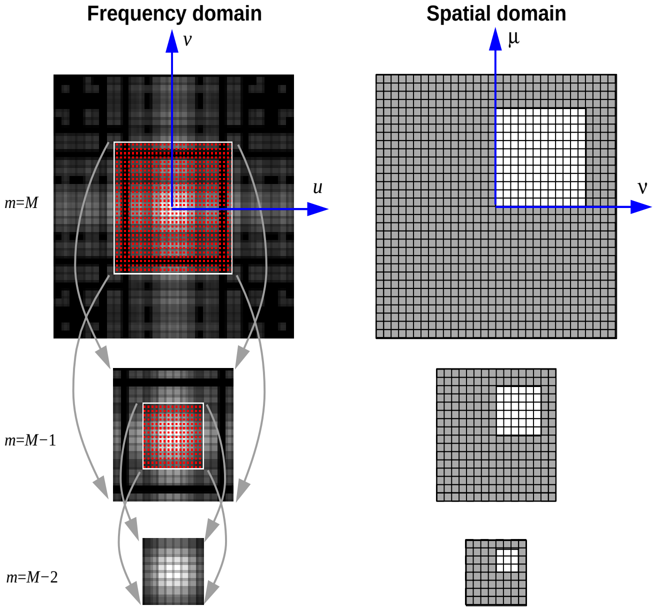


Fig. 8. Sampling of the slicing plane of 3D wavelets of different scale in frequency domain

A typical setting of the paramters is given for *w*=*h*=14 for B-spline wavelets of increasing order.

TABLE I: sampling rates for B-spline wavelets of increasing order

	j = 0 (Haar)	*j* = 2 (linear)	*j* = 4 (cubic)
m = 0	9	25	56
m = 1	17	48	110
m = 2	32	94	218
m = 3	63	187	434

Note, that the relations from above hold for optimal conditions since the implemented FFT restricts the number of samples to powers of two, we recommend a weak oversampling.

4.4 Some Remarks on Implementation

Although some advanced graphics workstations offer hard- and software support for fast texture accumulation, hardware independence encouraged us to implement the entire method as a software solution. More precisely, the textures are stored in terms of sparse matrix data structures, as illustrated in fig. 9.

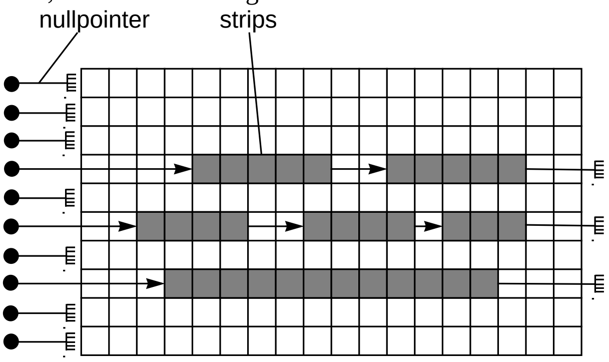


Fig. 9. Sparse matrix data structure to store the footprint textures

The accumulation is carried out efficiently by a software framebuffer. Since we restrict the summations to bounding boxes around the footprints, the accumulation is performed at nearly interactive rates.

Additional antialiasing operations are required to prevent artifacts from the resampling of the footprints in the framebuffer.

5 Examples

This section elucidates the benefits we obtain from the wavelet based volume rendering schemes explained above. Therefore, we present results computed on different volume data sets. In particular, the Visible Human Data Set was used to produce some of the images shown below. For all computations we employed cardinal B-spline wavelets of different order, since they offer versatile useful properties for rendering applications, such as:

- Closed form descriptions in spatial and frequency domain
- strict local support
- C^{n-2} continuity

In addition, we restrict the implementation to wavelets defined on the real line and renounced bounded intervals.

However, it should be stated, that both methods are not limited to a particular type of wavelet.

5.1 Wavelet Based Ray Tracing

Figure 10 shows images computed from the hydrogene volume data set, where linear scaling functions are contrasted against cubics for different levels of iteration. For illustration purposes, only the scaling functions are displayed. Obviously, the linear splines produce only C^0 continuous approximations, whereas cubics allow a smooth representation of the underlying volume data sets.

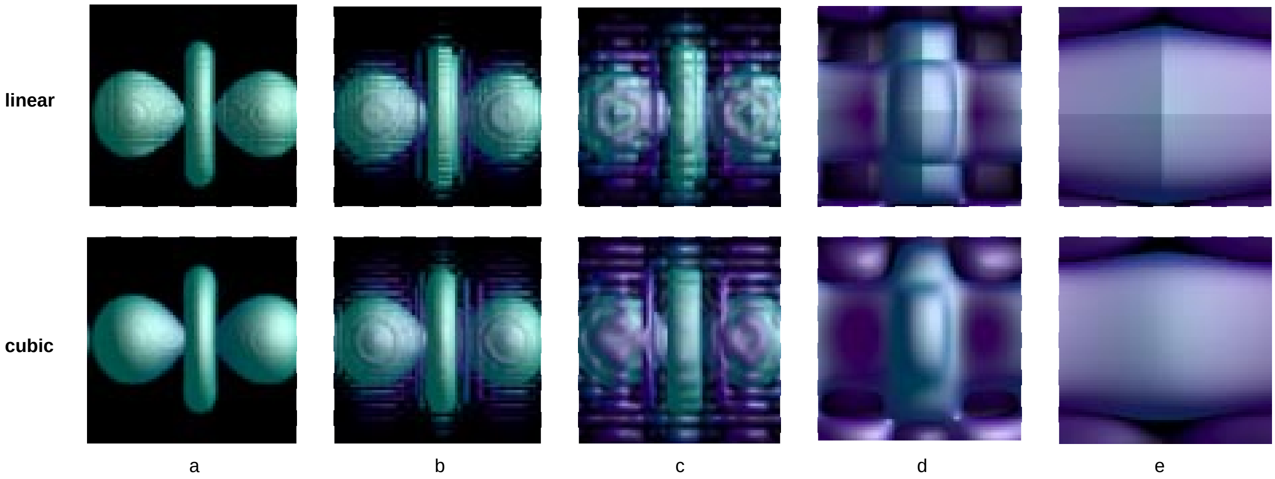


Fig. 10. Linear versus cubic B-spline wavlets for different levels of resolution. Only the scaling functions are employed for rendering. a) M = 0, compression rate: 0.0%, b) M = 0, compression rate: 88.0%, c) M = 2, compression rate: 99.5%, d) M = 4, compression rate: 99.97%, e) M = 5, compression rate: 99.997%

In order to depict the compression ratio to be achieved, fig. 11 represents approximations at different levels of detail. In this case, both wavelets and scaling functions contributed to the representation of the data and were thresholded appropriately to produce the results.

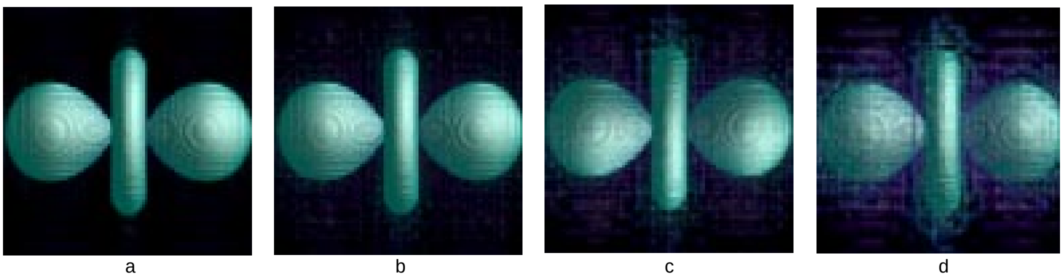


Fig. 11. Thresholding linear spline wavelet coefficients at M=2: compression rates: a) 63.3 %, b) 91.3 %, c) 94.8 %, d) 96.8 %

The presentations in fig. 12 reveal one of the most important properties of the wavelet transform: the localization. A local level of detail operation was carried out by thresholding the coefficients depending on the spatial position of the associated bases. For these purposes, a wavelet space filter was developed in [5].

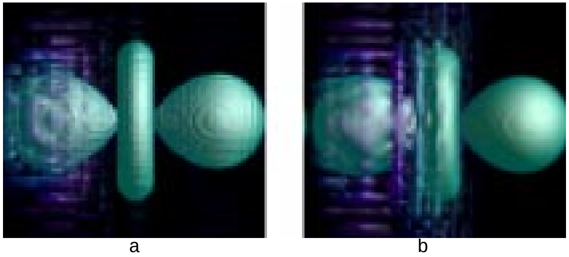


Fig. 12. Local level of detail operation by Gaussian filtering in wavelet space (M=2) a) linear, compression rate: 62.9 % , b) cubic, compression rate: 68.3 %

Similar results can be produced on the Visible Human Data Set. In fig. 13 the Gaussian filter was moved from back to front through the volume. As a result, different parts of the head are reconstructed at different levels of detail. Note, that Haar wavelets do not feature illumination.

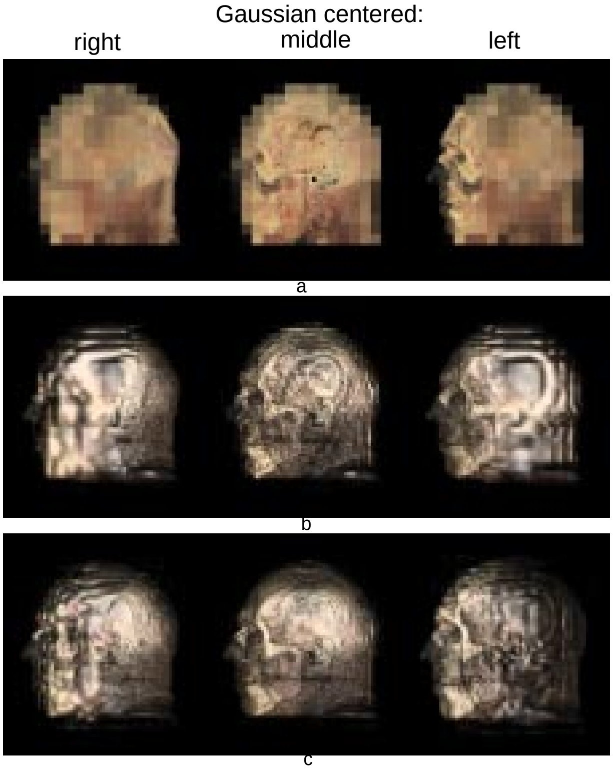


Fig. 13. Local level of detail operations on the Visible Human data set at M=3 a) Haar, b) linear B-Splines, c) cubic B-Splines (Data courtesy of the Visible Human Project, National Library of Medicine, copyright (C)1995)

Figure 14 shows a view into the visible human dataset. The data was taken from the RGB photo slices. The basis function for the RGB and α values were selected individually to display the influence of different B-spline functions. For the Haar function, no gradient can be calculated and the shading model can not be applied. We observe, that increasing the order of the basis functions results in a smoother appearance of the data.

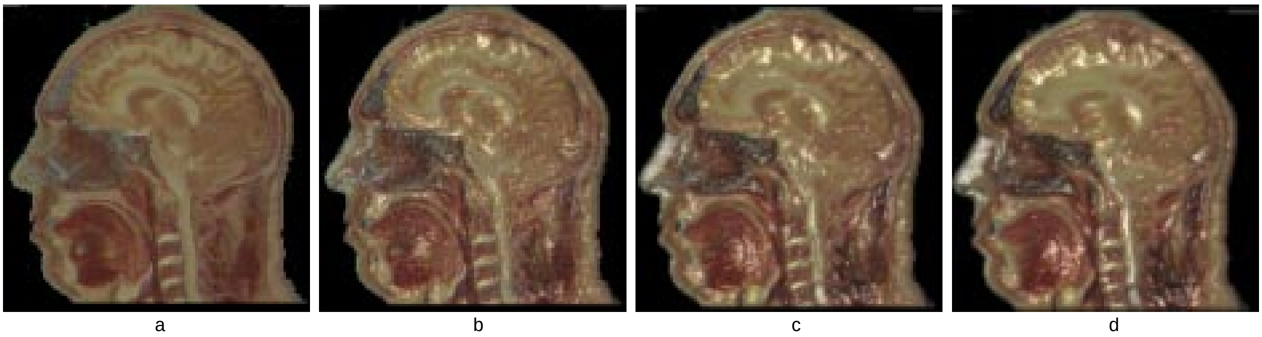


Fig. 14. Sagital view into the visible human dataset, rendered with different basis-functions: a) order 1(Haar), b) order 2, c) order 4, d) order 6

5.2 Wavelet-Based Splatting

As stated earlier, the splatting method progressively refines the image when more and more footprints are used to build up the representation. Consequently, it enables rendering schemes which allow interactive manipulation, such as translation, rotation or scaling. In order to get real time performance, only the most important splats are used for the reconstruction of the image. As soon as the image stands still, it is refined progressively. Therefore, the coefficients have to be sorted according to their magnitude. The resulting list is traversed during rendering and corresponding footprints are splatted into the framebuffer. Figure 15 shows a sequence of progressively refined images, computed on an SGI Indigo^2 / Mips R4400, 200MHz. After each step, the remaining error image was computed and is displayed along with rendering time and energy. Because of the high compression ratio achieved by the wavelet transform, the resulting images are visually equivalent after 4.1 s.

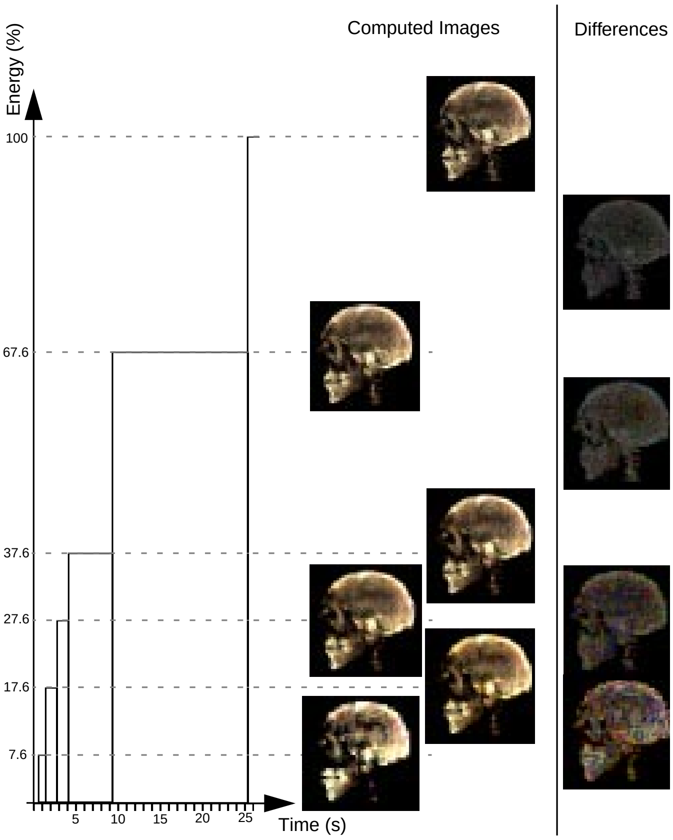


Fig. 15. Sequence of progressively refined images of the Visible Human Data Set for Haar wavelets. The error is shown after each refinement step.

In a similar way, local level of detail allows to focus the center of interest of the approximation.

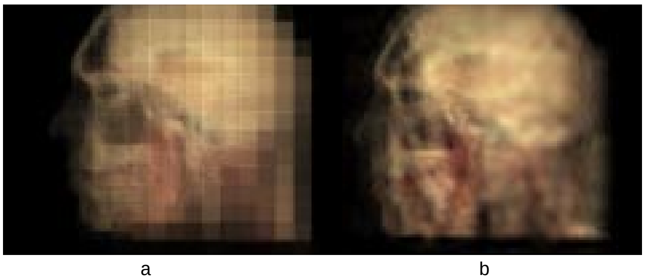


Fig. 16. Local LOD-control. a) Haar wavelet, compression rate: 65.5%, b) cubic B-Spline, compression rate: 46.2% (M=3)

This is depicted in fig. 16 for the same setup, as in fig. 13.

We also contrasted Haar wavelets against cubics to show up perfect reconstruction for different levels of approximation. Figure 17 gives an impression of the image quality to be achieved with our method.

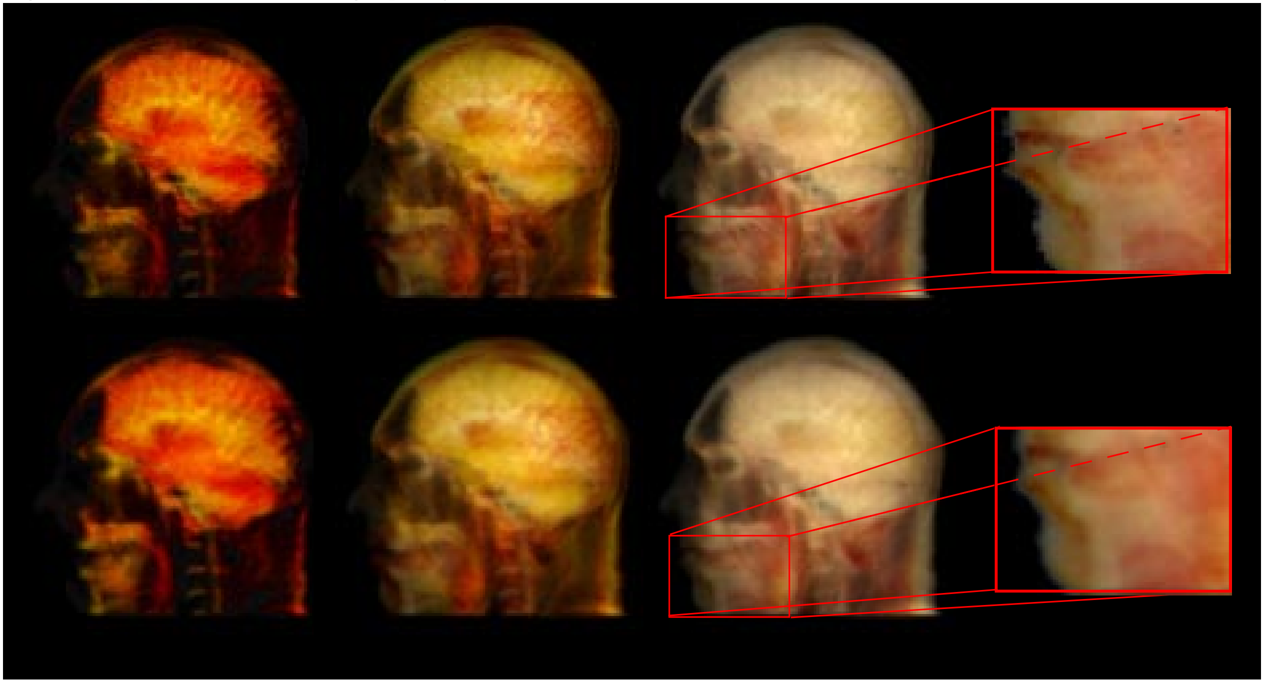


Fig. 17. Progressive reconstructions of the visible human dataset using 30, 60 and 90% of the M=0 scaling functions. Upper column: Haar wavelets, lower column cubic B-splines.

Note in particular the X-ray characteristics of the images. Although cubics are computational more expensive than Haar wavelets, the pictures prove that the image quality is similar.

6 Conclusions and Future Work

We have presented two fundamentally different approaches for wavelet based volume rendering. The first one, as a high level image order algorithm produces realistic images and is considered as a reference method. It encompasses all of the advanced features of the WT, but remains computationally expensive. Conversely, the second, object order splatting method was designed to achieve interactive rendering rates on low cost workstations and for network applications. The major shortcoming in image quality is exponential transfer and self-occlusion, which is in the focus of our future research. Future work also comprises shading algorithm for the splatting method. In summary, we believe that the results have illustrated the superiority of wavelet domain rendering methods.

7 Acknowledgement

This research was sponsored by the ETH research council under grant No. 41-2642.5.

References

[1] Brigham, E.O.: „The Fast Fourier Transform“. Prentice-Hall Inc., 1974

[2] Chui, C.K.: „An Introduction to Wavelets“. Academic Press, Boston, 1992

[3] Drebin, R.A.; Carpenter, L.; Hanrahan, P.: „Volume Rendering“. Computer Graphics, Vol. 22, No. 4, pp. 125-134, 1988

[4] Dudgeon, D.E.; Russel, M.M.: „Multidimensional Digital Signal Processing“. Prentice-Hall Inc, 1984

[5] Gross, M. H..; Staadt, O.; Gatti, R.: „Fast Multiresolution Surface Meshing“. Proceedings Visualization ‘95, IEEE Computer Society Press, pp. 135-142, 1995

[6] Gross, M. H.; Lippert, L.; Dreger, A.; Koch, R.: „A New Method to Approximate the Volume Rendering Equation Using Wavelets and Piecewise Polynomials“. Computers& Graphics, Vol. 19, No. 1, 1995

[7] Krüger, W.: „The Application of Transport Theory to Visualization of 3D Scalar Data Fields“. Computers in Physics, Jul/Aug , pp. 397-406, 1991

[8] Lacroute, P.; Levoy, M.: „ Fast Volume Rendering Using a Shear-Warp Factorisation of the Viewing Transform“. Computer Graphics , Proceedings of the SIGGRAPH ‘94, pp. 451-457, 1994

[9] Laur, D.; Hanrahan, P.: „Hierarchical Splatting: A Progressive Refinement Algorithm for Volume Rendering“. Computer Graphics and Applications, Volume 25, Number 4, pp. 285-288, 1991

[10] Levoy, M.: „Display of Surfaces from Volume Data“. IEEE Computer Graphics and Applications, Volume 8, Number 5, pp. 29-37, 1988

[11] Lippert, L; Gross, M. H.: „Fast Wavelet Based Volume Rendering by Accumulation of Transparent Texture Maps“. Computer Graphics Forum, Vol. 14, No. 3, pp. 431 - 443, 1995

[12] Lippert, L; Gross, M. H.: „Ray-tracing of Multiresolution B-Spline Volumes“. ETH Comp. Sc. Dept. internal report no. 239, 1995

[13] Malzbender, T.: „Fourier Volume Rendering“. ACM Transactions on Graphics, Vol. 12, No.3, pp. 233-250, July 1993

[14] Max, N.: „Optical Models for Volume Rendering“. in „Photorealistic rendering Techniques“ (G. Sarkas, P. Shirley, and S. Mueller, eds.), Springer Verlag, pp. 87-104, 1995

[15] Muraki, S.: „Multiscale 3D Edge Representation of Volume Data by a DOG Wavelet“. ACM Workshop on Volume Visualization Proceedings, pp. 35-42, 1994

[16] Papoulis, A.: „Signal Analysis“. McGraw-Hill Inc, 1977

[17] Quak, E.; Weyrich, N.: „Decomposition and Reconstruction Algorithms for Spline Wavelets on a Bounded Interval“. Appl. Comput. Harmon. Anal., Vol. 1, No.3, pp. 217-231, 1994.

[18] Sobierajski, L., Kaufman, A.: „Volumetric Ray Tracing“. ACM Workshop on Volume Visualization Proceedings, pp. 11-18, 1994

[19] Totsuka, T., Levoy, M.: „Frequency Domain Volume Rendering“. Computer Graphics Proceedings, Annual Conference Series, pp. 271-278, 1993

[20] Westermann, R.: „A Multiresolution Framework for Volume Rendering“. ACM Workshop on Volume Visualization Proceedings, pp. 51-57, 1994