Compression Domain Volume Rendering for Distributed Environments

L. Lippert, M. H. Gross, C. Kurmann

Department of Computer Science, ETH Zürich

Email: {lippert, grossm, kurmann}@inf.ethz.ch
WWW: http://www.inf.ethz.ch/department/WR/cg

Abstract

This paper describes a method for volume data compression and rendering which bases on wavelet splats. The underlying concept is especially designed for distributed and networked applications, where we assume a remote server to maintain large scale volume data sets, being inspected, browsed through and rendered interactively by a local client. Therefore, we encode the server‘s volume data using a newly designed wavelet based volume compression method. A local client can render the volumes immediately from the compression domain by using wavelet footprints, a method proposed earlier. In addition, our setup features full progression, where the rendered image is refined progressively as data comes in. Furthermore, framerate constraints are considered by controlling the quality of the image both locally and globally depending on the current network bandwidth or computational capabilities of the client. As a very important aspect of our setup, the client does not need to provide storage for the volume data and can be implemented in terms of a network application. The underlying framework enables to exploit all advantageous properties of the wavelet transform and forms a basis for both sophisticated lossy compression and rendering. Although coming along with simple illumination and constant exponential decay, the rendering method is especially suited for fast interactive inspection of large data sets and can be supported easily by graphics hardware.

Keywords: volume rendering, multiresolution, progressive compression, splatting, wavelets, networks, distributed applications

1. Introduction

Volume rendering, in general, has been a very important subfield of research in computer graphics. Since its invention [12], [16], countless algorithms have been proposed, [13], [14], most of which have been designed for providing high quality images at low computational efforts. In this context, recent advancements in graphics hardware help to exploit individual capabilities of graphics workstations [2]. If real time performance constrains the method and image quality can be relaxed, so-called splatting methods [15], [4] have proved to provide good results. Here, footprints of a volume primitive are computed in terms of a small pixmap texture and are superimposed in the framebuffer to carry out the final image. Introducing hierarchical basis functions, such as wavelets [17], rendering is carried out progressively by accumulating scaled and translated versions of self-similar textures.

Unfortunately, most rendering methods introduced so far relate to local computation environments. The problems arising with distributed environments and associated compression domain rendering has hardly been addressed in the literature [24], [10]. However, in times of distributed, world wide information systems and exponentially increasing volume data sets, requirements for rendering such data have changed. In many application scenarios, such as in medical imaging and information systems, situations arise, such as depicted in figure 1.

Often, volume data sets are stored and maintained by a data base server, which can be accessed by one or more local clients via a network. One of the fundamental tasks of volume rendering is to inspect interactively and to browse through the volume data base, where rendering quality can be relaxed for the sake of real time performance. Moreover, being a low end workstation, the computational power and storage capacity of the client are limited and the network‘s

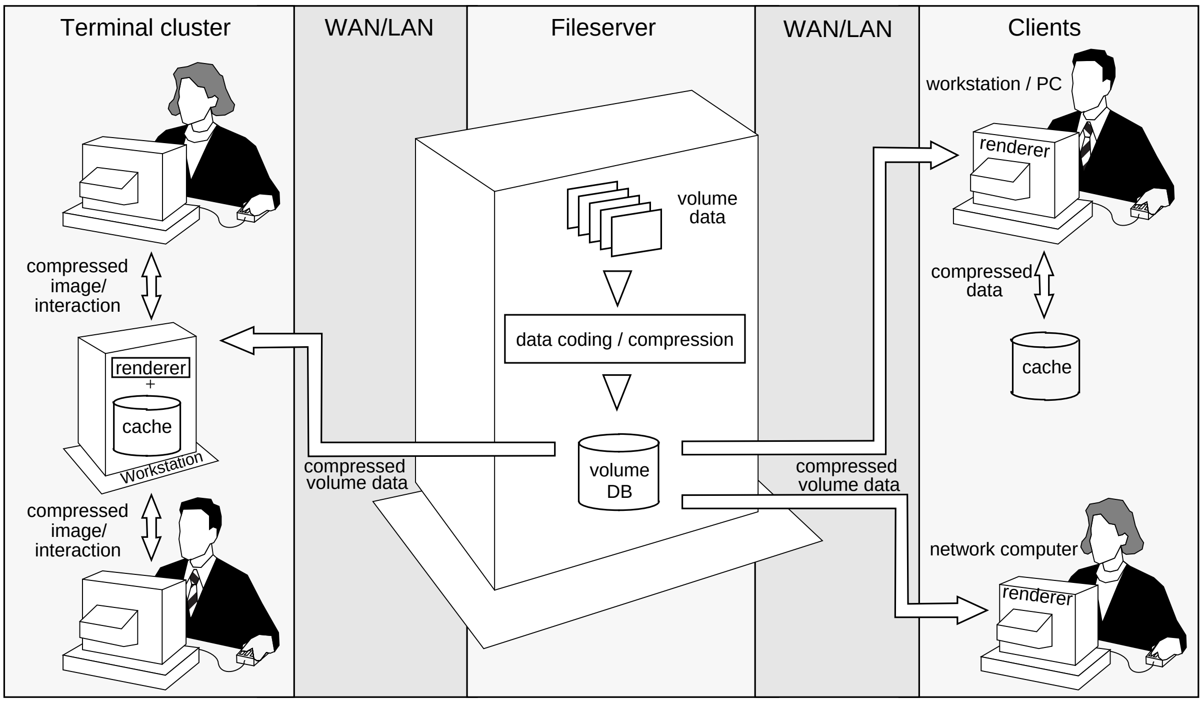


Figure 1: *Volume-rendering in a distributed client/server environment.*

bandwidth and latency might vary with its traffic load. Obviously, when designing a rendering method for these purposes, we have to consider the following issues:

- *Progressivity:* For enhanced interactivity, the image should be refined progressively as data comes in from the remote server. Image quality must be controlled as a function of the network's load and client's hardware setup.
- *Compression domains*: In order to store and transmit large scale volume data sets, compression schemes have to be employed. In order to avoid full decompression, a sophisticated rendering method should carry out computations in the compression domain at the client side.
- *Memory constraints:* In particular with upcoming network computers the capabilities of a local client might be reduced significantly. Hence, a data representation and rendering method is required, which avoids full expansion of the volume in the clients memory.

Along these lines, the research presented here aims at providing a unified framework for data compression and rendering in a distributed environment. Therefore, we first propose a progressive volume data compression scheme. It enables to store both RGB and intensity volume data efficiently on the server in terms of a bitstream and allows fast transmission over the network. To this end, we first decorrelate the RGB volume data and transform the computed intensities in a preprocess by using B-spline wavelets. The coefficients and positions are arranged with decreasing importance and proceed through various steps, such as quantization, encoding and bit allocation. The compression pipeline introduced essentially allows a progressive transmission, where the information lost can be controlled locally and globally. Second, the client carries out all rendering immediately in the compression domain. That is we do not need to perform full decompression of the data while using wavelet splats [17], [9]. To increase performance and visual quality of the images, we propose some significant enhancements of the method. Especially, multiview images can be carried out at low additional costs and constant exponential transfer is incorporated by phase shifts in Fourier domain.

Our paper is organized as follows: For reasons of readability, we briefly review some fundamentals of wavelet based splatting in section 2. In section 3 we describe some significant extensions of the rendering method to gain visual quality, such as multiviews, depth cueing and simple shading. The proposed compression pipeline is described in full detail in section 4. Results obtained with our setup are illustrated in section 5.

2. Fundamentals of Wavelet Splats

In this section we describe how wavelet splats can be defined and used for general volume rendering. Therefore, we briefly summarize the mathematical fundamentals of splat computation following the approach presented in [9]. In addition, a short description of low albedo rendering and wavelets are given. We assume, however, the reader is familiar with the mathematics of wavelets.

2.1. Low Albedo Volume Rendering

Volume rendering can be regarded as a composition of particles that, in addition to their own emission, receive light from surrounding lightsources and reflect these intensities towards the observer.

In classic volume rendering, the amount of light $I(t_L, \boldsymbol{x}, \boldsymbol{s})$ received at point $\boldsymbol{x}$ from direction $\boldsymbol{s}$ up to a ray length t_L is computed as:

$$I(t_L, \boldsymbol{x}, \boldsymbol{s}) = \int_0^{t_L} q(\boldsymbol{x} + t \cdot \boldsymbol{s}) e^{-\int_0^t \alpha(\boldsymbol{x} + s u) du} dt \tag{1}$$

where q denotes the volume source term and α the opacity function. For an isotropic medium with constant opacity α equation (1) reduces to

$$I(t_L, \boldsymbol{x}, \boldsymbol{s}) = \int_0^{t_L} q(\boldsymbol{x} + t \cdot \boldsymbol{s}) e^{-\alpha t} dt \tag{2}$$

In particular note that for $\alpha \equiv 0$ we end up in a X-ray-like image.

Especially in those cases, splatting has proved its usability for fast volume rendering [15]. In contrast to ray-casting, it allows to reduce the computational complexity for interpolation and integration to a minimum, since the preprojected footprints of high-order interpolation functions can be stored as lookup tables. The projections themselves can be computed with an accurate quadrature technique. Besides hierarchical splats [15], wavelet splatting [17] is a sophisticated extension.

2.2. B-spline Wavelets

Since we aim at a unified data representation for both volume data compression and rendering, we use Chui and Wang [3] B-spline wavelet bases of order j. They have many usefull properties, such as smoothness and vanishing moments. In addition, analytic expressions for scaling- and wavelet functions and their duals in the frequency- and spatial-domain are given. The mother-scaling function of order 1 is defined as:

$$\phi^{[1]}(x) = \begin{cases} 1 & 0 \le x < 1 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Compactly supported scaling functions of order j can be generated recursively in terms of cardinal B-splines:

$$\phi^{[j]}(x) = \frac{x}{j-1}\phi^{[j-1]}(x) + \frac{j-x}{j-1}\phi^{[j-1]}(x-1) \tag{4}$$

In the frequency-domain the corresponding functions collapse into *sinc*-polynomials of type:

$$\Phi^{[j]}(f) = \left(\frac{1-e^{-i2\pi f}}{i2\pi f}\right)^j \tag{5}$$

In the upper term f denotes the frequency and i the complex operator. Note, that the upper equation is fundamental for splat computation in Fourier domain. The semiorthogonality of the function system for $j > 1$ requires duals. For the case of B-spline wavelets all primary and dual functions have linear phase. Furthermore, only rational coefficients have to be used for the fast wavelet transform.

To fully characterize the wavelet transform, we need to determine the wavelet functions. For the corresponding wavelets we obtain:

$$\begin{aligned} \psi^{[j]}(x) &= \sum_{k=0}^{3j-2} q_{j,k} \phi^{[j]}(2x-k) \\ \text{with } q_{j,k} &= \frac{(-1)^k}{2^{j-1}} \sum_{l=0}^{j} \binom{j}{l} \phi^{[2j]}(k+1-l) \end{aligned} \tag{6}$$

Note, that their support can be computed straightforwardly as [0,2j-1]. Wavelets in three dimensions are obtained easily by non-standard tensor-product extensions [5].

Here volume decomposition is carried out with the duals, whereas reconstruction is performed using the primary functions (4), (6). Note furthermore, that implementations of linear time algorithms [20] can be more challenging, since additional basis transforms might be required.

2.3. Construction of Wavelet Splats

In wavelet splatting, the renderer computes the projection such as defined in (2). Taking into account the wavelet decomposition level m up to M and moving the summation outside the integral, the formulation collapses to:

$$\begin{aligned} I(\infty, \boldsymbol{X}, \boldsymbol{S}) = & \sum_{\substack{m=1 \\ p,q,r \in Z}}^{M} \sum_{type=1}^{7} d_{mpqr}^{type} \int_{-\infty}^{\infty} \psi_{mpqr}^{3,type}(\boldsymbol{x} + \boldsymbol{s}t) dt \\ & + \sum_{p,q,r \in Z} c_{Mpqr} \int_{-\infty}^{\infty} \phi_{Mpqr}^{3}(\boldsymbol{x} + \boldsymbol{s}t) dt \end{aligned} \tag{7}$$

where d_{mpqr}^{type} denote the wavelet coefficients of decomposition level m at the spatial position p, q, r and wavelet-type *type* and c_{Mpqr} the coefficient of the scaling function of level M. The computation of the line integrals for a particular view can be accomplished by Fourier projection slicing (FPS). It allows to compute accurate projections of any basis function. This theorem states that the 2D Fourier transform of a projection of a function $f(x,y,z)$ onto a given plane $\boldsymbol{P}$ equals a plane that slices the Fourier transform $F(\omega_1, \omega_2, \omega_3)$ parallel to $\boldsymbol{P}$ and intersects the origin.

Since many wavelet types such as B-splines come along with closed form representations in the frequency domain, it is straightforward to apply this theorem to get the required splats. Figure 2 depicts the setting, where an inverse FFT processes the slices to obtain the wavelet splat.

The intersection plane spanned by ***u***, ***v*** defines the 2D Fourier transform of the texture splats $I(u, v) = F(\omega_1(u, v), \omega_2(u, v), \omega_3(u, v))$, whereas the normal vector ***n*** of the plane equals the direction of the projection. The definition of the viewing parameters is figured out in spherical coordinates (α,β).

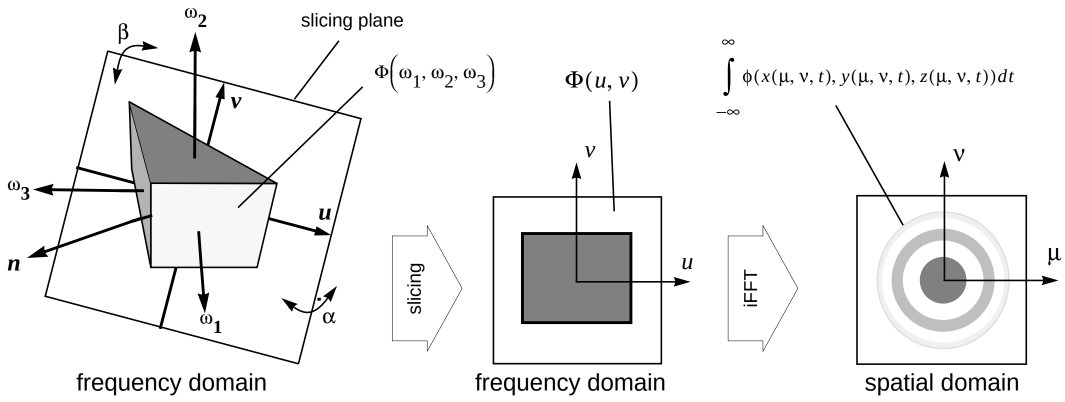


Figure 2: *Illustration of the Fourier projection slicing theorem in 3D for an idealized Shannon wavelet.*

Once the renderer builds the viewpoint dependent integral tables, the screen position of the table is calculated, mapped, weighted by the wavelet coefficient and accumulated into the framebuffer. Since the basis functions of different iteration levels *m* differ by dilation, only eight different splats of depth *M* have to be calculated. All other footprints are derived by subsampling in the spirit of a mipmap. Correct sampling and optimized data-structures of the calculated splats are discussed in [9].

3. Visual Enhancements

We are now in a position to discuss two significant rendering features which help to improve the visual quality of the generated images. In particular, we introduce a method for the computation of multiviews, being important in many applications, such as medical imaging. Exploiting the symmetry of 3D tensor-product wavelets allows the additional costs to be kept sufficiently low. Furthermore, we address the problem of exponential transfer functions in Fourier space. Phase shifts of the wavelet's FT enable exponential transfer to be incorporated and can be used to simulate shading operations on the fly.

3.1. Multiviews

The rendered images, obtained by our splatting approach, lack occlusion. Since this important visual cue is missing for the calculated X-ray images, depth information is not presented to the user. One way to overcome this drawback is the introduction of a multiview arrangement. Here, the volume is rendered simultaneously from different directions and presented to the user in a single window. Exploiting the coherence of different viewing angles given by the symmetry of tensor-product constructions, the basic single-view splatting approach can be extended to a multiview renderer without computational overhead for splat computation. We recall that the tensor-product functions are constructed from permutations of the 1D basis functions ϕ and ψ along the *x*, *y* and *z* directions.

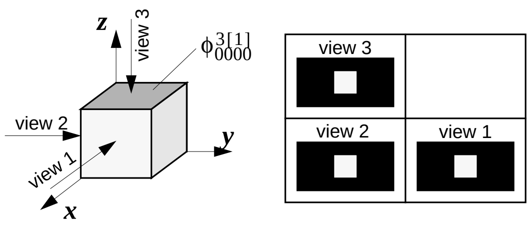


Figure 3: *Illustration of the multiview concept.*

Figure 3 exemplifies the idea. The footprint of the basis function $\phi_{0000}^{3[1]}$ (3D Haar scaling function of level 0, aligned to the origin) from the first viewing-direction equals the footprint from the second and third direction, e.g. the calculated footprints can be reused for rendering different views. Thus, we propose to generate a lookup table for the basis functions types 1 to 7 to ensure correct footprint calculation. A corresponding lookup table is given in Table 1 and illustrated in figure 4 for Haar functions respectively.

Table 1: Permutations and negations for multi view arrangements

	PERMUTATIONS	NEGATIONS
VIEW 1	[0 1 2 3 4 5 6 7]	[1 1 1 1 1 1 1 1]
VIEW 2	[0 1 4 5 2 3 6 7]	[1 1 -1 -1 1 1 -1 -1]
VIEW 3	[0 4 2 6 1 5 3 7]	[1 1 1 1 -1 -1 -1 -1]

In order to combine the three footprints for the eight different basis functions, eight textures are calculated for view 1 by the FPS-theorem. The footprint for the basis function $\phi\phi\psi$ is equal for view 1 and 2, whereas the splat of the view 3 is given by the footprint of the function $\psi\phi\phi$ (texture 4) of view 1. The corresponding permutations are given for all basis functions and for three views in Table 1. In some cases it is also necessary to multiply the intensity with -1. The same permutation tables can also be used for higher order wavelets.

Note that these permutations are equal for each viewing direction. Note furthermore that the angles between the individual viewing planes are not constant. Moreover, they depend on the initial selection of the camera parameters. A little algebra reveals that the three projection-planes P_i, spanned by the vectors $(\boldsymbol{u}_i,\boldsymbol{v}_i)$ are given by

View 1: $(\boldsymbol{u}_1, \boldsymbol{v}_1)$ with

$$\boldsymbol{u}_1 = \begin{bmatrix} -\sin\alpha \\ \cos\alpha \\ 0 \end{bmatrix} = \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix} \qquad \boldsymbol{v}_1 = \begin{bmatrix} -\cos\alpha\sin\beta \\ -\sin\alpha\sin\beta \\ \cos\beta \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}$$

View 2: $(\boldsymbol{u}_2, \boldsymbol{v}_2)$ with

$$\boldsymbol{u}_2 = \begin{bmatrix} -\cos\alpha \\ -\sin\alpha \\ 0 \end{bmatrix} = \begin{bmatrix} -u_y \\ u_x \\ u_z \end{bmatrix} \quad \boldsymbol{v}_2 = \begin{bmatrix} -\sin\alpha\sin\beta \\ \cos\alpha\sin\beta \\ \cos\beta \end{bmatrix} = \begin{bmatrix} -v_y \\ v_x \\ v_z \end{bmatrix}$$

View 3: $(\boldsymbol{u}_3, \boldsymbol{v}_3)$ with

$$\boldsymbol{u}_3 = \begin{bmatrix} 0 \\ \cos\alpha \\ \sin\alpha \end{bmatrix} = \begin{bmatrix} u_z \\ u_y \\ -u_x \end{bmatrix} \quad \boldsymbol{v}_3 = \begin{bmatrix} -\cos\beta \\ -\sin\alpha\sin\beta \\ \cos\alpha\sin\beta \end{bmatrix} = \begin{bmatrix} -v_z \\ v_y \\ -v_x \end{bmatrix}$$

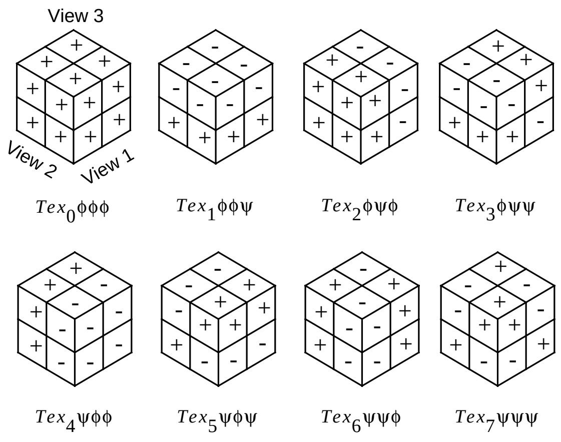


Figure 4: *Different wavelet types figured out by non-standard tensor-product extensions for the Haar case.*

The triple view rendering is illustrated in figure 5, where a classified data set is displayed from three directions. The image was grabbed directly from the screen as it is displayed to the user. Skin is colored white, brain tissue red and a tumor is colored in blue.

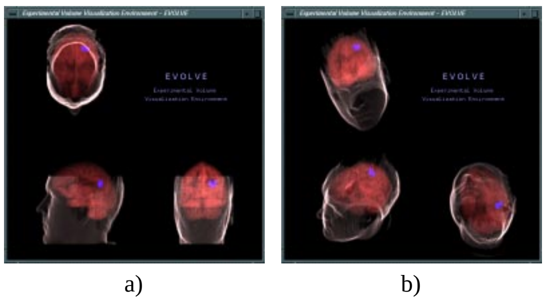


Figure 5: *Triple views seen from different viewing angles as applied to classified MR-data set (volume size: 256 x 128 x 256, max. decomposition level M = 2) a)* $\alpha = \beta = 0.0$ *b)* $\alpha = 0.66$, $\beta = 0.91$.

3.2. Depth Cueing

Up to now, only linear depth cueing has been considered in the frequency domain [22]. In order to compute constant exponential decay as in (2) we propose a depth cueing approach implemented as a two step solution. In the first step the cued footprints have to be calculated by the FPS-theorem and an additional phase-shift. Second, the textures have to be weighted according to the distance between the projection plane and the basis functions.

Footprint Computation In order to set up the Fourier relations for exponentially weighted functions, let's first consider the relations in the spatial domain. Let $h(t)$ denote the function that is depth cued along t and let k be the constant exponential extinction coefficient. We obtain the new function $\tilde{h}(t)$ as:

$$\begin{aligned} \tilde{h}(t) &= h(t)e^{kt} \\ &= h(t)e^{-i2\pi f_0 t} \quad \text{where} \quad f_0 = \frac{k}{-2\pi i} = \frac{ki}{2\pi} \end{aligned} \tag{8}$$

In the frequency domain the Fourier transform of $\tilde{h}(t)$ can be written as:

$$\begin{aligned} \tilde{H}(f) &= \int_{-\infty}^{\infty} \tilde{h}(t)e^{i2\pi ft}dt \\ &= \int_{-\infty}^{\infty} h(t)e^{i2\pi(f-f_0)t}dt \\ &= H(f-f_0) \end{aligned} \tag{9}$$

Obviously, the Fourier transform of the exponentially depth-cued function $\tilde{h}(t)$ is computed by shifting $H(f)$ with the imaginary frequency f_0. This can be interpreted as a "*frequency phase shift*". For the B-spline scaling functions and wavelets of order j in the frequency domain we have

$$\tilde{\Phi}^{[j]}(f) = \left(\frac{1-e^{-i2\pi(f-f_0)}}{i2\pi(f-f_0)}\right)^j$$
$$\tilde{\Psi}^{[j]}(f) = R^{[j]}\left(e^{-i\pi(f-f_0)}\right)\tilde{\Phi}^{[j]}\left(\frac{f}{2}\right) \tag{10}$$

where

$$R^{[j]}\left(e^{-i\pi(f-f_0)}\right) = \frac{1}{2}\sum_{n=-\infty}^{\infty} q_{j,n}\cdot\left(e^{-i\pi(f-f_0)}\right)^n \tag{11}$$

In 3D the direction of the shift operation in the spectrum can be chosen without any restriction and independently from the viewing direction. It corresponds to the depth-cueing direction in the spatial domain. We define the cueing vector $\boldsymbol{L}$ for any pair of viewing angles (α_l, β_l) as

$$\boldsymbol{L}(\alpha_l, \beta_l) = |\boldsymbol{k}|\begin{bmatrix}-\cos\alpha_l\cos\beta_l\\ \sin\alpha_l\cos\beta_l\\ \sin\beta_l\end{bmatrix} = \begin{bmatrix}k_x\\ k_y\\ k_z\end{bmatrix} \tag{12}$$

to determine the cueing direction and extinction coefficient. Figure 6 shows the calculated splats resulting from this method.

Distance Weighting For correct depth cueing of the volume, the computed footprints have to be weighted according to their distance from a reference plane which can be regarded as a planar light source, perpendicular to the depth-cueing vector $\boldsymbol{L}$ that points towards the volume midpoint $\boldsymbol{V}_M$. Let $\boldsymbol{L}_{init}$ represent the intersection point of the volume's bounding sphere and its tangent plane, the weighting factor ω_d for each basis function centered at $\boldsymbol{B}_M = \boldsymbol{P} + \boldsymbol{L}_{init}$ can be computed as:

$$\omega_d = e^{-dist} \quad \text{where} \quad dist = \frac{\boldsymbol{L}\cdot\boldsymbol{P}}{|\boldsymbol{L}|} \tag{13}$$

During the accumulation process this factor is multiplied with the wavelet coefficient and the accumulation step proceeds straightforwardly. The geometric relationships are depicted in figure 7.

As a result we come up with smooth and correctly depth-cued volumes, such as shown in figure 6.

To illustrate the performance of our method, we applied the approach to the visible human CT-data set [18]. The influence of different factors $|\boldsymbol{k}|$ and different cueing directions are displayed in figure 8 Note that exponential depth cueing increases the rendering performance, since less textures have to be accumulated. Moreover, interactive manipulation of the cueing direction allows us to simulate simple shading operations in the wavelet domain and enhances the visual quality significantly.

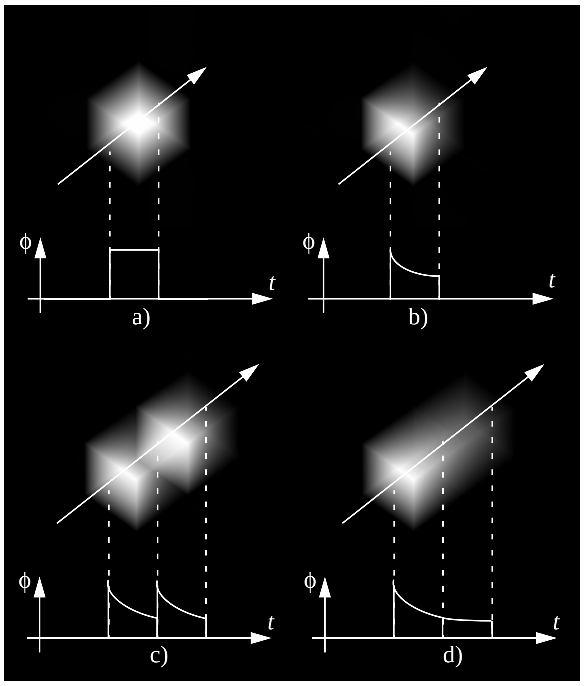
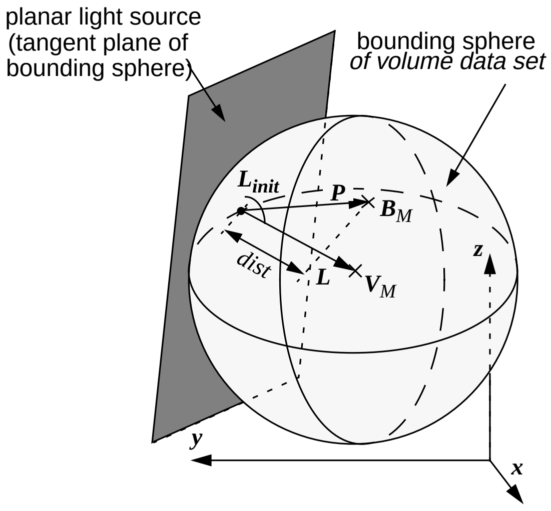


Figure 6: *Exponentially depth-cued scaling function (j=1) a) |**k**|=0.0. b) |**k**|=0.6. Distance weighted footprints: c) depth cued splats without distance weighting, d) corrected splats using distance weighting.*



planar light source (tangent plane of bounding sphere)
bounding sphere of volume data set
L_init
P
B_M
dist
L
V_M
z
y
x

Figure 7: *Geometric relationships for distance-weighting of basis functions.*

4. Progressive Compression and Transmission

As motivated earlier, our approach is targeted at networked applications where, for instance, a local client with low computational power browses through a remote database. Thus, in order to transmit our volume data efficiently we have to find appropriate compression strategies. It is clear that the underlying framework of the wavelet decomposi-

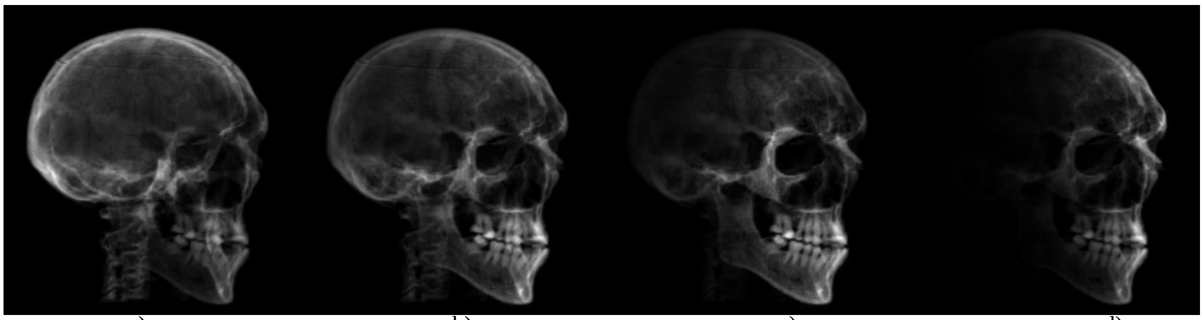


Figure 8: *Influence of different cueing parameters and directions.(data source: Visible Human Project, courtesy National Library of Medicine, copyright (C) 1995). Volume size: 256^3, max. decomposition level M=3. a-c):* $\alpha = \alpha_l = 2.5$, $\beta = \beta_l = 0.1$, *a)* $|\boldsymbol{k}| = 0$, *b)* $|\boldsymbol{k}| = 0.1$, *c)* $|\boldsymbol{k}| = 0.2$, *d)* $\alpha = 2.5$, $\alpha_l = \pi/4$, $\beta = 0.1$, $\beta_l = 0.3$, $|\boldsymbol{k}| = 0.2$.

tion proposes to develop an optimized compression technique that allows progressive transmission, one pass decompression and direct rendering at interactive frame rates. Moreover, the wavelet domain rendering avoids full decompression of the data prior to the splatting which itself as an image based method does not require to store the full volume data at the client's side. Although much research has been done on wavelet compression methods [21], [23] the specific needs of compression domain rendering encouraged us to develop a new compression pipeline which will be explained below. For the sake of brevity we restrict our description to the issues essential for our method and recommend further readings in the fundamentals of data compression [11].

4.1. Overview

Figure 9 illustrates the data flow in our compression and rendering setup. The data preprocessing comprises five stages. It enables both intensity and RGB volumes to be handled, which might be the result of an optional data classification step [1]. In case of RGB volumes the second step consists of a colorspace optimization which essentially decorrelates the data and allows color sensitive quantization. Next, a wavelet transform is performed independently on the three channels. Lossy compression is carried out by an oracle [8] which operates locally or globally in the wavelet domain. The final step includes a data compression and encoding scheme to achieve a binary output stream that can be stored locally or transmitted directly through a network. Note that forward compression does not have any real-time constraints as opposed to the decompression.

In our implementation the software decompressor works at a rate of ~30 k wavelet-coefficients/sec. on an Indy R4400 workstation and allows on-line decompression. The renderer splats the computed footprints weighted with the decoded wavelet-coefficients directly into a software or hardware accumulation buffer. In addition, our framework incorporates a local cache to store coefficients at the client's side. The required splats are computed locally. Since the wavelet coefficients are transmitted in significance order the rendering quality is fully controlled by a user-defined framerate, the client's hardware and, if no cache mechanism is enabled, also by the bandwidth of the network. Hence, this concept balances CPU, network and graphics performance and allows scalability. In a minimum configuration we have to provide client storage only for the eight mother-wavelet splats and three Huffman tables. Together they take less than 5KB of memory even for huge data sets. This even allows the scheme to run on some of the upcoming network computers.

4.2. Colorspace Transformations

If the initial volume is given in RGB, it is critical to transform the volume into an optimized colorspace prior to compression. Here, we assume the optimized space to be spanned by the three vectors $\boldsymbol{C}_1$, $\boldsymbol{C}_2$ and $\boldsymbol{C}_3$. This allows to assign an additional significance to each vector. As a result we get two independent significance weights per coefficient which affect encoding and quantization. The first weight is defined by the energy of the associated function [8]. The second one is a global significance determined by the colorspace-coordinates. For instance, a coefficient with the coordinates (1,0,0) is regarded as more relevant than a coefficient (0,1,0) if the vector $\boldsymbol{C}_1$ is considered to be more significant than $\boldsymbol{C}_2$.

In addition to RGB we employ two colorspaces: The first one is data-independent and equals the YIQ-colorspace obtained by a simple matrix transform [19]. The *Y* component encodes the luminance information, whereas the chromaticity is encoded in *I* and *Q*. We followed the NTSC bandwidth conventions and assigned a factor 4 to *Y*, 1.5 to *I* and 0.6 to *Q*. In practical use this colorspace allows to compute a black and white image (*Y*) as a rough sketch and to refine color progressively. Thus, progression is figured out both in the spatial *and* in color domain. Alternatively, our second colorspace is calculated by a statistically optimal principle component analysis (PCA) or Karhunen-Loève expansion [7] whose matrix has to be computed individually for each data set. Although the solutions of the eigenproblems are computationally more challenging on offline forward compression, yet they provide better results (see section 5). In this case the absolute values of the eigenvalues are taken to describe the significance of the corresponding eigenvector. Note that this step can be skipped for intensity volumes, such as raw CT or MRI data sets.

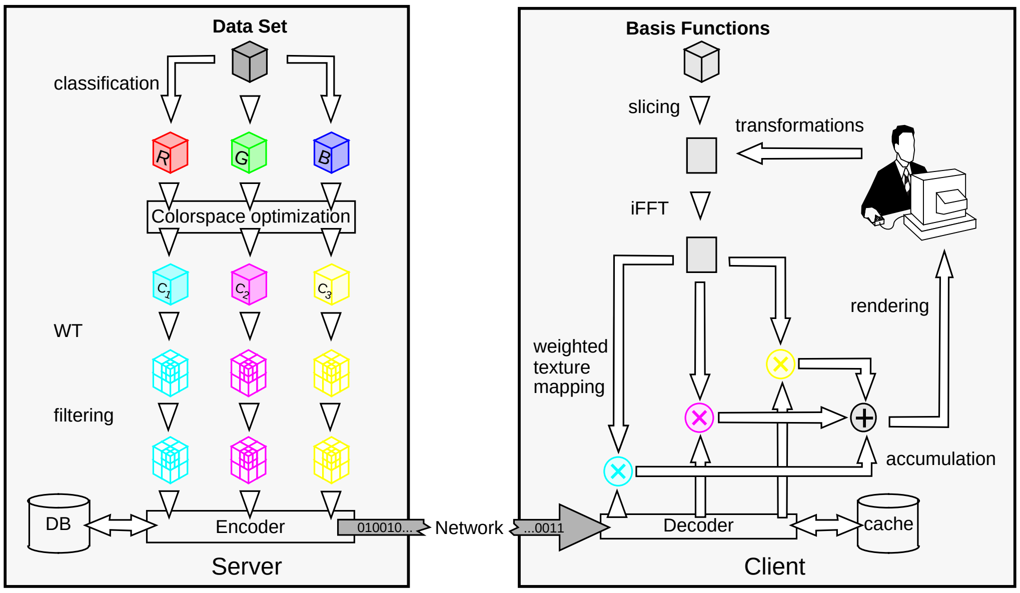


Figure 9: *Setup for distributed compression domain rendering.*

4.3. Data Compression Pipeline

The algorithmic steps for data compression are executed sequentially in the pipeline summarized in figure 10 and convert the wavelet transformed data sets into a sequential bitstream.

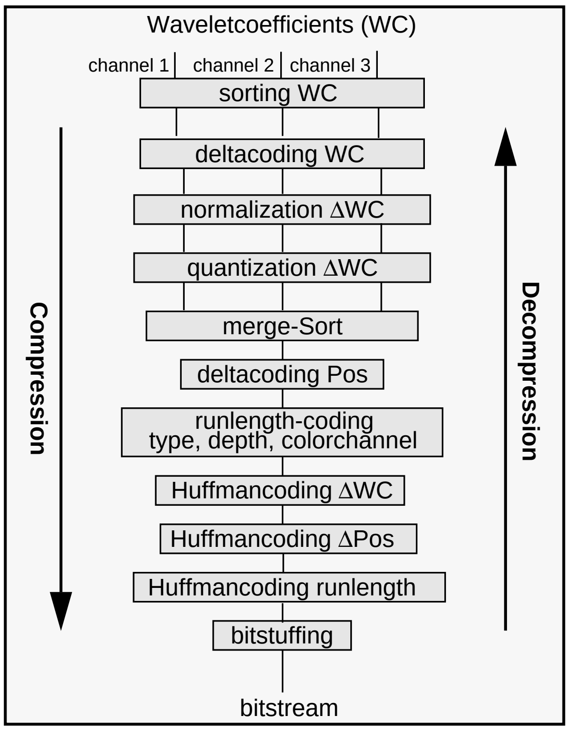


Figure 10: *Pipeline representing the individual steps of the compression scheme.*

Therefore, we start with a sorting operation that generates a sequence of coefficients and positional data in significance order. The significance score S is determined for each color channel $\mathbf{C} \in \{\mathbf{C}_1, \mathbf{C}_2, \mathbf{C}_3\}$ and coefficient $w(\mathbf{C}) \in \{d^{type}_{mpqr}(\mathbf{C}), c_{Mpqr}(\mathbf{C})\}$ individually according to its associated wavelet energy E and color channel importance I.

$$S(w, \mathbf{C}) = E(w(\mathbf{C})) \cdot I(\mathbf{C}) \qquad (14)$$

Table 2 summarizes the importance factors for the three different colorspaces.

Table 2: Colorspace significance assignment

COLORSPACE	IMPORTANCE I		
	$I(C_1)$	$I(C_2)$	$I(C_3)$
RGB	1	1	1
YIQ	4	1.5	0.6
EIGENVECTORS (PCA)	EIGEN-VALUE 1	EIGEN-VALUE 2	EIGEN-VALUE 3

For each color channel ($\mathbf{C}$), wavelet type (*type*) and decomposition level (m) a deltacoding, normalization and quantization operation is performed separately depending on the individual ranges of the coefficients w. More precisely, if quantization is restricted to P bits for a given sequence of N+1 significance sorted non-zero wavelet and scaling function coefficients $(w_n(\boldsymbol{C}, m, type))_{n = 0, \ldots, N}$, we compute the coefficient's delta factor $\Delta(\boldsymbol{C}, m, type)$ as:

$$\Delta(\boldsymbol{C}, m, type) = \frac{\max_{n = 0, \ldots, N-1}(|w_n(\boldsymbol{C}, m, type)| - |w_{n+1}(\boldsymbol{C}, m, type)|)}{2^P} \quad (15)$$

This factor is used to store the coefficient's range with respect to the assigned bytes. Thus, it has to be transmitted once for each wavelet type, decomposition level and color coordinate. In contrast, the normalized and quantized difference of two coefficients δ has to be transmitted for each coefficient individually. It is computed as:

$$\delta(\boldsymbol{C}, m, type, n) = round\left(\frac{(|w_n(\boldsymbol{C}, m, type)| - |w_{n+1}(\boldsymbol{C}, m, type)|)}{\Delta(\boldsymbol{C}, m, type)}\right) \quad (16)$$

Upon reconstruction we end up with approximated wavelet and scaling function coefficients $\tilde{w}_n(\boldsymbol{C}, m, type)$, $n = 1, \ldots, N$ which can be computed by:

$$\tilde{w}_{n+1}(\boldsymbol{C}, m, type) = sign(n+1)(|\tilde{w}_n(\boldsymbol{C}, m, type)| - \delta(\boldsymbol{C}, m, type, n) \cdot \Delta(\boldsymbol{C}, m, type)) \quad (17)$$

Since the coefficients are sorted according to their individual scores we optimize the residual approximation error as a function of the parameters introduced above. Note that the sign of each coefficient is encoded by an additional flag, refered as *sign*(*n*).

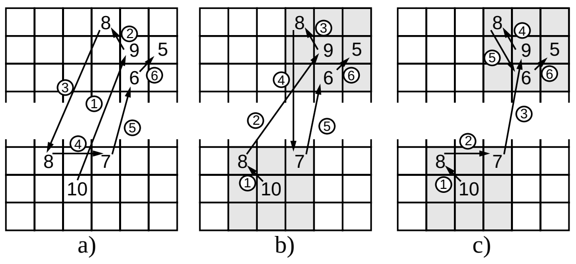


Figure 11: *Effect of spatial clustering:*
a) Without clustering
b) With clustering (position threshold: 3, coefficient threshold = 2)
c) With clustering (position threshold: 3, coefficient threshold = 4).

We choose to limit the quantization P to eight bits, since standard framebuffers use eight bits for RGBα each. However, observations in practice encourage us to reduce quantization to even three bits without significant lost of visual quality (see section 5). The three data sequences are merged and sorted according the scores of their wavelet coefficients. In particular, the sorted sequence requires for each coefficient to encode additionally its spatial position, wavelettype and color-channel in a lossless scheme. In order to overcome the drawbacks in compression performance arising from this requirement we introduce the following spatial clustering mechanism, which balances spatial and score constraints upon compression: Within a predefined radius around the coefficient with the highest score the algorithm selects a neighbor as a successor in the bitstream if its score exceeds a threshold. If not, the coefficient with the highest score is selected independently of its spatial position. The corresponding coefficients are labeled to avoid multiple storage. Figure 11 illustrates the ordering with and without clustering for the 2D case. The calculated coefficient weights are given in the corresponding box and the arrows denote the sorting order. Without clustering the arrow length and therefore the positional deltas are nearly equiprobable, whereas for clustered volumes mostly short arrows (small differences in position) appear. This reduces the entropy and brings up a better compression rate of the Huffman-coded positional deltas. That is we balance spatial coherence and the energy contribution sorting order of the coefficients. Note that clustering also speeds up the rendering process, since splats outside the field of view can be detected easily and skipped without further computation.

Additional runlength-codings of wavelettype, decomposition depth m and colorchannel are performed and transmitted as variable length Hufman tag codes. The third Huffman-table encodes the deltas δ of wavelet-coefficients. These three tables together take about 2kBytes and have to be transmitted separately prior to the data. In addition the transmitted meta-data includes information about the basis vectors of the colorspace, maximum depth of the wavelet transform (M), exact initial wavelet coefficients ($w_0(\boldsymbol{C}, m, type)$) and the coefficient delta factors ($\Delta(\boldsymbol{C}, m, type)$).

The reconstruction scheme has to decode all required information, such as the spatial position, wavelet type, depth m, colorchannel and the data value. According to the composition step, each tag is encoded as described in Table 3. For computational efficiency, we propose to precompute 10-bit Huffman look-up tables.

Table 3: Coding schemes:

TAG	CODING SCHEMES
SPATIAL POSITION	SPATIAL CLUSTERING, DELTA CODING, HUFFMAN-TABLE
TYPE	RUNLENGTH, HUFFMAN-TABLE
DEPTH	RUNLENGTH, HUFFMAN-TABLE
COLOR CHANNEL	RUNLENGTH, HUFFMAN-TABLE
SCALAR FACTOR (SIGN)	SINGLE BIT
SCALAR FACTOR (VALUE)	DELTA CODING, HUFFMAN-TABLE

Figure 12 illustrates a fraction of the bitstream as generated by our method. Note, that the number of bits varies as a function of the individual Huffman codes.

5. Results

To investigate the performance of the proposed method, we applied the approach to the RGB-Visible Human Dataset of size 128x128x128 voxels (3 x 8 bits/voxel). In addition to

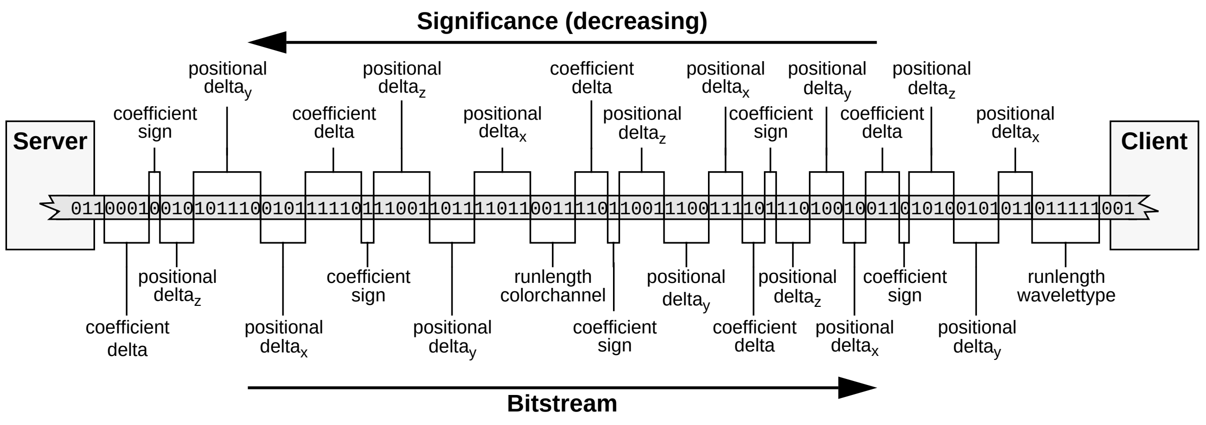


Figure 12: *Fraction of the bitstream generated by the compression scheme*

the RGB data set we generated a scalar representation of the volume by extracting the intensities Y from the RGB values. The wavelet decomposition was performed with Haar wavelets up to level M=3.The effects of lossy data compression of the Y-data set are illustrated in figure 13. In order to quantize image quality we define an L^2 image measure Q_I conforming to the signal-noise ratio (*SNR*) in [dB] well known from signal processing applications as:

$$Q_I = 20 \cdot \log_{10}\left(\frac{\sum_{pixel}\sum_{color}[i_{\mathrm{ref}}(pix, col)]^2}{\sum_{pixel}\sum_{color}[i_{\mathrm{im}}(pix, col) - i_{\mathrm{ref}}(pix, col)]^2}\right) \quad (18)$$

where, $i_{\mathrm{im}}(pix, col)$ denotes the intensity of a given *pix*el of the computed image for the *col*or-component in RGB-colorspace or the intensity Y for the Y-data set, e.g. $col \in \{R, G, B\}$ or $col \equiv Y$ respectively. Note specifically that in image compression ratios > 40 dB refer to reasonable visual qualities and at ratios >60 dB images are perceived as „noise-free“. The reference image was generated by the proposed splatting method for M=0 and 100% of the coefficients. We observe that ratios >60 dB are achieved at compression gains of almost 95 %.

The colorplates in figure 14 display the image quality achieved from the RGB data set for different colorspaces and compression rates with respect to the original data size of 6291456 bytes. The qualitative differences of the three colorspaces reveal mostly for small datasizes. Note that YIQ favourises the *Y*-component and renders greyscale images at high compression rates. This is contrasted by the RGB color space where the method reconstructs the volume both in the spatial and colorspace domain and ends up in a poorer image quality. Finally the best results are obtained by the PCA-based color representations. However, as progression proceeds the representations converge to each other. It is clear that the entropy of the color information is lower than in the Y channel. Therefore, we observe higher compression gains (SNRs) in figure 14 than in figure 13. Nevertheless even on intensity based compression the results are compelling and enable to use compressed versions of the volume as visual abstracts in large data bases.

We consider the volume’s L^2 approximation energy measure Q_V which is computed relatively to the uncompressed volume from the underlying framework of the WT as:

$$Q_V = \left(\frac{\sum_{n=0}^{N} E(w_n)}{\sum_{n=0}^{volume\ size-1} E(w_n)} \cdot 100\right) \quad [\%] \quad (19)$$

The PCA-based colorspace turns out to give the best L^2 approximations for a predefined compression rate as depicted in table 4.

This table also sums the performance and fidelity of our algorithm. Timings are given for a SGI-Indy workstation (MIPS R 4400/150 MHz) and a SGI Maximum Impact workstation (MIPS R10000/195 MHz). Both workstations use our software-accumulation scheme as introduced in [9]. The resolution of the rendered image was 160x180 pixels. For the delta-coding of the wavelet coefficients we assigned three bits. The timings reveal, that we still achieve interactive framerates for fast previewing. Note, in particular that competitive high quality renderers, such as shear warp factorization [14] are significantly slower at these data sizes and require careful setting of the transfer function for speed-up. Our proposed splatting technique is well-suited for hardware support [17]. The hardware assisted accumulation of the calculated splats is done within the accumulation buffer or uses alpha-blending operations, depending on the available hardware platform. Hardware support allows to further increase the rendering speed significantly, especially for the generation of high resolution images.

In order to investigate compression gain achieved by higher order spline wavelets we compared in Figure 15 the Haar transform to linear and cubic B-spline wavelets, where the MRI-volume data set of size 256x128x256 (8 bits/

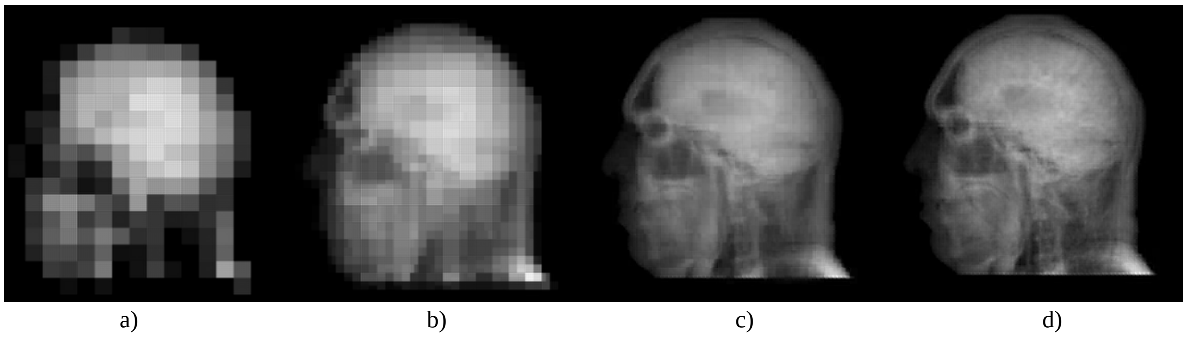


Figure 13: *Compression of intensity data set. (data source: Visible Human Project, RGB-data set converted to Y). a) data size: 1496 bytes (rel. data size: 0.07 %), Q_I = 19.19 dB. b) data size: 3164 bytes (rel. data size: 0.15 %), Q_I = 31.23 dB. c) data size: 26198 bytes (rel. data size: 1.25 %), Q_I = 48.71 dB. d) data size: 122680 bytes (rel. data size: 5.85 %), Q_I = 61.17 dB.*

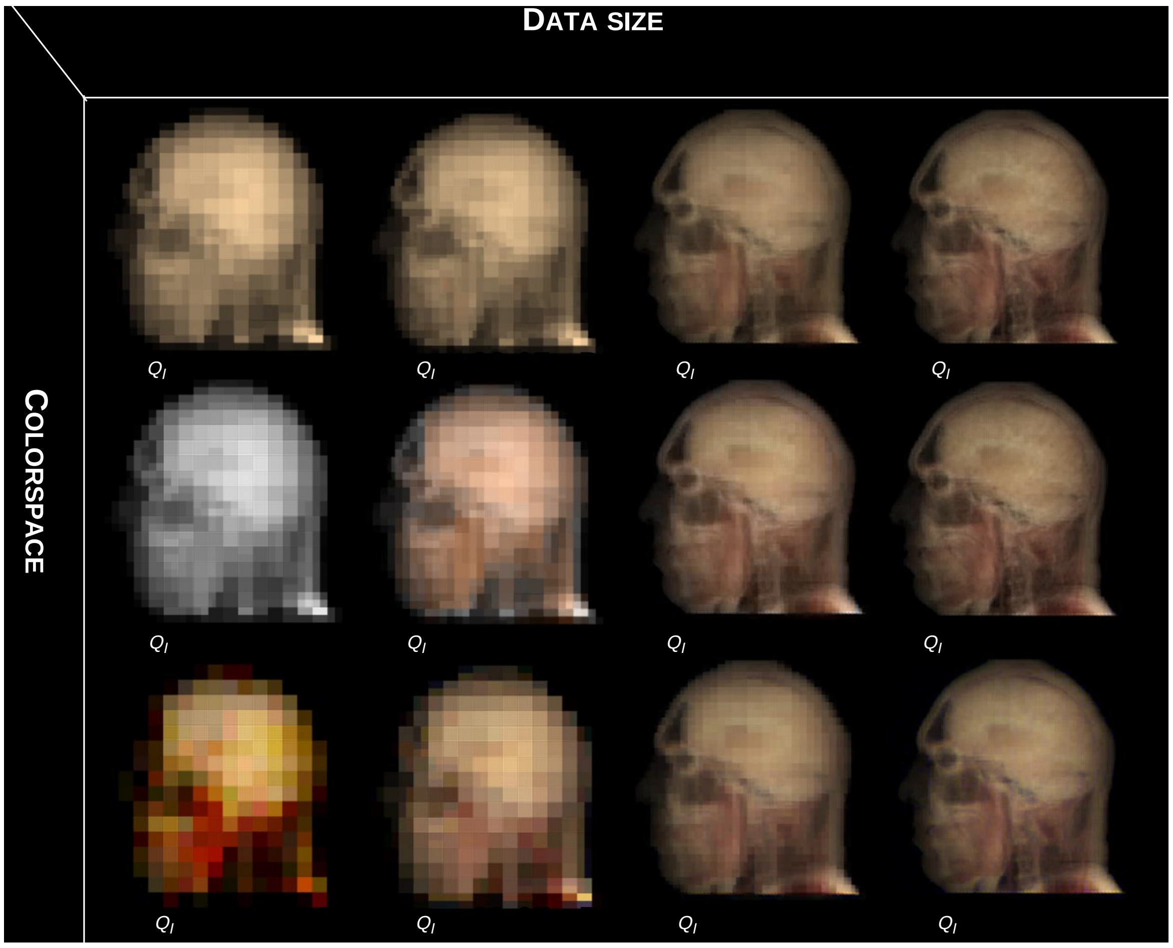


Figure 14: *Progressive compression in three different colorspaces. (data source: Visible Human Project, RGB-data set). Volume size: 128^3, max. decomposition level M=3.*

voxel) is displayed. The wavelet transform was performed up to *M*=2. The blocky structure of the Haar wavelet clearly leads to a blocky representation of the data set whereas the higher order wavelets come up with a smooth, somewhat blurry representation. Obviously, increasing numbers of vanishing moments lead to higher compression rates. A major drawback of higher order wavelets however is the increasing support which affects the splat size. As a consequence both FFT computation and the footprint accumulation are computationally more expensive. Furthermore the performance of spatial clustering drops in efficiency with increasing support.

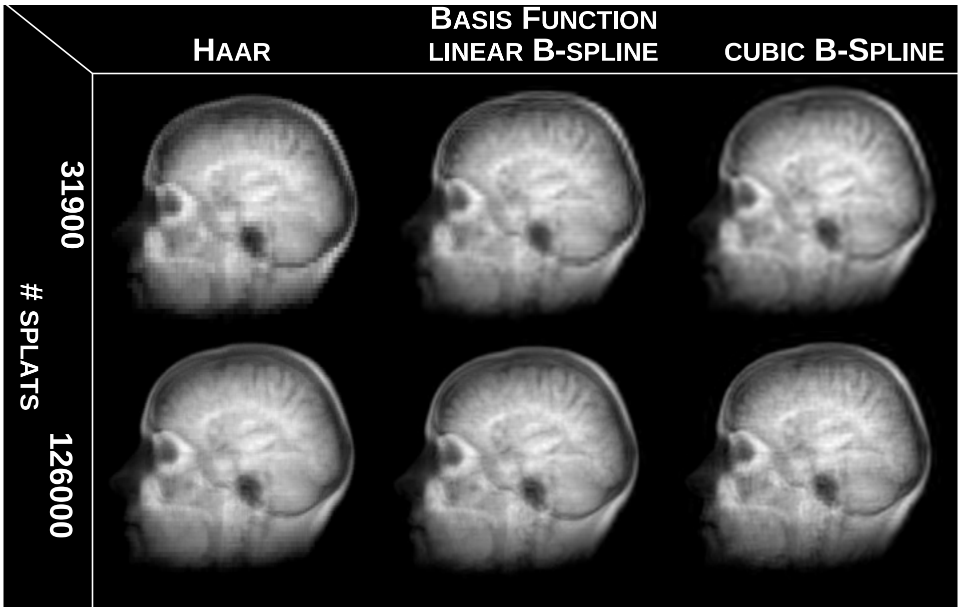


Figure 15: *Images rendered with different basis functions and data-sizes. MRI-data set volume size: 256x128x256, 8 bits/voxel, max. decomposition level M=2. Compressed datasizes (absolute/relative to original data set): upper row: Haar: 69241 bytes (0.825 %), linear: 69662 bytes (0.83 %), cubic: 77724 bytes (0.92 %), lower row: Haar: 273740 bytes (3.26 %), linear: 305972 bytes (3.65 %), cubic: 359603 bytes (4.29 %).*

Table 4: Performance of rendering algorithm:

	RGB				YIQ				COMPUTED BY PCA			
	# COEFF	Q_V in [%]	TIME (INDY) IN [SEC.]	TIME (IMPACT) IN [SEC.]	# COEFF	Q_V in [%]	TIME (INDY) IN [SEC].	TIME (IMPACT) IN [SEC.]	# COEFF	Q_V in [%]	TIME (INDY) IN EC.[SEC.]	TIME (IMPACT) IN [SEC.]
4.6 KBYTE	2155	65.38	0.44	0.15	2175	81.61	0.46	0.15	2184	85.61	0.45	0.15
9.4 KBYTE	4480	82.73	0.9	0.31	4221	86.91	0.93	0.3	3820	88.27	0.86	0.26
79 KBYTE	31714	92.14	3.21	1.14	35655	94.6	3.06	1.09	30554	94.74	2.61	0.9
368 KBYTE	170761	96.82	11.52	4.01	178065	98.22	12.55	3.75	172240	98.47	10.77	3.58

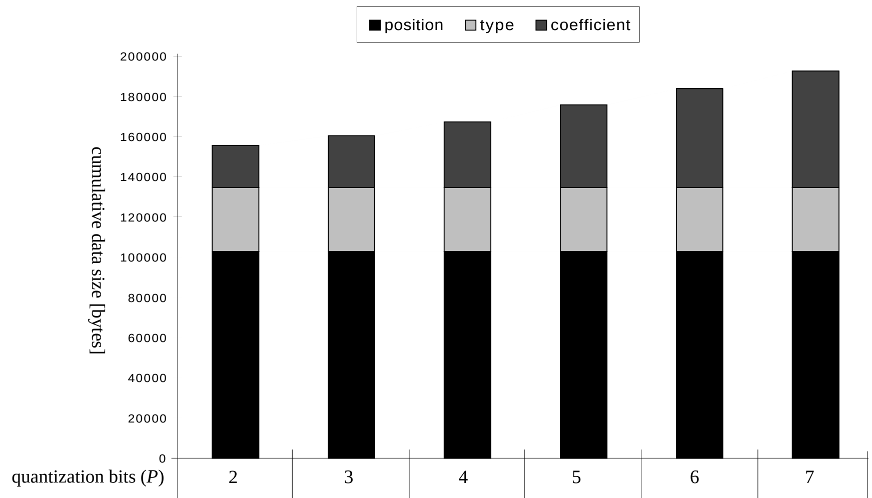
Thus a trade-off between compression gain and rendering time must be found, depending on the hardware used and available network bandwidth. To compare performance we compressed the data set using the lossless Lempel-Ziv coding-scheme, such as implemented in the gzip-procedure. The file size was reduced to 1586754 bytes (compression rate: 18.91 %). However this scheme allows only lossless compression and does not support hierachical decompression and compression domain rendering. Moreover, the data set has to be reconstructed after transmission and requires full volume memory space at the client side. Figure 16 illustrates how the data size can be adjusted by balancing the trade-off between the amount of quantization bits P and relative quantization error Q_E. The error metric we used counts for the relative energy difference of all wavelet coefficients ($w(\boldsymbol{C}) \in \{d^{type}_{mpqr}(\boldsymbol{C}), c_{Mpqr}(\boldsymbol{C})\}$) and their quantized counterparts ($\tilde{w}(\boldsymbol{C})$) defined as:

$$Q_E = \left(\frac{\sum_{\boldsymbol{C} \in \{\boldsymbol{C}_1, \boldsymbol{C}_2, \boldsymbol{C}_3\}} \sum_{w} r(w(\boldsymbol{C}))}{\sum_{\boldsymbol{C} \in \{\boldsymbol{C}_1, \boldsymbol{C}_2, \boldsymbol{C}_3\}} \sum_{w} E(w(\boldsymbol{C}))} \cdot 100 \right) [\%] \tag{20}$$

where the residual term $r(coeff)$ is given as

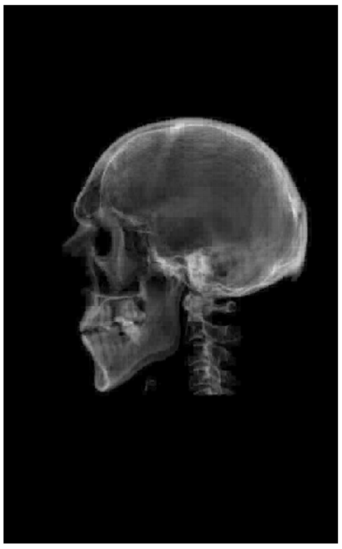
$$r(w(\boldsymbol{C})) = E(w(\boldsymbol{C}) - \tilde{w}(\boldsymbol{C})) \tag{21}$$

Figure 16 shows a breakdown of the data size of a given volume data set displayed on the right side for an increasing number of quantization bits. The total length of each bar represents the overall data size. Each bar is divided into fractions representing the data sizes for positional data, type and coefficient-value. Positional data and type are lossless and hence not influenced by the amount of quantization bits. The figure shows that the data size increases linearly with the number of bits, whereas the quantization error decreases exponentially.



quantization bits (P)	2	3	4	5	6	7
Q_E in [%]	1.674	0.355	0.0806	0.0217	0.000598	0.000229

a)



b)

Figure 16: *Relation between data size and number of bits for quantization (data source: Visible Human Project, CT-data set). Volume size: 256^3, max. decomposition level M=3, no. of splats 74340. a) proportions of coefficient, type and positional data and resulting data size.. b) reference image of compressed data set.*

6. Conclusions and Future Work

We have presented a method for progressive compression domain volume visualization based on a unified wavelet data representation and rendering framework. As an image based approach we don‘t need to fully decompress and store the 3D data set upon rendering. Moreover, the volume can be rendered immediately from the compressed bitstream, which features progressive organization and allows to refine the image according to the provided data. The two essential features *progression* and *compression* make the framework especially suited for networked and distributed environments. In particular the limited memory requirements of our image based method enable to run it even on very low cost computers. In this context the framework offers flexibility in balancing the trade-off between network performance and local computational capabilities of client and server. In addition, since the performance is mostly driven by the underlying CPU capabilities we expect an performance upscaling with each new generation of processors. However, although we proposed some further algorithms to enhance visual quality the method is considered as a fast and low quality rendering technique, which can operate as an interactive volume data browser.

At this point in time we are working on a Java-applet [6] for further evaluations. In addition we will examine the usefulness of an arithmetic coding scheme. Since hardware accumulation boosts the performance significantly, we will provide an OpenGL interface for 3D PC graphics boards. Furthermore the transmission of selectively refined, view-dependent data over a low-bandwidth network will be investigated.

7. Acknowledgments

The authors would like to thank R. Dittrich, A. Eggenberger, A. Hubeli, J. Pittet, and P. Ruser for implementing parts of the framework. This work was supported by the ETH research council under grant No. 41-2642.5.

8. References

[1] C. Busch and M. H. Gross. “Interactive neural network texture analysis and visualization for surface reconstruction in medical imaging.” In R. J. Hubbold and R. Juan, editors, *Eurographics ’93*, pages 49–60, Oxford, UK, 1993. Eurographics, Blackwell Publishers.

[2] B. Cabral, N. Cam, and J. Foran. “Accelerated volume rendering and tomographic reconstruction using texture mapping hardware.” In A. Kaufman and W. Krueger, editors, *1994 Symposium on Volume Visualization*, pages 91–98. ACM SIGGRAPH, Oct. 1994. ISBN 0-89791-741-3.

[3] C. K. Chui and J. Z. Wang. “A cardinal spline approach to wavelets.” *Proc. Amer. Math. Soc.*, 113:785–793, 1991.

[4] R. Crawfis and N. Max. “Direct volume visualization of three-dimensional vector fields.” *1992 Workshop on Volume Visualization*, pages 55–60, 1992.

[5] I. Daubechies. *Ten Lectures on Wavelets*. CBMS-NSF regional conference series in applied mathematics, no.61, SIAM, 1992.

[6] E. E. V. V. Environment). "Homepage." http://www.inf.ethz.ch/personal/lippert/EVOLVE.

[7] K. Fukunaga. *Introduction to Statistical Pattern Recognition. 2nd Ed.*. Academic Press, New York, 1990.

[8] M. H. Gross. "L^2 optimal oracles and compression strategies for semiorthogonal wavelets." Technical Report 254, Computer Science Department, ETH Zürich, 1996. http://www.inf.ethz.ch/publications/tr200.html.

[9] M. H. Gross, L. Lippert, R. Dittrich, and S. Häring. "Two methods for wavelet-based volume rendering." Technical Report 247, ETH-Zürich, 1996.

[10] M. Haley and E. Blake. "Incremental volume rendering using hierarchical compression." *Computer Graphics Forum*, 15(3):44–55, 1996.

[11] G. Held and T. R. Marshall. *Data Compression*. John Wiley & Sons Ltd., 1991.

[12] J. T. Kajiya and B. P. Von Herzen. "Ray tracing volume densities." In H. Christiansen, editor, *Computer Graphics (SIGGRAPH '84 Proceedings)*, volume 18, pages 165–174, July 1984.

[13] A. Kaufman. *Volume Visualization*. IEEE Computer Society Press, 1990.

[14] P. Lacroute and M. Levoy. "Fast volume rendering using a shear–warp factorization of the viewing transformation." In A. Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 451–458. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.

[15] D. Laur and P. Hanrahan. "Hierarchical splatting: A progressive refinement algorithm for volume rendering." In T. W. Sederberg, editor, *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 285–288, July 1991.

[16] M. Levoy. "Display of surfaces from volume data." *IEEE Computer Graphics and Applications*, 8(3):29–37, May 1988.

[17] L. Lippert and M. H. Gross. "Fast wavelet based volume rendering by accumulation of transparent texture maps." In *Proceedings of Eurographics '95*, pages 431–443, 1995.

[18] National Library of Medicine. *The Visible Human Project*. http://www.nlm.nih.gov/research/visible/visible_human.html, 1995.

[19] D. H. Pritchard. "U.S. color television fundamentals – A review." *IEEE Transactions on Consumer Electronics*, 23(4):467–478, Nov. 1977.

[20] E. Quak and N. Weyrich. "Decomposition and reconstruction algorithms for spline wavelets on a bounded inverval." *Applied and Computational Harmonic Analysis*, 1(3):217–231, June 1994.

[21] E. J. Stollnitz, T. D. DeRose, and D. Salesin. *Wavelets for Computer Graphics*. Morgan Kaufmann Publishers, Inc., 1996.

[22] T. Totsuka and M. Levoy. "Frequency domain volume rendering." In J. T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 271–278, Aug. 1993.

[23] M. V. Wickerhauser. *Adapted Wavelet Analysis from Theory to Software*. A. K. Peters, Ltd., 1994.

[24] B. Yeo and B. Liu. "Volume rendering of DCT-based compressed 3D scalar data." *IEEE Transactions on Visualization and Computer Graphics*, 1(1):29–43, Mar. 1995. ISSN 1077-2626.

Compression Domain Volume Rendering for Distributed Environments

L. Lippert, M. H. Gross, C. Kurmann

Department of Computer Science, ETH Zürich

Email: {lippert, grossm, kurmann}@inf.ethz.ch
WWW: http://www.inf.ethz.ch/department/WR/cg

Abstract

This paper describes a method for volume data compression and rendering which bases on wavelet splats. The underlying concept is especially designed for distributed and networked applications, where we assume a remote server to maintain large scale volume data sets, being inspected, browsed through and rendered interactively by a local client. Therefore, we encode the server‘s volume data using a newly designed wavelet based volume compression method. A local client can render the volumes immediately from the compression domain by using wavelet footprints, a method proposed earlier. In addition, our setup features full progression, where the rendered image is refined progressively as data comes in. Furthermore, framerate constraints are considered by controlling the quality of the image both locally and globally depending on the current network bandwidth or computational capabilities of the client. As a very important aspect of our setup, the client does not need to provide storage for the volume data and can be implemented in terms of a network application. The underlying framework enables to exploit all advantageous properties of the wavelet transform and forms a basis for both sophisticated lossy compression and rendering. Although coming along with simple illumination and constant exponential decay, the rendering method is especially suited for fast interactive inspection of large data sets and can be supported easily by graphics hardware.

Keywords: volume rendering, multiresolution, progressive compression, splatting, wavelets, networks, distributed applications

1. Introduction

Volume rendering, in general, has been a very important subfield of research in computer graphics. Since its invention [12], [16], countless algorithms have been proposed, [13], [14], most of which have been designed for providing high quality images at low computational efforts. In this context, recent advancements in graphics hardware help to exploit individual capabilities of graphics workstations [2]. If real time performance constrains the method and image quality can be relaxed, so-called splatting methods [15], [4] have proved to provide good results. Here, footprints of a volume primitive are computed in terms of a small pixmap texture and are superimposed in the framebuffer to carry out the final image. Introducing hierarchical basis functions, such as wavelets [17], rendering is carried out progressively by accumulating scaled and translated versions of self-similar textures.

Unfortunately, most rendering methods introduced so far relate to local computation environments. The problems arising with distributed environments and associated compression domain rendering has hardly been addressed in the literature [24], [10]. However, in times of distributed, world wide information systems and exponentially increasing volume data sets, requirements for rendering such data have changed. In many application scenarios, such as in medical imaging and information systems, situations arise, such as depicted in figure 1.

Often, volume data sets are stored and maintained by a data base server, which can be accessed by one or more local clients via a network. One of the fundamental tasks of volume rendering is to inspect interactively and to browse through the volume data base, where rendering quality can be relaxed for the sake of real time performance. Moreover, being a low end workstation, the computational power and storage capacity of the client are limited and the network‘s

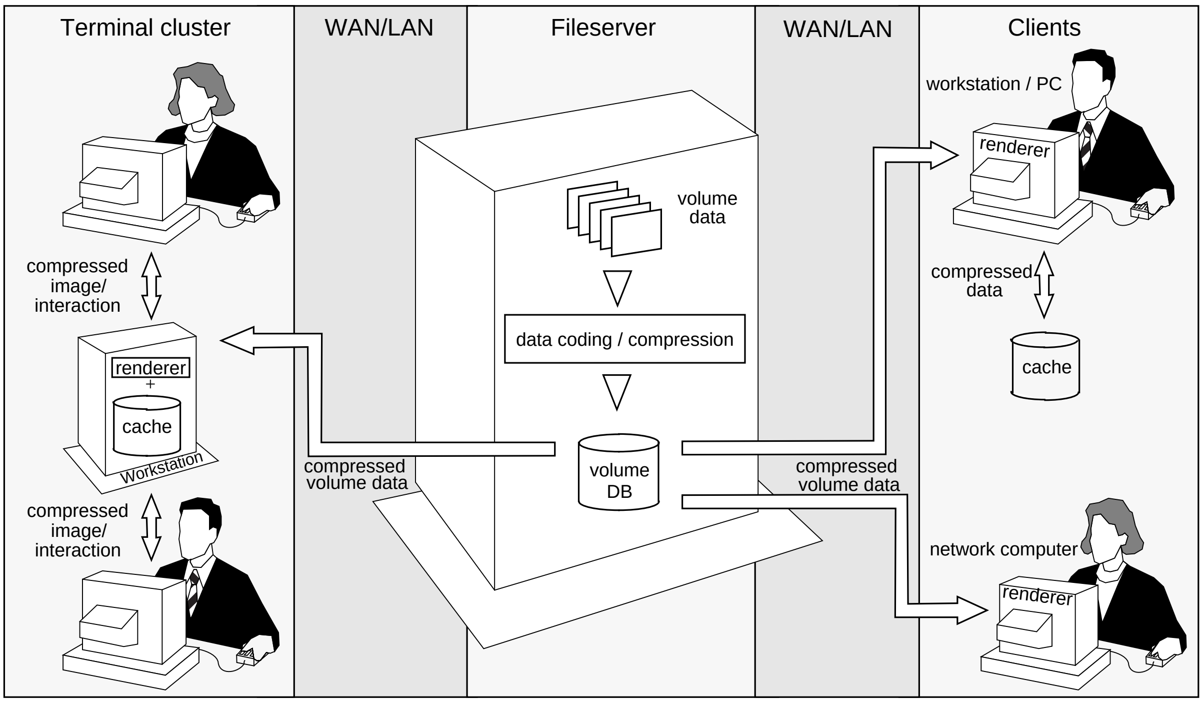


Figure 1: *Volume-rendering in a distributed client/server environment.*

bandwidth and latency might vary with its traffic load. Obviously, when designing a rendering method for these purposes, we have to consider the following issues:

- *Progressivity:* For enhanced interactivity, the image should be refined progressively as data comes in from the remote server. Image quality must be controlled as a function of the network's load and client's hardware setup.
- *Compression domains*: In order to store and transmit large scale volume data sets, compression schemes have to be employed. In order to avoid full decompression, a sophisticated rendering method should carry out computations in the compression domain at the client side.
- *Memory constraints:* In particular with upcoming network computers the capabilities of a local client might be reduced significantly. Hence, a data representation and rendering method is required, which avoids full expansion of the volume in the clients memory.

Along these lines, the research presented here aims at providing a unified framework for data compression and rendering in a distributed environment. Therefore, we first propose a progressive volume data compression scheme. It enables to store both RGB and intensity volume data efficiently on the server in terms of a bitstream and allows fast transmission over the network. To this end, we first decorrelate the RGB volume data and transform the computed intensities in a preprocess by using B-spline wavelets. The coefficients and positions are arranged with decreasing importance and proceed through various steps, such as quantization, encoding and bit allocation. The compression pipeline introduced essentially allows a progressive transmission, where the information lost can be controlled locally and globally. Second, the client carries out all rendering immediately in the compression domain. That is we do not need to perform full decompression of the data while using wavelet splats [17], [9]. To increase performance and visual quality of the images, we propose some significant enhancements of the method. Especially, multiview images can be carried out at low additional costs and constant exponential transfer is incorporated by phase shifts in Fourier domain.

Our paper is organized as follows: For reasons of readability, we briefly review some fundamentals of wavelet based splatting in section 2. In section 3 we describe some significant extensions of the rendering method to gain visual quality, such as multiviews, depth cueing and simple shading. The proposed compression pipeline is described in full detail in section 4. Results obtained with our setup are illustrated in section 5.

2. Fundamentals of Wavelet Splats

In this section we describe how wavelet splats can be defined and used for general volume rendering. Therefore, we briefly summarize the mathematical fundamentals of splat computation following the approach presented in [9]. In addition, a short description of low albedo rendering and wavelets are given. We assume, however, the reader is familiar with the mathematics of wavelets.

2.1. Low Albedo Volume Rendering

Volume rendering can be regarded as a composition of particles that, in addition to their own emission, receive light from surrounding lightsources and reflect these intensities towards the observer.

In classic volume rendering, the amount of light $I(t_L, \boldsymbol{x}, \boldsymbol{s})$ received at point $\boldsymbol{x}$ from direction $\boldsymbol{s}$ up to a ray length t_L is computed as:

$$I(t_L, \boldsymbol{x}, \boldsymbol{s}) = \int_0^{t_L} q(\boldsymbol{x} + t \cdot \boldsymbol{s}) e^{-\int_0^t \alpha(\boldsymbol{x} + s u) du} dt \qquad (1)$$

where q denotes the volume source term and α the opacity function. For an isotropic medium with constant opacity α equation (1) reduces to

$$I(t_L, \boldsymbol{x}, \boldsymbol{s}) = \int_0^{t_L} q(\boldsymbol{x} + t \cdot \boldsymbol{s}) e^{-\alpha t} dt \qquad (2)$$

In particular note that for $\alpha \equiv 0$ we end up in a X-ray-like image.

Especially in those cases, splatting has proved its usability for fast volume rendering [15]. In contrast to ray-casting, it allows to reduce the computational complexity for interpolation and integration to a minimum, since the preprojected footprints of high-order interpolation functions can be stored as lookup tables. The projections themselves can be computed with an accurate quadrature technique. Besides hierarchical splats [15], wavelet splatting [17] is a sophisticated extension.

2.2. B-spline Wavelets

Since we aim at a unified data representation for both volume data compression and rendering, we use Chui and Wang [3] B-spline wavelet bases of order j. They have many usefull properties, such as smoothness and vanishing moments. In addition, analytic expressions for scaling- and wavelet functions and their duals in the frequency- and spatial-domain are given. The mother-scaling function of order 1 is defined as:

$$\phi^{[1]}(x) = \begin{cases} 1 & 0 \le x < 1 \\ 0 & \text{otherwise} \end{cases} \qquad (3)$$

Compactly supported scaling functions of order j can be generated recursively in terms of cardinal B-splines:

$$\phi^{[j]}(x) = \frac{x}{j-1}\phi^{[j-1]}(x) + \frac{j-x}{j-1}\phi^{[j-1]}(x-1) \qquad (4)$$

In the frequency-domain the corresponding functions collapse into *sinc*-polynomials of type:

$$\Phi^{[j]}(f) = \left(\frac{1-e^{-i2\pi f}}{i2\pi f}\right)^j \qquad (5)$$

In the upper term f denotes the frequency and i the complex operator. Note, that the upper equation is fundamental for splat computation in Fourier domain. The semiorthogonality of the function system for $j > 1$ requires duals. For the case of B-spline wavelets all primary and dual functions have linear phase. Furthermore, only rational coefficients have to be used for the fast wavelet transform.

To fully characterize the wavelet transform, we need to determine the wavelet functions. For the corresponding wavelets we obtain:

$$\psi^{[j]}(x) = \sum_{k=0}^{3j-2} q_{j,k}\phi^{[j]}(2x-k)$$
$$\text{with } q_{j,k} = \frac{(-1)^k}{2^{j-1}} \sum_{l=0}^{j} \binom{j}{l} \phi^{[2j]}(k+1-l) \qquad (6)$$

Note, that their support can be computed straightforwardly as [0,2j-1]. Wavelets in three dimensions are obtained easily by non-standard tensor-product extensions [5].

Here volume decomposition is carried out with the duals, whereas reconstruction is performed using the primary functions (4), (6). Note furthermore, that implementations of linear time algorithms [20] can be more challenging, since additional basis transforms might be required.

2.3. Construction of Wavelet Splats

In wavelet splatting, the renderer computes the projection such as defined in (2). Taking into account the wavelet decomposition level m up to M and moving the summation outside the integral, the formulation collapses to:

$$I(\infty, \boldsymbol{X}, \boldsymbol{S}) = \sum_{m=1}^{M} \sum_{\substack{type=1 \\ p,q,r \in Z}}^{7} d_{mpqr}^{type} \int_{-\infty}^{\infty} \psi_{mpqr}^{3,type}(\boldsymbol{x} + \boldsymbol{s}t) dt + \sum_{p,q,r \in Z} c_{Mpqr} \int_{-\infty}^{\infty} \phi_{Mpqr}^{3}(\boldsymbol{x} + \boldsymbol{s}t) dt \qquad (7)$$

where d_{mpqr}^{type} denote the wavelet coefficients of decomposition level m at the spatial position p, q, r and wavelet-type *type* and c_{Mpqr} the coefficient of the scaling function of level M. The computation of the line integrals for a particular view can be accomplished by Fourier projection slicing (FPS). It allows to compute accurate projections of any basis function. This theorem states that the 2D Fourier transform of a projection of a function $f(x,y,z)$ onto a given plane $\boldsymbol{P}$ equals a plane that slices the Fourier transform $F(\omega_1, \omega_2, \omega_3)$ parallel to $\boldsymbol{P}$ and intersects the origin.

Since many wavelet types such as B-splines come along with closed form representations in the frequency domain, it is straightforward to apply this theorem to get the required splats. Figure 2 depicts the setting, where an inverse FFT processes the slices to obtain the wavelet splat.

The intersection plane spanned by ***u***, ***v*** defines the 2D Fourier transform of the texture splats $I(u, v) = F(\omega_1(u, v), \omega_2(u, v), \omega_3(u, v))$, whereas the normal vector ***n*** of the plane equals the direction of the projection. The definition of the viewing parameters is figured out in spherical coordinates (α,β).

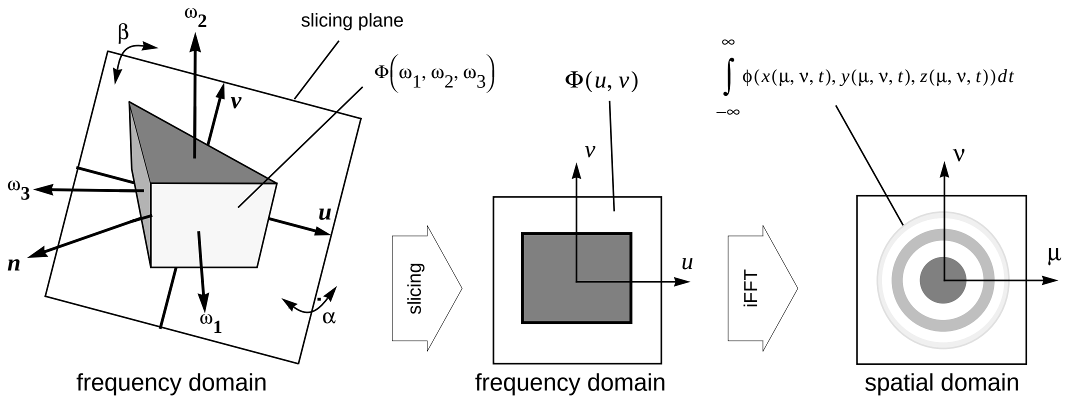


Figure 2: *Illustration of the Fourier projection slicing theorem in 3D for an idealized Shannon wavelet.*

Once the renderer builds the viewpoint dependent integral tables, the screen position of the table is calculated, mapped, weighted by the wavelet coefficient and accumulated into the framebuffer. Since the basis functions of different iteration levels *m* differ by dilation, only eight different splats of depth *M* have to be calculated. All other footprints are derived by subsampling in the spirit of a mipmap. Correct sampling and optimized data-structures of the calculated splats are discussed in [9].

3. Visual Enhancements

We are now in a position to discuss two significant rendering features which help to improve the visual quality of the generated images. In particular, we introduce a method for the computation of multiviews, being important in many applications, such as medical imaging. Exploiting the symmetry of 3D tensor-product wavelets allows the additional costs to be kept sufficiently low. Furthermore, we address the problem of exponential transfer functions in Fourier space. Phase shifts of the wavelet's FT enable exponential transfer to be incorporated and can be used to simulate shading operations on the fly.

3.1. Multiviews

The rendered images, obtained by our splatting approach, lack occlusion. Since this important visual cue is missing for the calculated X-ray images, depth information is not presented to the user. One way to overcome this drawback is the introduction of a multiview arrangement. Here, the volume is rendered simultaneously from different directions and presented to the user in a single window. Exploiting the coherence of different viewing angles given by the symmetry of tensor-product constructions, the basic single-view splatting approach can be extended to a multiview renderer without computational overhead for splat computation. We recall that the tensor-product functions are constructed from permutations of the 1D basis functions ϕ and ψ along the *x*, *y* and *z* directions.

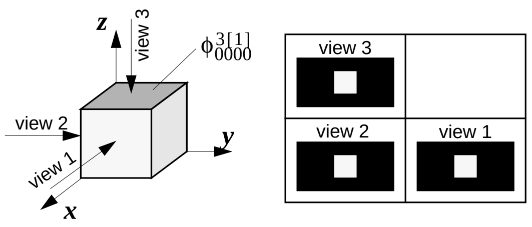


Figure 3: *Illustration of the multiview concept.*

Figure 3 exemplifies the idea. The footprint of the basis function $\phi^{3[1]}_{0000}$ (3D Haar scaling function of level 0, aligned to the origin) from the first viewing-direction equals the footprint from the second and third direction, e.g. the calculated footprints can be reused for rendering different views. Thus, we propose to generate a lookup table for the basis functions types 1 to 7 to ensure correct footprint calculation. A corresponding lookup table is given in Table 1 and illustrated in figure 4 for Haar functions respectively.

Table 1: Permutations and negations for multi view arrangements

	PERMUTATIONS	NEGATIONS
VIEW 1	[0 1 2 3 4 5 6 7]	[1 1 1 1 1 1 1 1]
VIEW 2	[0 1 4 5 2 3 6 7]	[1 1 -1 -1 1 1 -1 -1]
VIEW 3	[0 4 2 6 1 5 3 7]	[1 1 1 1 -1 -1 -1 -1]

In order to combine the three footprints for the eight different basis functions, eight textures are calculated for view 1 by the FPS-theorem. The footprint for the basis function $\phi\phi\psi$ is equal for view 1 and 2, whereas the splat of the view 3 is given by the footprint of the function $\psi\phi\phi$ (texture 4) of view 1. The corresponding permutations are given for all basis functions and for three views in Table 1. In some cases it is also necessary to multiply the intensity with -1. The same permutation tables can also be used for higher order wavelets.

Note that these permutations are equal for each viewing direction. Note furthermore that the angles between the individual viewing planes are not constant. Moreover, they depend on the initial selection of the camera parameters. A little algebra reveals that the three projection-planes P_i, spanned by the vectors $(\boldsymbol{u}_i, \boldsymbol{v}_i)$ are given by

View 1: $(\boldsymbol{u}_1, \boldsymbol{v}_1)$ with

$$\boldsymbol{u}_1 = \begin{bmatrix} -\sin\alpha \\ \cos\alpha \\ 0 \end{bmatrix} = \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix} \qquad \boldsymbol{v}_1 = \begin{bmatrix} -\cos\alpha\sin\beta \\ -\sin\alpha\sin\beta \\ \cos\beta \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}$$

View 2: $(\boldsymbol{u}_2, \boldsymbol{v}_2)$ with

$$\boldsymbol{u}_2 = \begin{bmatrix} -\cos\alpha \\ -\sin\alpha \\ 0 \end{bmatrix} = \begin{bmatrix} -u_y \\ u_x \\ u_z \end{bmatrix} \quad \boldsymbol{v}_2 = \begin{bmatrix} -\sin\alpha\sin\beta \\ \cos\alpha\sin\beta \\ \cos\beta \end{bmatrix} = \begin{bmatrix} -v_y \\ v_x \\ v_z \end{bmatrix}$$

View 3: $(\boldsymbol{u}_3, \boldsymbol{v}_3)$ with

$$\boldsymbol{u}_3 = \begin{bmatrix} 0 \\ \cos\alpha \\ \sin\alpha \end{bmatrix} = \begin{bmatrix} u_z \\ u_y \\ -u_x \end{bmatrix} \quad \boldsymbol{v}_3 = \begin{bmatrix} -\cos\beta \\ -\sin\alpha\sin\beta \\ \cos\alpha\sin\beta \end{bmatrix} = \begin{bmatrix} -v_z \\ v_y \\ -v_x \end{bmatrix}$$

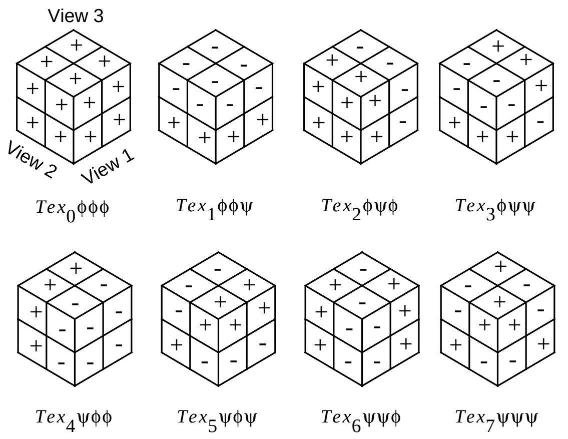


Figure 4: *Different wavelet types figured out by non-standard tensor-product extensions for the Haar case.*

The triple view rendering is illustrated in figure 5, where a classified data set is displayed from three directions. The image was grabbed directly from the screen as it is displayed to the user. Skin is colored white, brain tissue red and a tumor is colored in blue.

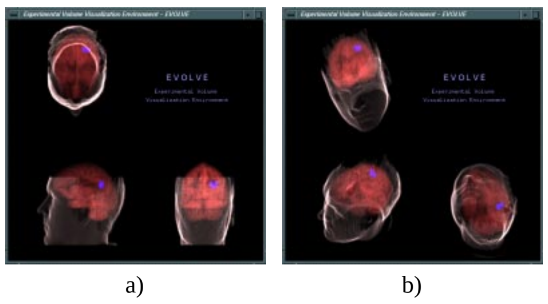


Figure 5: *Triple views seen from different viewing angles as applied to classified MR-data set (volume size: 256 x 128 x 256, max. decomposition level M = 2) a)* $\alpha = \beta = 0.0$ *b)* $\alpha = 0.66$, $\beta = 0.91$.

3.2. Depth Cueing

Up to now, only linear depth cueing has been considered in the frequency domain [22]. In order to compute constant exponential decay as in (2) we propose a depth cueing approach implemented as a two step solution. In the first step the cued footprints have to be calculated by the FPS-theorem and an additional phase-shift. Second, the textures have to be weighted according to the distance between the projection plane and the basis functions.

Footprint Computation In order to set up the Fourier relations for exponentially weighted functions, let's first consider the relations in the spatial domain. Let $h(t)$ denote the function that is depth cued along t and let k be the constant exponential extinction coefficient. We obtain the new function $\tilde{h}(t)$ as:

$$\begin{aligned} \tilde{h}(t) &= h(t)e^{kt} \\ &= h(t)e^{-i2\pi f_0 t} \quad \text{where } f_0 = \frac{k}{-2\pi i} = \frac{ki}{2\pi} \end{aligned} \tag{8}$$

In the frequency domain the Fourier transform of $\tilde{h}(t)$ can be written as:

$$\begin{aligned} \tilde{H}(f) &= \int_{-\infty}^{\infty} \tilde{h}(t)e^{i2\pi ft}dt \\ &= \int_{-\infty}^{\infty} h(t)e^{i2\pi(f-f_0)t}dt \\ &= H(f-f_0) \end{aligned} \tag{9}$$

Obviously, the Fourier transform of the exponentially depth-cued function $\tilde{h}(t)$ is computed by shifting $H(f)$ with the imaginary frequency f_0. This can be interpreted as a “*frequency phase shift*”. For the B-spline scaling functions and wavelets of order j in the frequency domain we have

$$\tilde{\Phi}^{[j]}(f) = \left(\frac{1-e^{-i2\pi(f-f_0)}}{i2\pi(f-f_0)}\right)^j$$
$$\tilde{\Psi}^{[j]}(f) = R^{[j]}\left(e^{-i\pi(f-f_0)}\right)\tilde{\Phi}^{[j]}\left(\frac{f}{2}\right) \tag{10}$$

where

$$R^{[j]}\left(e^{-i\pi(f-f_0)}\right) = \frac{1}{2}\sum_{n=-\infty}^{\infty} q_{j,n}\cdot\left(e^{-i\pi(f-f_0)}\right)^n \tag{11}$$

In 3D the direction of the shift operation in the spectrum can be chosen without any restriction and independently from the viewing direction. It corresponds to the depth-cueing direction in the spatial domain. We define the cueing vector $\boldsymbol{L}$ for any pair of viewing angles (α_l, β_l) as

$$\boldsymbol{L}(\alpha_l,\beta_l) = |\boldsymbol{k}|\begin{bmatrix}-\cos\alpha_l\cos\beta_l\\ \sin\alpha_l\cos\beta_l\\ \sin\beta_l\end{bmatrix} = \begin{bmatrix}k_x\\ k_y\\ k_z\end{bmatrix} \tag{12}$$

to determine the cueing direction and extinction coefficient. Figure 6 shows the calculated splats resulting from this method.

Distance Weighting For correct depth cueing of the volume, the computed footprints have to be weighted according to their distance from a reference plane which can be regarded as a planar light source, perpendicular to the depth-cueing vector $\boldsymbol{L}$ that points towards the volume midpoint $\boldsymbol{V}_M$. Let $\boldsymbol{L}_{init}$ represent the intersection point of the volume’s bounding sphere and its tangent plane, the weighting factor ω_d for each basis function centered at $\boldsymbol{B}_M = \boldsymbol{P} + \boldsymbol{L}_{init}$ can be computed as:

$$\omega_d = e^{-dist} \quad \text{where} \quad dist = \frac{\boldsymbol{L}\cdot\boldsymbol{P}}{|\boldsymbol{L}|} \tag{13}$$

During the accumulation process this factor is multiplied with the wavelet coefficient and the accumulation step proceeds straightforwardly. The geometric relationships are depicted in figure 7.

As a result we come up with smooth and correctly depth-cued volumes, such as shown in figure 6.

To illustrate the performance of our method, we applied the approach to the visible human CT-data set [18]. The influence of different factors $|\boldsymbol{k}|$ and different cueing directions are displayed in figure 8 Note that exponential depth cueing increases the rendering performance, since less textures have to be accumulated. Moreover, interactive manipulation of the cueing direction allows us to simulate simple shading operations in the wavelet domain and enhances the visual quality significantly.

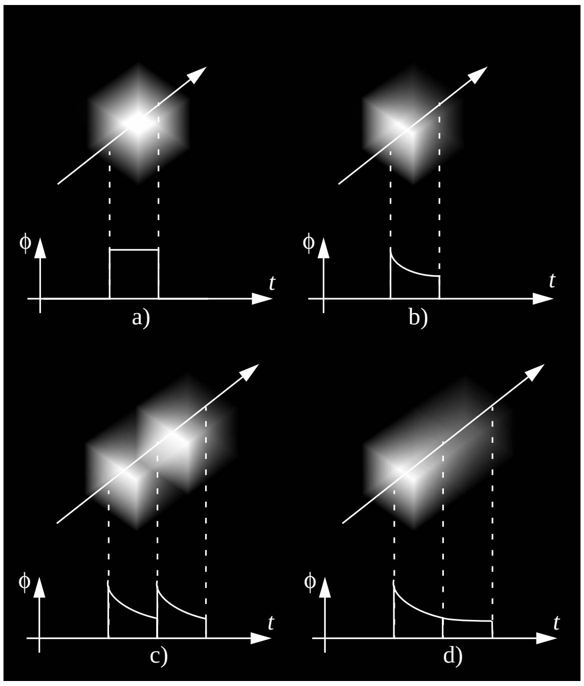
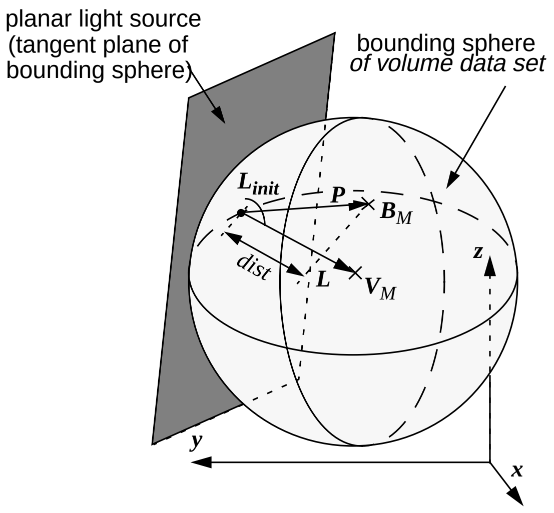


Figure 6: *Exponentially depth-cued scaling function (j=1) a) |**k**|=0.0. b) |**k**|=0.6. Distance weighted footprints: c) depth cued splats without distance weighting, d) corrected splats using distance weighting.*



planar light source (tangent plane of bounding sphere)
bounding sphere of volume data set
L_init
P
B_M
dist
L
V_M
z
y
x

Figure 7: *Geometric relationships for distance-weighting of basis functions.*

4. Progressive Compression and Transmission

As motivated earlier, our approach is targeted at networked applications where, for instance, a local client with low computational power browses through a remote database. Thus, in order to transmit our volume data efficiently we have to find appropriate compression strategies. It is clear that the underlying framework of the wavelet decomposi-

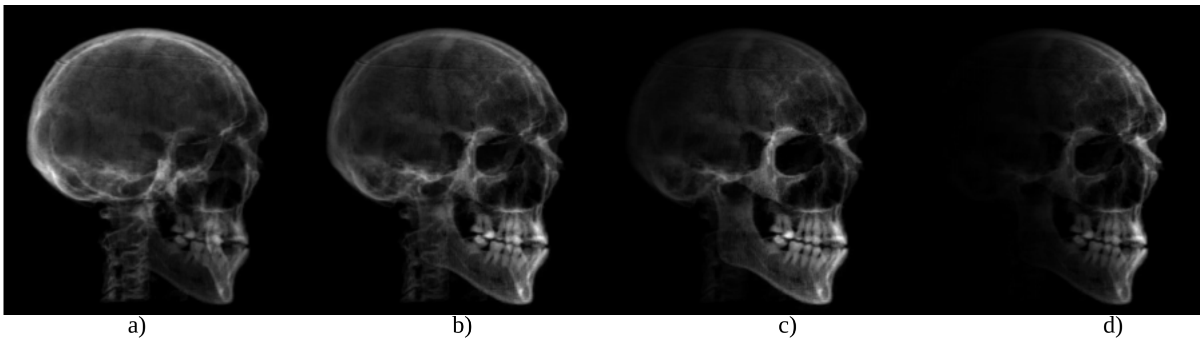


Figure 8: *Influence of different cueing parameters and directions.(data source: Visible Human Project, courtesy National Library of Medicine, copyright (C) 1995). Volume size: 256^3, max. decomposition level M=3. a-c):* $\alpha = \alpha_l = 2.5$, $\beta = \beta_l = 0.1$, *a)* $|\mathbf{k}| = 0$, *b)* $|\mathbf{k}| = 0.1$, *c)* $|\mathbf{k}| = 0.2$, *d)* $\alpha = 2.5$, $\alpha_l = \pi/4$, $\beta = 0.1$, $\beta_l = 0.3$, $|\mathbf{k}| = 0.2$.

tion proposes to develop an optimized compression technique that allows progressive transmission, one pass decompression and direct rendering at interactive frame rates. Moreover, the wavelet domain rendering avoids full decompression of the data prior to the splatting which itself as an image based method does not require to store the full volume data at the client's side. Although much research has been done on wavelet compression methods [21], [23] the specific needs of compression domain rendering encouraged us to develop a new compression pipeline which will be explained below. For the sake of brevity we restrict our description to the issues essential for our method and recommend further readings in the fundamentals of data compression [11].

4.1. Overview

Figure 9 illustrates the data flow in our compression and rendering setup. The data preprocessing comprises five stages. It enables both intensity and RGB volumes to be handled, which might be the result of an optional data classification step [1]. In case of RGB volumes the second step consists of a colorspace optimization which essentially decorrelates the data and allows color sensitive quantization. Next, a wavelet transform is performed independently on the three channels. Lossy compression is carried out by an oracle [8] which operates locally or globally in the wavelet domain. The final step includes a data compression and encoding scheme to achieve a binary output stream that can be stored locally or transmitted directly through a network. Note that forward compression does not have any real-time constraints as opposed to the decompression.

In our implementation the software decompressor works at a rate of ~30 k wavelet-coefficients/sec. on an Indy R4400 workstation and allows on-line decompression. The renderer splats the computed footprints weighted with the decoded wavelet-coefficients directly into a software or hardware accumulation buffer. In addition, our framework incorporates a local cache to store coefficients at the client's side. The required splats are computed locally. Since the wavelet coefficients are transmitted in significance order the rendering quality is fully controlled by a user-defined framerate, the client's hardware and, if no cache mechanism is enabled, also by the bandwidth of the network. Hence, this concept balances CPU, network and graphics performance and allows scalability. In a minimum configuration we have to provide client storage only for the eight mother-wavelet splats and three Huffman tables. Together they take less than 5KB of memory even for huge data sets. This even allows the scheme to run on some of the upcoming network computers.

4.2. Colorspace Transformations

If the initial volume is given in RGB, it is critical to transform the volume into an optimized colorspace prior to compression. Here, we assume the optimized space to be spanned by the three vectors $\mathbf{C}_1$, $\mathbf{C}_2$ and $\mathbf{C}_3$. This allows to assign an additional significance to each vector. As a result we get two independent significance weights per coefficient which affect encoding and quantization. The first weight is defined by the energy of the associated function [8]. The second one is a global significance determined by the colorspace-coordinates. For instance, a coefficient with the coordinates (1,0,0) is regarded as more relevant than a coefficient (0,1,0) if the vector $\mathbf{C}_1$ is considered to be more significant than $\mathbf{C}_2$.

In addition to RGB we employ two colorspaces: The first one is data-independent and equals the YIQ-colorspace obtained by a simple matrix transform [19]. The Y component encodes the luminance information, whereas the chromaticity is encoded in I and Q. We followed the NTSC bandwidth conventions and assigned a factor 4 to Y, 1.5 to I and 0.6 to Q. In practical use this colorspace allows to compute a black and white image (Y) as a rough sketch and to refine color progressively. Thus, progression is figured out both in the spatial *and* in color domain. Alternatively, our second colorspace is calculated by a statistically optimal principle component analysis (PCA) or Karhunen-Loève expansion [7] whose matrix has to be computed individually for each data set. Although the solutions of the eigenproblems are computationally more challenging on offline forward compression, yet they provide better results (see section 5). In this case the absolute values of the eigenvalues are taken to describe the significance of the corresponding eigenvector. Note that this step can be skipped for intensity volumes, such as raw CT or MRI data sets.

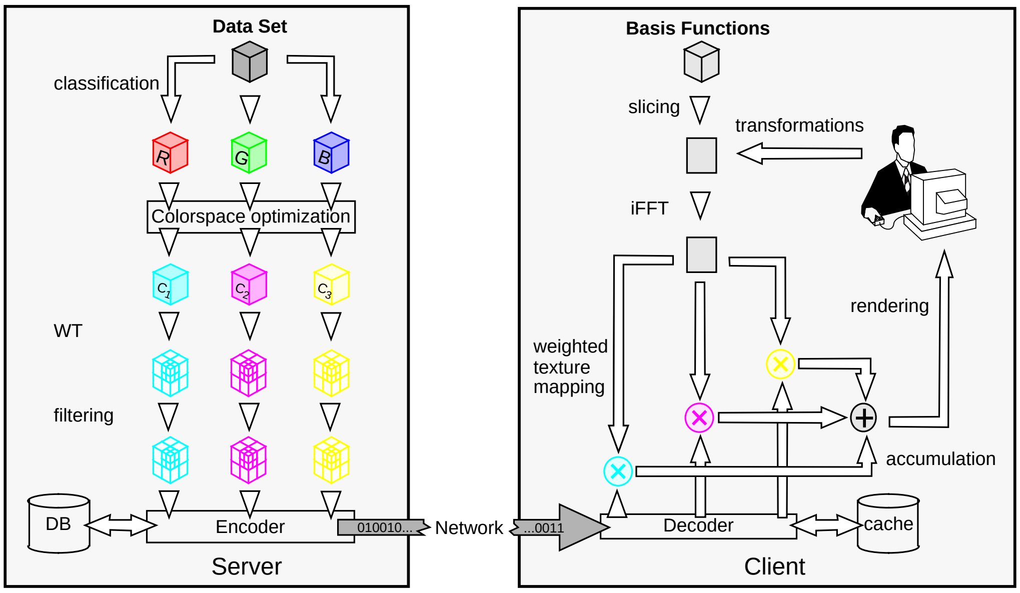


Figure 9: *Setup for distributed compression domain rendering.*

4.3. Data Compression Pipeline

The algorithmic steps for data compression are executed sequentially in the pipeline summarized in figure 10 and convert the wavelet transformed data sets into a sequential bitstream.

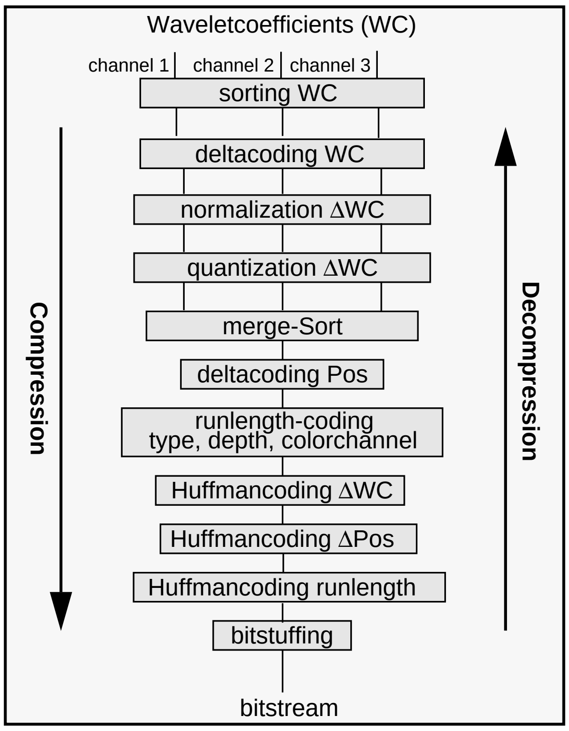


Figure 10: *Pipeline representing the individual steps of the compression scheme.*

Therefore, we start with a sorting operation that generates a sequence of coefficients and positional data in significance order. The significance score S is determined for each color channel $\mathbf{C} \in \{\mathbf{C}_1, \mathbf{C}_2, \mathbf{C}_3\}$ and coefficient $w(\mathbf{C}) \in \{d^{type}_{mpqr}(\mathbf{C}), c_{Mpqr}(\mathbf{C})\}$ individually according to its associated wavelet energy E and color channel importance I.

$$S(w, \mathbf{C}) = E(w(\mathbf{C})) \cdot I(\mathbf{C}) \tag{14}$$

Table 2 summarizes the importance factors for the three different colorspaces.

Table 2: Colorspace significance assignment

COLORSPACE	IMPORTANCE *I*		
	$I(C_1)$	$I(C_2)$	$I(C_3)$
RGB	1	1	1
YIQ	4	1.5	0.6
EIGENVECTORS (PCA)	EIGEN-VALUE 1	EIGEN-VALUE 2	EIGEN-VALUE 3

For each color channel ($\mathbf{C}$), wavelet type (*type*) and decomposition level (m) a deltacoding, normalization and quantization operation is performed separately depending on the individual ranges of the coefficients w. More precisely, if quantization is restricted to P bits for a given sequence of *N*+1 significance sorted non-zero wavelet and scaling function coefficients $(w_n(\boldsymbol{C}, m, type))_{n=0,\ldots,N}$, we compute the coefficient's delta factor $\Delta(\boldsymbol{C}, m, type)$ as:

$$\Delta(\boldsymbol{C}, m, type) = \frac{\max\limits_{n=0,\ldots,N-1}(|w_n(\boldsymbol{C}, m, type)| - |w_{n+1}(\boldsymbol{C}, m, type)|)}{2^P} \tag{15}$$

This factor is used to store the coefficient's range with respect to the assigned bytes. Thus, it has to be transmitted once for each wavelet type, decomposition level and color coordinate. In contrast, the normalized and quantized difference of two coefficients δ has to be transmitted for each coefficient individually. It is computed as:

$$\delta(\boldsymbol{C}, m, type, n) = round\left(\frac{(|w_n(\boldsymbol{C}, m, type)| - |w_{n+1}(\boldsymbol{C}, m, type)|)}{\Delta(\boldsymbol{C}, m, type)}\right) \tag{16}$$

Upon reconstruction we end up with approximated wavelet and scaling function coefficients $\tilde{w}_n(\boldsymbol{C}, m, type)$, $n = 1, \ldots, N$ which can be computed by:

$$\tilde{w}_{n+1}(\boldsymbol{C}, m, type) = sign(n+1)(|\tilde{w}_n(\boldsymbol{C}, m, type)| - \delta(\boldsymbol{C}, m, type, n) \cdot \Delta(\boldsymbol{C}, m, type)) \tag{17}$$

Since the coefficients are sorted according to their individual scores we optimize the residual approximation error as a function of the parameters introduced above. Note that the sign of each coefficient is encoded by an additional flag, refered as *sign*(*n*).

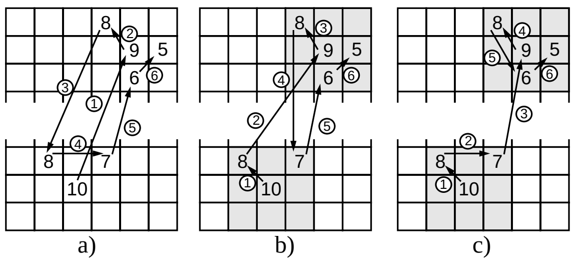


Figure 11: *Effect of spatial clustering:*
a) Without clustering
b) With clustering (position threshold: 3, coefficient threshold = 2)
c) With clustering (position threshold: 3, coefficient threshold = 4).

We choose to limit the quantization *P* to eight bits, since standard framebuffers use eight bits for RGBα each. However, observations in practice encourage us to reduce quantization to even three bits without significant lost of visual quality (see section 5). The three data sequences are merged and sorted according the scores of their wavelet coefficients. In particular, the sorted sequence requires for each coefficient to encode additionally its spatial position, wavelettype and color-channel in a lossless scheme. In order to overcome the drawbacks in compression performance arising from this requirement we introduce the following spatial clustering mechanism, which balances spatial and score constraints upon compression: Within a predefined radius around the coefficient with the highest score the algorithm selects a neighbor as a successor in the bitstream if its score exceeds a threshold. If not, the coefficient with the highest score is selected independently of its spatial position. The corresponding coefficients are labeled to avoid multiple storage. Figure 11 illustrates the ordering with and without clustering for the 2D case. The calculated coefficient weights are given in the corresponding box and the arrows denote the sorting order. Without clustering the arrow length and therefore the positional deltas are nearly equiprobable, whereas for clustered volumes mostly short arrows (small differences in position) appear. This reduces the entropy and brings up a better compression rate of the Huffman-coded positional deltas. That is we balance spatial coherence and the energy contribution sorting order of the coefficients. Note that clustering also speeds up the rendering process, since splats outside the field of view can be detected easily and skipped without further computation.

Additional runlength-codings of wavelettype, decomposition depth *m* and colorchannel are performed and transmitted as variable length Hufman tag codes. The third Huffman-table encodes the deltas δ of wavelet-coefficients. These three tables together take about 2kBytes and have to be transmitted separately prior to the data. In addition the transmitted meta-data includes information about the basis vectors of the colorspace, maximum depth of the wavelet transform (*M*), exact initial wavelet coefficients $(w_0(\boldsymbol{C}, m, type))$ and the coefficient delta factors $(\Delta(\boldsymbol{C}, m, type))$.

The reconstruction scheme has to decode all required information, such as the spatial position, wavelet type, depth *m*, colorchannel and the data value. According to the composition step, each tag is encoded as described in Table 3. For computational efficiency, we propose to precompute 10-bit Huffman look-up tables.

Table 3: Coding schemes:

TAG	CODING SCHEMES
SPATIAL POSITION	SPATIAL CLUSTERING, DELTA CODING, HUFFMAN-TABLE
TYPE	RUNLENGTH, HUFFMAN-TABLE
DEPTH	RUNLENGTH, HUFFMAN-TABLE
COLOR CHANNEL	RUNLENGTH, HUFFMAN-TABLE
SCALAR FACTOR (SIGN)	SINGLE BIT
SCALAR FACTOR (VALUE)	DELTA CODING, HUFFMAN-TABLE

Figure 12 illustrates a fraction of the bitstream as generated by our method. Note, that the number of bits varies as a function of the individual Huffman codes.

5. Results

To investigate the performance of the proposed method, we applied the approach to the RGB-Visible Human Dataset of size 128x128x128 voxels (3 x 8 bits/voxel). In addition to

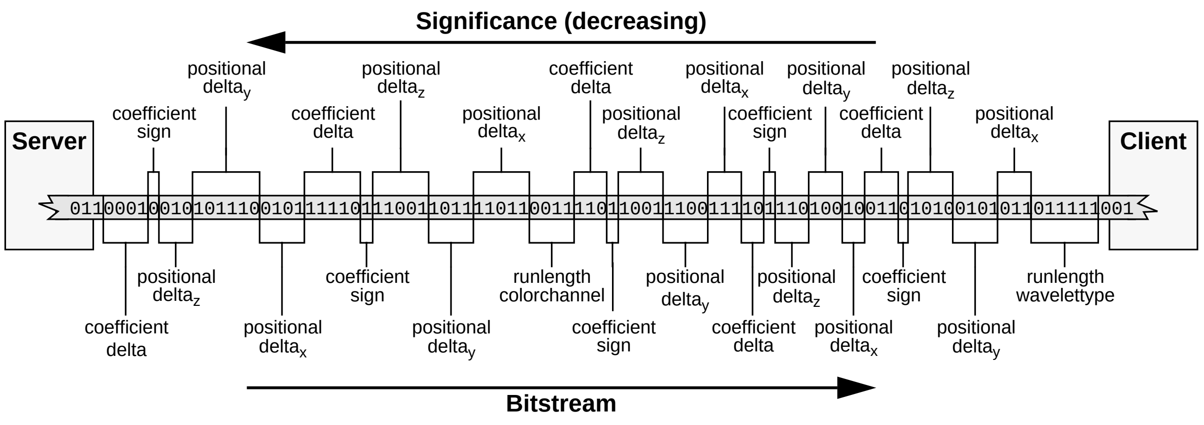


Figure 12: *Fraction of the bitstream generated by the compression scheme*

the RGB data set we generated a scalar representation of the volume by extracting the intensities *Y* from the RGB values. The wavelet decomposition was performed with Haar wavelets up to level *M*=3.The effects of lossy data compression of the Y-data set are illustrated in figure 13. In order to quantize image quality we define an L^2 image measure Q_I conforming to the signal-noise ratio (*SNR*) in [dB] well known from signal processing applications as:

$$Q_I = 20 \cdot \log_{10}\left(\frac{\sum_{pixel} \sum_{color} [i_{\mathrm{ref}}(pix, col)]^2}{\sum_{pixel} \sum_{color} [i_{\mathrm{im}}(pix, col) - i_{\mathrm{ref}}(pix, col)]^2} \right) \quad (18)$$

where, $i_{\mathrm{im}}(pix, col)$ denotes the intensity of a given *pix*el of the computed image for the *col*or-component in RGB-colorspace or the intensity *Y* for the Y-data set, e.g. $col \in \{R, G, B\}$ or $col \equiv Y$ respectively. Note specifically that in image compression ratios > 40 dB refer to reasonable visual qualities and at ratios >60 dB images are perceived as „noise-free“. The reference image was generated by the proposed splatting method for *M*=0 and 100% of the coefficients. We observe that ratios >60 dB are achieved at compression gains of almost 95 %.

The colorplates in figure 14 display the image quality achieved from the RGB data set for different colorspaces and compression rates with respect to the original data size of 6291456 bytes. The qualitative differences of the three colorspaces reveal mostly for small datasizes. Note that YIQ favourises the *Y*-component and renders greyscale images at high compression rates. This is contrasted by the RGB color space where the method reconstructs the volume both in the spatial and colorspace domain and ends up in a poorer image quality. Finally the best results are obtained by the PCA-based color representations. However, as progression proceeds the representations converge to each other. It is clear that the entropy of the color information is lower than in the *Y* channel. Therefore, we observe higher compression gains (SNRs) in figure 14 than in figure 13. Nevertheless even on intensity based compression the results are compelling and enable to use compressed versions of the volume as visual abstracts in large data bases.

We consider the volume’s L^2 approximation energy measure Q_V which is computed relatively to the uncompressed volume from the underlying framework of the WT as:

$$Q_V = \left(\frac{\sum_{n=0}^{N} E(w_n)}{\sum_{n=0}^{volume\ size - 1} E(w_n)} \cdot 100 \right) \quad [\%] \quad (19)$$

The PCA-based colorspace turns out to give the best L^2 approximations for a predefined compression rate as depicted in table 4.

This table also sums the performance and fidelity of our algorithm. Timings are given for a SGI-Indy workstation (MIPS R 4400/150 MHz) and a SGI Maximum Impact workstation (MIPS R10000/195 MHz). Both workstations use our software-accumulation scheme as introduced in [9]. The resolution of the rendered image was 160x180 pixels. For the delta-coding of the wavelet coefficients we assigned three bits. The timings reveal, that we still achieve interactive framerates for fast previewing. Note, in particular that competitive high quality renderers, such as shear warp factorization [14] are significantly slower at these data sizes and require careful setting of the transfer function for speed-up. Our proposed splatting technique is well-suited for hardware support [17]. The hardware assisted accumulation of the calculated splats is done within the accumulation buffer or uses alpha-blending operations, depending on the available hardware platform. Hardware support allows to further increase the rendering speed significantly, especially for the generation of high resolution images.

In order to investigate compression gain achieved by higher order spline wavelets we compared in Figure 15 the Haar transform to linear and cubic B-spline wavelets, where the MRI-volume data set of size 256x128x256 (8 bits/

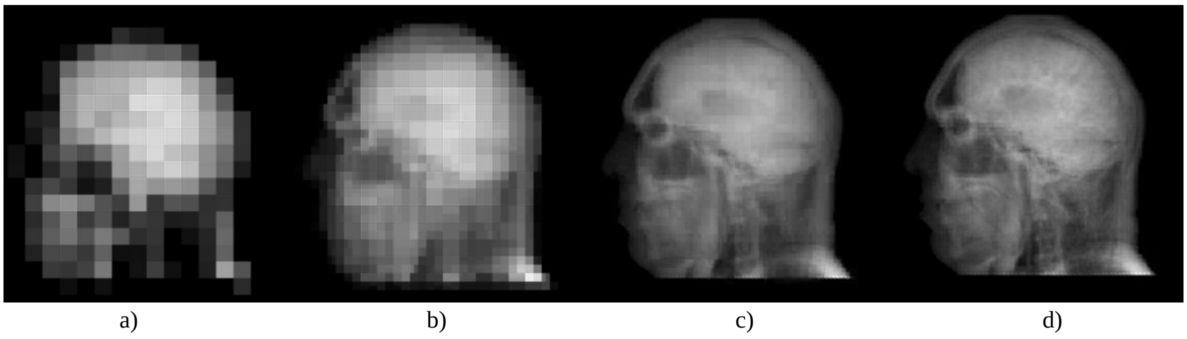


Figure 13: *Compression of intensity data set. (data source: Visible Human Project, RGB-data set converted to Y). a) data size: 1496 bytes (rel. data size: 0.07 %), Q_I = 19.19 dB. b) data size: 3164 bytes (rel. data size: 0.15 %), Q_I = 31.23 dB. c) data size: 26198 bytes (rel. data size: 1.25 %), Q_I = 48.71 dB. d) data size: 122680 bytes (rel. data size: 5.85 %), Q_I = 61.17 dB.*

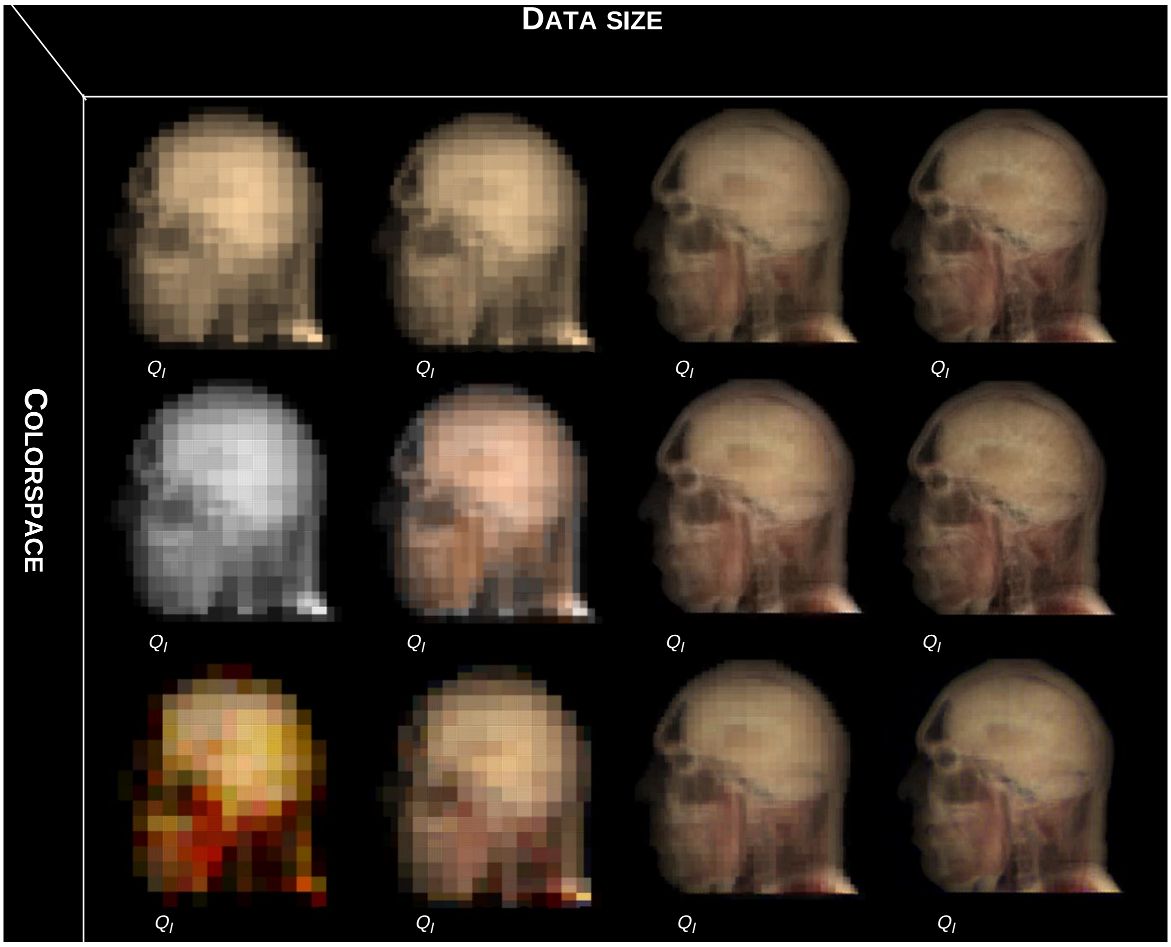


Figure 14: *Progressive compression in three different colorspaces. (data source: Visible Human Project, RGB-data set). Volume size: 128^3, max. decomposition level M=3.*

voxel) is displayed. The wavelet transform was performed up to *M*=2. The blocky structure of the Haar wavelet clearly leads to a blocky representation of the data set whereas the higher order wavelets come up with a smooth, somewhat blurry representation. Obviously, increasing numbers of vanishing moments lead to higher compression rates. A major drawback of higher order wavelets however is the increasing support which affects the splat size. As a consequence both FFT computation and the footprint accumulation are computationally more expensive. Furthermore the performance of spatial clustering drops in efficiency with increasing support.

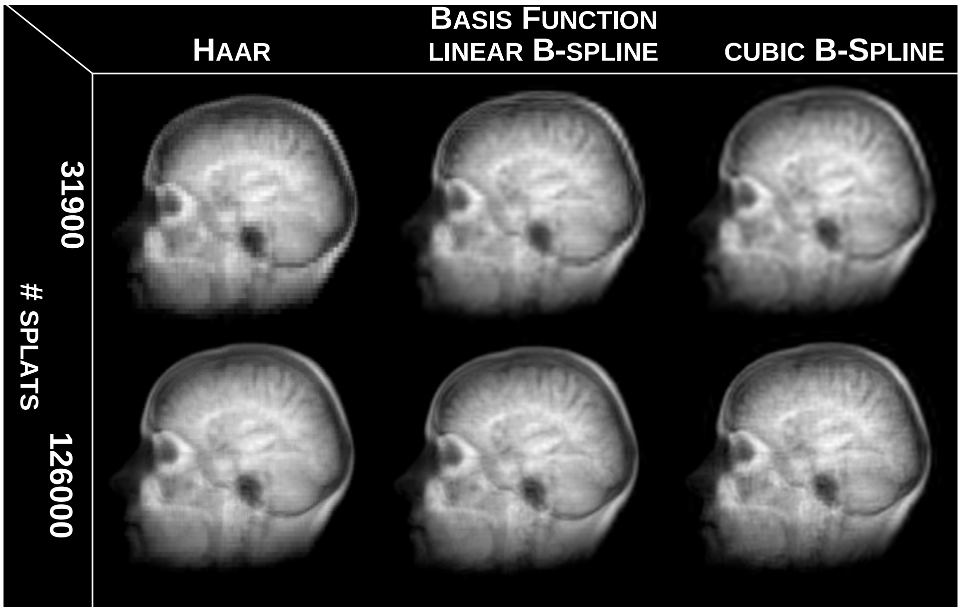


Figure 15: *Images rendered with different basis functions and data-sizes. MRI-data set volume size: 256x128x256, 8 bits/voxel, max. decomposition level M=2. Compressed datasizes (absolute/relative to original data set): upper row: Haar: 69241 bytes (0.825 %), linear: 69662 bytes (0.83 %), cubic: 77724 bytes (0.92 %), lower row: Haar: 273740 bytes (3.26 %), linear: 305972 bytes (3.65 %), cubic: 359603 bytes (4.29 %).*

Table 4: Performance of rendering algorithm:

	RGB				YIQ				computed by PCA			
	# coeff	Q_V in [%]	time (Indy) in [sec.]	time (Impact) in [sec.]	# coeff	Q_V in [%]	time (Indy) in [sec].	time (Impact) in [sec.]	# coeff	Q_V in [%]	time (Indy) in ec.[sec.]	time (Impact) in [sec.]
4.6 kByte	2155	65.38	0.44	0.15	2175	81.61	0.46	0.15	2184	85.61	0.45	0.15
9.4 kByte	4480	82.73	0.9	0.31	4221	86.91	0.93	0.3	3820	88.27	0.86	0.26
79 kByte	31714	92.14	3.21	1.14	35655	94.6	3.06	1.09	30554	94.74	2.61	0.9
368 kByte	170761	96.82	11.52	4.01	178065	98.22	12.55	3.75	172240	98.47	10.77	3.58

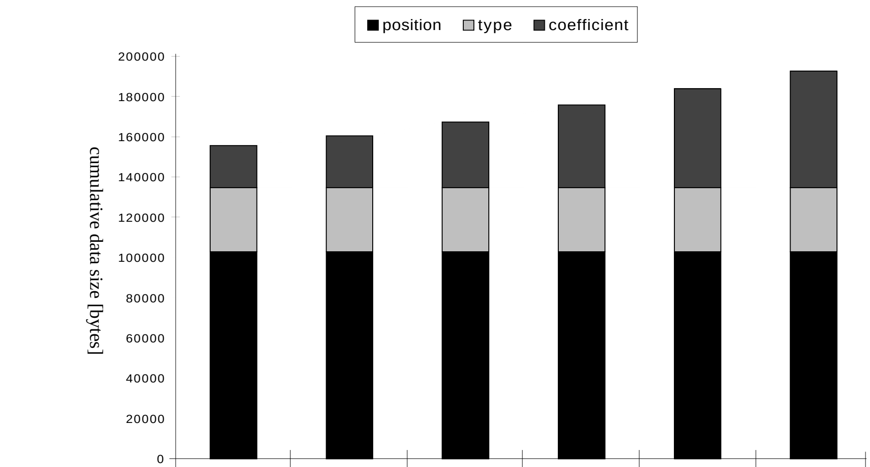
Thus a trade-off between compression gain and rendering time must be found, depending on the hardware used and available network bandwidth. To compare performance we compressed the data set using the lossless Lempel-Ziv coding-scheme, such as implemented in the gzip-procedure. The file size was reduced to 1586754 bytes (compression rate: 18.91 %). However this scheme allows only lossless compression and does not support hierachical decompression and compression domain rendering. Moreover, the data set has to be reconstructed after transmission and requires full volume memory space at the client side. Figure 16 illustrates how the data size can be adjusted by balancing the trade-off between the amount of quantization bits P and relative quantization error Q_E. The error metric we used counts for the relative energy difference of all wavelet coefficients ($w(\boldsymbol{C}) \in \{d^{type}_{mpqr}(\boldsymbol{C}), c_{Mpqr}(\boldsymbol{C})\}$) and their quantized counterparts ($\tilde{w}(\boldsymbol{C})$) defined as:

$$Q_E = \left(\frac{\sum_{\boldsymbol{C} \in \{\boldsymbol{C}_1, \boldsymbol{C}_2, \boldsymbol{C}_3\}} \sum_{w} r(w(\boldsymbol{C}))}{\sum_{\boldsymbol{C} \in \{\boldsymbol{C}_1, \boldsymbol{C}_2, \boldsymbol{C}_3\}} \sum_{w} E(w(\boldsymbol{C}))} \cdot 100 \right) [\%] \qquad (20)$$

where the residual term $r(coeff)$ is given as

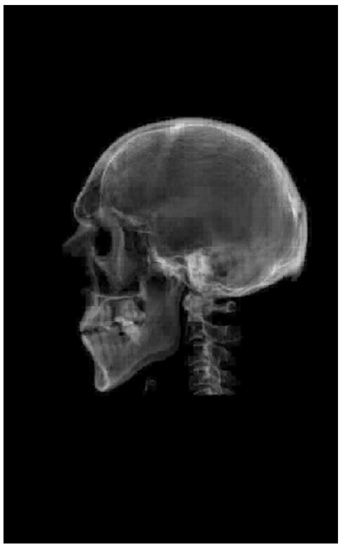
$$r(w(\boldsymbol{C})) = E(w(\boldsymbol{C}) - \tilde{w}(\boldsymbol{C})) \qquad (21)$$

Figure 16 shows a breakdown of the data size of a given volume data set displayed on the right side for an increasing number of quantization bits. The total length of each bar represents the overall data size. Each bar is divided into fractions representing the data sizes for positional data, type and coefficient-value. Positional data and type are lossless and hence not influenced by the amount of quantization bits. The figure shows that the data size increases linearly with the number of bits, whereas the quantization error decreases exponentially.



quantization bits (P)	2	3	4	5	6	7
Q_E in [%]	1.674	0.355	0.0806	0.0217	0.000598	0.000229

a)



b)

Figure 16: *Relation between data size and number of bits for quantization (data source: Visible Human Project, CT-data set). Volume size: 256^3, max. decomposition level M=3, no. of splats 74340. a) proportions of coefficient, type and positional data and resulting data size.. b) reference image of compressed data set.*

6. Conclusions and Future Work

We have presented a method for progressive compression domain volume visualization based on a unified wavelet data representation and rendering framework. As an image based approach we don‘t need to fully decompress and store the 3D data set upon rendering. Moreover, the volume can be rendered immediately from the compressed bitstream, which features progressive organization and allows to refine the image according to the provided data. The two essential features *progression* and *compression* make the framework especially suited for networked and distributed environments. In particular the limited memory requirements of our image based method enable to run it even on very low cost computers. In this context the framework offers flexibility in balancing the trade-off between network performance and local computational capabilities of client and server. In addition, since the performance is mostly driven by the underlying CPU capabilities we expect an performance upscaling with each new generation of processors. However, although we proposed some further algorithms to enhance visual quality the method is considered as a fast and low quality rendering technique, which can operate as an interactive volume data browser.

At this point in time we are working on a Java-applet [6] for further evaluations. In addition we will examine the usefulness of an arithmetic coding scheme. Since hardware accumulation boosts the performance significantly, we will provide an OpenGL interface for 3D PC graphics boards. Furthermore the transmission of selectively refined, view-dependent data over a low-bandwidth network will be investigated.

7. Acknowledgments

The authors would like to thank R. Dittrich, A. Eggenberger, A. Hubeli, J. Pittet, and P. Ruser for implementing parts of the framework. This work was supported by the ETH research council under grant No. 41-2642.5.

8. References

[1] C. Busch and M. H. Gross. “Interactive neural network texture analysis and visualization for surface reconstruction in medical imaging.” In R. J. Hubbold and R. Juan, editors, *Eurographics ’93*, pages 49–60, Oxford, UK, 1993. Eurographics, Blackwell Publishers.

[2] B. Cabral, N. Cam, and J. Foran. “Accelerated volume rendering and tomographic reconstruction using texture mapping hardware.” In A. Kaufman and W. Krueger, editors, *1994 Symposium on Volume Visualization*, pages 91–98. ACM SIGGRAPH, Oct. 1994. ISBN 0-89791-741-3.

[3] C. K. Chui and J. Z. Wang. “A cardinal spline approach to wavelets.” *Proc. Amer. Math. Soc.*, 113:785–793, 1991.

[4] R. Crawfis and N. Max. “Direct volume visualization of three-dimensional vector fields.” *1992 Workshop on Volume Visualization*, pages 55–60, 1992.

[5] I. Daubechies. *Ten Lectures on Wavelets*. CBMS-NSF regional conference series in applied mathematics, no.61, SIAM, 1992.

[6] E. E. V. V. Environment). “Homepage.” http://www.inf.ethz.ch/personal/lippert/EVOLVE.

[7] K. Fukunaga. *Introduction to Statistical Pattern Recognition. 2nd Ed.*. Academic Press, New York, 1990.

[8] M. H. Gross. “L^2 optimal oracles and compression strategies for semiorthogonal wavelets.” Technical Report 254, Computer Science Department, ETH Zürich, 1996. http://www.inf.ethz.ch/publications/tr200.html.

[9] M. H. Gross, L. Lippert, R. Dittrich, and S. Häring. “Two methods for wavelet-based volume rendering.” Technical Report 247, ETH-Zürich, 1996.

[10] M. Haley and E. Blake. “Incremental volume rendering using hierarchical compression.” *Computer Graphics Forum*, 15(3):44–55, 1996.

[11] G. Held and T. R. Marshall. *Data Compression*. John Wiley & Sons Ltd., 1991.

[12] J. T. Kajiya and B. P. Von Herzen. “Ray tracing volume densities.” In H. Christiansen, editor, *Computer Graphics (SIGGRAPH ’84 Proceedings)*, volume 18, pages 165–174, July 1984.

[13] A. Kaufman. *Volume Visualization*. IEEE Computer Society Press, 1990.

[14] P. Lacroute and M. Levoy. “Fast volume rendering using a shear–warp factorization of the viewing transformation.” In A. Glassner, editor, *Proceedings of SIGGRAPH ’94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 451–458. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.

[15] D. Laur and P. Hanrahan. “Hierarchical splatting: A progressive refinement algorithm for volume rendering.” In T. W. Sederberg, editor, *Computer Graphics (SIGGRAPH ’91 Proceedings)*, volume 25, pages 285–288, July 1991.

[16] M. Levoy. “Display of surfaces from volume data.” *IEEE Computer Graphics and Applications*, 8(3):29–37, May 1988.

[17] L. Lippert and M. H. Gross. “Fast wavelet based volume rendering by accumulation of transparent texture maps.” In *Proceedings of Eurographics ’95*, pages 431–443, 1995.

[18] National Library of Medicine. *The Visible Human Project*. http://www.nlm.nih.gov/research/visible/visible_human.html, 1995.

[19] D. H. Pritchard. “U.S. color television fundamentals – A review.” *IEEE Transactions on Consumer Electronics*, 23(4):467–478, Nov. 1977.

[20] E. Quak and N. Weyrich. “Decomposition and reconstruction algorithms for spline wavelets on a bounded inverval.” *Applied and Computational Harmonic Analysis*, 1(3):217–231, June 1994.

[21] E. J. Stollnitz, T. D. DeRose, and D. Salesin. *Wavelets for Computer Graphics*. Morgan Kaufmann Publishers, Inc., 1996.

[22] T. Totsuka and M. Levoy. “Frequency domain volume rendering.” In J. T. Kajiya, editor, *Computer Graphics (SIGGRAPH ’93 Proceedings)*, volume 27, pages 271–278, Aug. 1993.

[23] M. V. Wickerhauser. *Adapted Wavelet Analysis from Theory to Software*. A. K. Peters, Ltd., 1994.

[24] B. Yeo and B. Liu. “Volume rendering of DCT-based compressed 3D scalar data.” *IEEE Transactions on Visualization and Computer Graphics*, 1(1):29–43, Mar. 1995. ISSN 1077-2626.