# Interpolating and Downsampling RGBA Volume Data

Martin Kraus and Kai Bürger

September 18, 2008

## Abstract

For about twenty years the standard color sampling method for RGBA volume data has been the interpolation of opacity-weighted colors. In this work, we discuss the underlying approximations and derive an improved sampling method, which avoids some of these approximations. With the help of several examples and experiments we demonstrate that our new technique is in fact preferable for direct volume rendering of RGBA data. Moreover, we show that the same improvement can be applied to downsampling filters. Finally, we propose a procedure to design improved downsampling filters for the generation of pyramid RGBA volume data such as mipmap volume textures.

## 1 Introduction

Interpolating and downsampling uniform volume grids that specify color and opacity for each voxel is of particular interest for hardware-based volume rendering because many GPUs (graphics processing units) and graphics APIs (application programming interfaces) support this kind of volume data by means of RGBA (red, green, blue, and alpha) volume mipmap textures. However, a thorough understanding of interpolation and downsampling is necessary to avoid aliasing artifacts and incorrect renderings. While direct volume rendering with RGBA volume textures has received less attention in recent years, new hardware features and the development of multi-resolution volume renderers supporting multi-dimensional transfer functions is likely to lead to new interest in this subject as discussed in Section 2.

In the past, hardware support for RGBA volume textures was usually restricted to a precision of 8 bits per component, which imposed severe limitations on the achievable image quality. In fact, this limited precision of volume textures is insufficient to store opacity-weighted colors [1, 10] while it is well known that interpolation of opacity-weighted colors is preferable in direct volume rendering [2, 14]. Since modern GPUs offer support for floating-point RGBA volume textures, one might think that this issue has been resolved already.

As we will show in Section 3, however, floating-point textures also allow us to remove another important approximation that was considered inevitable for hardware-based volume rendering in the past — namely the approximation of extinction coefficients by opacities. Thus, it is preferable to perform the sampling of floating-point RGBA volume textures in a rather different way than the current state of the art suggests.

Our new sampling technique trivially affects the downsampling of RGBA volume data. However, in Section 4 we also propose a procedure based on quasi-convolution pyramid blurring [4] to design appropriate downsampling filters for specific sampling filters; in particular, we determine appropriate downsampling filters for trilinear interpolation and triquadratic B-spline filtering [12]. First, however, we will discuss related work and the use of RGBA volumes texture in general in the following section.

## 2 Background and Related Work

In the early 1990s, research in direct volume rendering was focused on volume rendering techniques that either rendered RGBA data [2, 3, 7, 8] or applied transfer functions to voxel data *before* interpolating the resulting RGBA data (i.e., pre-classification volume rendering) [14]. During the

1990s, however, the interest in post-classification volume rendering increased, i.e., in techniques that apply transfer functions *after* the interpolation of scalar data. In volume visualization, in particular, this allowed for faster changes of the transfer functions, smaller data set sizes, and sharper renderings of isosurfaces and similar structures. Post-classification benefitted from the introduction of graphics hardware features such as the OpenGL extensions for "paletted textures" and "dependent textures" as well as the publication of post-classification volume rendering algorithms [11, 13]. Today, research in direct volume rendering for volume visualization is in fact focused on post-classification techniques for scalar volume data.

Multi-resolution techniques for level-of-detail rendering of RGBA volume data had been researched in the early 1990 [2, 7, 8], and many of these approaches appeared to be applicable to multi-resolution methods for post-classification volume rendering of scalar volume data. However, researchers soon realized that the simplification of scalar volume data in general, and the downsampling of uniform, scalar volume grids in particular, requires different downsampling filters than RGBA volume data. Specifically, LaMar et al. [6] report considerably improved results with subsampling instead of linear filtering, while Kraus and Ertl [5] suggested topology-guided downsampling to preserve more of the topology of the scalar field in the downsampling process. In fact, if the topology of the scalar data field could be preserved perfectly, then the rendering of the downsampled data will include the same colors as a rendering of the original data although the distribution of colors in space may strongly differ. Moreover, a perfect preservation of the topology is not possible and its benefits are only effective for one-dimensional transfer functions while multi-dimensional transfer functions can result in arbitrarily strong color differences.

Another approach to the downsampling of scalar volume data was proposed by Younesy et al. [15], who suggested to compute data histograms in a neighborhood of each downsampled voxel and approximate these data distributions by Gaussians for a more efficient color computation. Unfortunately, their results are apparently computed by a pre-classification volume renderer with downsampled scalar data: a rather uncommon combination that is particularly prone to aliasing effects.

In summary, downsampling of scalar volume data for post-classification volume rendering is considerably more difficult and less well understood than downsampling of RGBA volume data. In fact, it is unclear how to achieve the same image quality; in particular, for multi-dimensional transfer functions. Thus, the best approach to compute multi-resolution hierarchies of uniform volume grids (e.g., mipmap volume textures) for direct volume rendering with high image quality appears to be based on pre-classification and downsampling of RGBA volume textures.

Moreover, some features of modern GPUs, namely floating-point textures and the render-to-texture functionality, provide ample hardware support for generating and processing RGBA volume textures. Thus, real-time pre-classification of volume data and volume mipmap generation with complex downsampling filters is possible on GPUs.

Apart from downsampling RGBA volume data, this work is also concerned with interpolation of RGBA volume data. Unfortunately, it is not quite clear, who introduced opacity-weighted color interpolation in volume rendering—Drebin et al. [3] are among the first who have published this concept. Our improvements in comparison to opacity-weighted color interpolation are also based on previous work by Max [9] and Wittenbrink et al. [14] and are discussed in the next section.

# 3 Interpolating RGBA Volume Data

## 3.1 Optical Model

We employ the optical model and notation described by Max [9], which consists of an extinction coefficient $\tau(\mathbf{x})$ and an emitted color $g(\mathbf{x}) \stackrel{\text{def}}{=} C(\mathbf{x})\tau(\mathbf{x})$ for any point $\mathbf{x}$ in space. For a ray $\mathbf{x}(s) \stackrel{\text{def}}{=} \mathbf{x}_0 + s\hat{\mathbf{d}}$ starting with color $I(0)$ at $\mathbf{x}(0)$ in direction of the unit vector $\hat{\mathbf{d}}$, the integrated color $I(s)$ received at $\mathbf{x}(s)$ is:

$$I(s) = I(0)\exp\left(-\int_0^s \tau(\mathbf{x}(s'))\mathrm{d}s'\right) + \int_0^s C(\mathbf{x}(s'))\tau(\mathbf{x}(s')) \qquad (1)$$

$$\times \exp\left(-\int_{s'}^{s} \tau(\mathbf{x}(s''))\mathrm{d}s''\right)\mathrm{d}s'.$$

In order to discretize this integral, a short sampling distance $d$ is chosen, such that $C(\mathbf{x})$ and $\tau(\mathbf{x})$ can be considered constant. In this case, the integral can be solved exactly (see Equation 8 in [9]):

$$\begin{aligned} I(s) &= I(s-d)\exp\left(-\tau(\mathbf{x}(s))d\right) \\ &\quad + C(\mathbf{x}(s))\left(1-\exp\left(-\tau(\mathbf{x}(s))d\right)\right). \end{aligned} \tag{2}$$

Employing the opacity $\alpha(\mathbf{x}(s))$ for distance $d$ with

$$\alpha(\mathbf{x}(s)) \stackrel{\text{def}}{=} 1-\exp\left(-\tau(\mathbf{x}(s))d\right), \tag{3}$$

this equation reduces to the familiar compositing equation for back-to-front blending:

$$\begin{aligned} I(s) &= I(s-d)(1-\alpha(\mathbf{x}(s))) \\ &\quad + C(\mathbf{x}(s))\alpha(\mathbf{x}(s)). \end{aligned} \tag{4}$$

Details of the implementation of back-to-front and front-to-back blending are provided, for example, by Wittenbrink et al. [14].

We note that many volume rendering tools don't allow to specify extinction coefficients $\tau \in [0,\infty]$ but opacities $\alpha \in [0,1]$, in particular by means of transfer functions. In these cases, a distance $d$ is chosen (sometimes implicitly), which allows to bijectively map extinction coefficients to opacities according to Equation 3. The inverse relation is:

$$\tau = \frac{-\ln(1-\alpha)}{d}. \tag{5}$$

Furthermore, we consider an extension of the described optical model to $n$ materials (see, for example, the work by Drebin et al. [3]) with constant material colors $C_i$ and extinction coefficients $\tau_i$ for $i = 1,\ldots,n$. Each point defines a set of percentages $p_i(\mathbf{x})$, $i = 1,\ldots,n$, which specify the amount of material $i$ at point $\mathbf{x}$. For this multi-material model, the emitted color at $\mathbf{x}$ is:

$$g(\mathbf{x}) = \sum_{i=1}^{n} p_i(\mathbf{x})C_i\tau_i \tag{6}$$

and the total extinction coefficient is

$$\tau(\mathbf{x}) = \sum_{i=1}^{n} p_i(\mathbf{x})\tau_i. \tag{7}$$

## 3.2 Opacity-Based Sampling

If the color $C(\mathbf{x})$ and the opacity $\alpha(\mathbf{x})$ at a point $\mathbf{x}$ are obtained by interpolating colors and opacities specified at discrete voxel positions, then the product $C(\mathbf{x})\alpha(\mathbf{x})$ is interpolated with the squared interpolation functions. This can result in a form of color bleeding as discussed in detail by Wittenbrink et al. [14]. Therefore, opacity-weighted colors $\tilde{C} \stackrel{\text{def}}{=} C\alpha$ and opacities $\alpha$ for some distance (which does not need to be the sampling distance) are stored or computed per voxel and then interpolated for each sampling point as suggested, for example, by Drebin et al. [3].

The preference for opacity-weighted colors can be justified in a multi-material optical model. We employ the approximations

$$\begin{aligned} \alpha(\mathbf{x}) &= 1-\exp\left(-\tau(\mathbf{x})d\right) && (8)\\ &= 1-\exp\left(-\sum_{i=1}^{n} p_i(\mathbf{x})\tau_i d\right) && (9)\\ &\approx \sum_{i=1}^{n} p_i(\mathbf{x})\left(1-\exp(-\tau_i d)\right) && (10)\\ &= \sum_{i=1}^{n} p_i(\mathbf{x})\alpha_i && (11) \end{aligned}$$

and

$$\begin{aligned} \tilde{C}(\mathbf{x}) &\stackrel{\text{def}}{=} g(\mathbf{x})d && (12)\\ &= \sum_{i=1}^{n} p_i(\mathbf{x})C_i\tau_i d && (13)\\ &\approx \sum_{i=1}^{n} p_i(\mathbf{x})C_i\alpha_i && (14)\\ &= \sum_{i=1}^{n} p_i(\mathbf{x})\tilde{C}_i && (15) \end{aligned}$$

with $\tilde{C}_i \stackrel{\text{def}}{=} C_i\alpha_i$. These approximations are valid for small $\tau_i d$, $i = 1,\ldots,n$. Thus, $\alpha(\mathbf{x})$ and $\tilde{C}(\mathbf{x})$ are linear combinations of the $p_i(\mathbf{x})$ with material constants $\tilde{C}_i$ and $\alpha_i$. Therefore, instead of storing a potentially large set of percentages $p_i$, $i = 1,\ldots,n$ for each voxel and interpolating between these percentages to compute $\alpha(\mathbf{x})$ and $\tilde{C}(\mathbf{x})$, we can also just store one opacity $\alpha$ and one opacity-weighted color $\tilde{C}$ for each voxel and interpolate between this voxel data directly, which is exactly what the interpolation of opacity-weighted colors suggests even

without explicitly specifying materials and material percentages $p_i$.

After interpolating opacities and opacity-weighted colors, a so-called *opacity correction* is necessary if opacities are stored for a different distance $d'$ than the actual sampling distance $d$, for example, because the sampling distance $d$ is chosen adaptively.

We will refer to the resulting procedure as *opacity-based sampling*, which consists of the following steps:

1. interpolation of the opacity-weighted color $\tilde{C}'$ and the opacity $\alpha'$ for distance $d'$ at the sampling position,
2. opacity correction for the actual sampling distance $d$ instead of $d'$:
$$\alpha = 1 - (1 - \alpha')^{d/d'} \tag{16}$$
3. opacity correction of the opacity-weighted color:
$$\tilde{C} = \tilde{C}' \frac{\alpha}{\alpha'}. \tag{17}$$

The resulting opacity $\alpha$ and opacity-weighted color $\tilde{C}$ is then composited as required by Equation 4.

### 3.3 Extinction-Based Sampling

While the interpolation of opacity-weighted colors is generally accepted in volume rendering, it is only an approximation to the preferable interpolation of extinction coefficients $\tau$ and extinction-weighted colors $\tilde{C}^{(\tau)} \stackrel{\text{def}}{=} C\tau$.

Analogously to opacity-weighted colors, the weighting with extinction coefficients can be explained in the multi-material model. We have (without any approximations):

$$\tau(\mathbf{x}) = \sum_{i=1}^{n} p_i(\mathbf{x})\tau_i \tag{18}$$

and

$$\begin{aligned} \tilde{C}^{(\tau)}(\mathbf{x}) &\stackrel{\text{def}}{=} g(\mathbf{x}) && (19)\\ &= \sum_{i=1}^{n} p_i(\mathbf{x}) C_i \tau_i && (20)\\ &= \sum_{i=1}^{n} p_i(\mathbf{x}) \tilde{C}_i^{(\tau)}. && (21) \end{aligned}$$

Thus, $\tau(\mathbf{x})$ and $\tilde{C}^{(\tau)}(\mathbf{x})$ are linear combinations of the percentages $p_i(\mathbf{x})$ with constants $\tilde{C}^{(\tau)}$ and $\tau_i$, $i = 1, \ldots, n$. Therefore, we can interpolate voxel attributes $\tau$ and $\tilde{C}^{(\tau)}$ instead of percentages $p_i$, $i = 1, \ldots, n$. In contrast to oppacity-based sampling, no approximations are necessary to derive this result.

We will refer to the interpolation of extinction coefficients and extinction-weighted colors as *extinction-based sampling*, which consists of these steps:

1. interpolation of the extinction-weighted color $\tilde{C}^{(\tau)}$ and the extinction coefficient $\tau$ at the sampling position,
2. computation of $\alpha$ for the sampling distance $d$:
$$\alpha = 1 - \exp(-\tau d), \tag{22}$$
3. computation of the opacity-weighted color $\tilde{C}$:
$$\tilde{C} = \tilde{C}^{(\tau)} \frac{\alpha}{\tau}. \tag{23}$$

Again, the resulting opacity $\alpha$ and opacity-weighted color $\tilde{C}$ are composited according to Equation 4.

### 3.4 Comparison

In general, opacity-based and extinction-based sampling of voxel data compute different results because $\alpha(\mathbf{x})$ is a non-linear function of $\tau(\mathbf{x})d'$, and therefore the interpolation of opacities and opacity-weighted colors is an approximation, which is only appropriate for small products $\tau(\mathbf{x})d'$ as can be shown by a Taylor expansion of $\alpha(\tau(\mathbf{x})d')$ for $\tau(\mathbf{x})d' \approx 0$:

$$\alpha(\tau(\mathbf{x})d') = \tau(\mathbf{x})d' + O\left((\tau(\mathbf{x})d')^2\right). \tag{24}$$

It is important to note that opacity-based sampling not only suffers from discretization errors because of a finite sampling distance $d$ but also from the mapping of voxel data $\tau(\mathbf{x})$ to $\alpha(\mathbf{x})$ for a finite distance $d'$. The correct result is obtained only in the limit of $d \to 0$ *and* $d' \to 0$. For $d' \to 0$, however, voxel opacities and opacity-weighted colors will exponentially approach 0, which can result in severe numerical problems. Extinction-based sampling, on the other hand, allows us to perform the

same computations without the need to deal with infinitely small quantities.

One advantage of opacity-weighted colors is their limited data range $[0, 1]$. However, a fixed-point precision of 8 bits is usually insufficient for opacity-weighted colors [1, 10]; thus, at least 16 bit numbers have to be used for many applications. However, 16 bit floating-point numbers require no additional memory; therefore, the limited data range required by opacity-weighted colors appears to be less important in actual implementations.

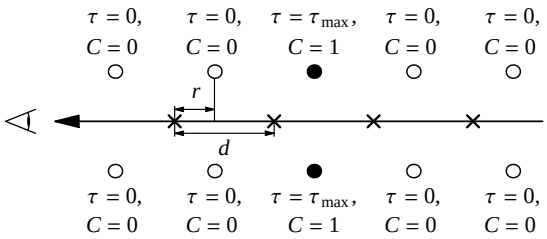


Figure 1: Exemplary ray integration in a two-dimensional grid (see main text for details).

## 3.5 Example

While many researchers are aware of the approximation implied by opacity-based sampling, it is usually not considered problematic. In fact, Wittenbrink et al. [14] present an example, where opacity-weighted color interpolation results in an integrated color, which is independent of the sampling positions. In particular, the sampling positions may be identical to the voxel positions; thus, the employed approximation does not affect the result in Wittenbrink et al.'s example.

There are, however, also examples that clearly demonstrate the benefits of interpolating extinction-weighted colors instead of opacity-weighted colors. One of these examples is illustrated in Figure 1: A ray traverses a uniform grid towards the camera on the left-hand side. Sampling positions are depicted by crosses and parameterized by the uniform sampling distance $d$ and the projected distance $r$ relative to a vertex of the grid. All distances are measured in units of the distance between the grid's vertices; in other words: the distance between two neighboring vertices of the grid is 1. Sampling is performed by bilinear interpolation of the vertex attributes; i.e., either opacities and opacity-weighted colors for opacity-based sampling, or extinction coefficients and extinction-weighted colors for extinction-based sampling. The filled dots represent vertices with color $C = 1$, extinction coefficient $\tau = \tau_{\max}$, the corresponding extinction-weighted color $\tilde{C}^{(\tau)} = \tau_{\max}$, opacity for the distance $d' = d$: $\alpha = 1 - \exp(-\tau_{\max} d')$, and opacity-weighted color $\tilde{C} = \alpha$. The circles represent vertices with color $C = 0$, extinction coefficient $\tau = 0$, the corresponding extinction-weighted color $\tilde{C}^{(\tau)} = 0$, opacity $\alpha = 0$, and opacity-weighted color $\tilde{C} = 0$.

Figure 4a shows four plots of the resulting integrated opacity $\alpha_{\text{ray}}$ of the ray for opacity-based sampling for values $\tau_{\max} = 1$, 2, 4, and 8 in dependency of $d = d'$ and $r$ while Figure 4b shows the results for extinction-based sampling. Note that all deviations from a constant height are deviations due to sampling errors. Obviously, extinction-based sampling shows significantly less discretization errors, in particular for $0.5 < d < 1$, which is a typical range of sampling distances for real-time and interactive volume rendering.

The choice of the distance $d'$ to compute the voxels' opacities is crucial: a smaller value of $d'$ improves the results for the same value of $\tau_{\max}$. However, the same problems will then appear for correspondingly larger values of $\tau_{\max}$. Moreover, smaller values of $d'$ will result in exponentially smaller voxel opacities, which are likely to result in numerical problems. In our example, we set $d' = d$, which is a typical choice for uniform sampling because it avoids the need to perform any opacity correction. Choosing a larger distance $d' > d$ results in considerably stronger deviations and will *not converge* to the correct result for $d' = \text{const}$ and $d \to 0$ since the correct, integrated opacity will be underestimated due to the non-linear mapping between extinction coefficients and voxel opacities. On the other hand, the whole issue of choosing an appropriate value of $d'$ is resolved by extinction-based sampling, which does not employ any distance except the sampling distance $d$.

## 3.6 Experiments

To conclude this section, we show several direct volume renderings of synthetic data sets. Figures 5 and 6 depict an axis-aligned, one-voxel-wide wall of

opaque voxels in an otherwise transparent volume. Opacity-based sampling in Figure 5 shows the first sampling artifacts for $d = d' \approx 0.6$ while the first rendering artifacts of extinction-based sampling in Figure 6 appear for $d = d' \approx 1.6$.

In Figure 7, a set of scattered, opaque voxels is rendered. The rendering with opacity-based sampling in Figure 7a shows stronger diamond-shaped interpolation artifacts and a greater variation of the intensity of the projected voxels in comparison to extinction-based sampling in Figure 7b. The variation of the projected intensity depending on the viewing parameters is particularly noticeable in animations.

# 4 Downsampling RGBA Volume Data

After discussing interpolation in RGBA volume grids, this section is concerned with downsampling RGBA volume grids. While interpolation is crucial for the maxification of texture data, downsampling is just as important for the minification case. Thus, both topics have to be addressed for mipmap texturing with RGBA volume data.

## 4.1 Opacity-Based vs. Extinction-Based Downsampling

Analogously to the case of interpolation, filtering of RGBA volume data should be performed on extinction coefficients and extinction-weighted colors instead of opacity and opacity-weighted colors. The difference is illustrated by Figures 2 and 3. In Figure 2, $2 \times 2 \times 2 = 8$ voxels are downsampled to one coarser voxel by averaging voxel data. Four of the finer voxels are almost opaque with the extinction coefficient $\tau = 4$ and four are transparent with $\tau = 0$. The average extinction coefficient is $\tau' = 2$ and this is the downsampled value for the coarser voxel. (For $d' = 2$ this corresponds to an opacity of $\exp(-2\times 2) = 0.98$.) The result is plausible because the combined transparency through an opaque and a transparent voxel equals the transparency of the coarser voxel:

$$\exp(-\tau d) \times \exp(-0d) = \exp(-\tau' d'); \qquad (25)$$

thus, the opacity of the coarser voxel is the same as the opacity of an axis-aligned orthogonal projection

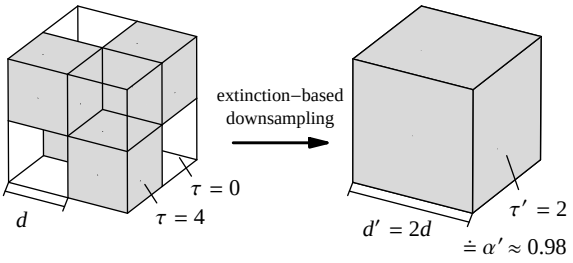


Figure 2: Downsampling by averaging the extinction coefficients of 8 voxels.

of the 8 finer voxels, and it is also the same as the opacity of one of the finer voxels. In general, any opaque structures of the data set are "blurred" in the downsampling process, but their total opacity is not reduced; for example, a thin surface will become thicker by the downsampling, but rays intersecting the surface will experience approximately the same opacity in spite of the downsampling.

The situation is different if opacities are downsampled as illustrated in Figure 3. $\tau = 4$ corresponds to $\alpha \approx 0.98$ for $d = 1$, while $\tau = 0$ corresponds to $\alpha = 0$. Thus, the average opacity of the 8 voxels is 0.49. If the larger size of the coarser voxel is taken into account by means of an opacity correction (see Section 3), the resulting opacity of the coarser voxel is $\alpha' \approx 0.74$, which is significantly less than the corresponding opacity computed by downsampling extinction coefficients. In fact, downsampling opacities will in general result in a loss of opacity in the downsampling process.

Thus, RGBA volume data should not be downsampled by filtering opacities and opacity-weighted colors but extinction coefficients and extinction-weighted colors although the former approach is usually employed [2, 7, 8].

## 4.2 Filter Design for Downsampling

While there are many approaches to the design of pyramid filters, there are no clear recommendations for the design of appropriate downsampling filters for mipmap RGBA texture generation. Thus, the $2 \times 2 \times 2$ box filter is often employed although it is known to generate results of inferior quality under many circumstances.

Here we propose to employ the procedure described by Kraus [4] to design appropriate down-

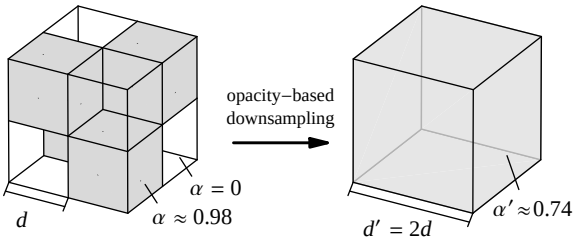


Figure 3: Downsampling by averaging the opacities of 8 voxels and performing opacity correction for the coarser voxel.

sampling filters depending on the employed volume sampling; for example, hardware-supported trilinear interpolation or B-spline filtering as suggested by Sigg and Hadwiger [12]. Apart from being symmetric, an appropriate downsampling filter should have a small support to avoid excessive blurring. Furthermore, the result of sampling the downsampled data should be as close as possible to blurring with a convolution filter since all deviations from this kind of blur tend to reveal the coarse resolution of the downsampled grid.

This task, however, is very close to the problem addressed by Kraus [4], namely the design of an appropriate downsampling filter for a given upsampling filter under the constraint that the resulting pyramid algorithm is as close as possible to blurring by a convolution filter. In particular, he discusses the case of an upsampling filter that is equivalent to quadratic B-spline filtering. Restricting the set of candidates for the downsampling filter to symmetric, 4-tap filters and analyzing the deviations of the resulting pyramid blur from a convolution filter by means of two root-mean-square errors ($\varepsilon$ for the whole filter and $\varepsilon_0$ for the center), he determines an approximation to the best filter in one dimension: $\frac{1}{64}(\,13\,19\,19\,13\,)$ and corresponding tensor products in higher dimensions.

We propose to employ the three-dimensional tensor version of this downsampling filter for mipmap generation if the volume data is sampled by triquadratic B-spline filtering [12]. To determine an appropriate downsampling filter for the more common case of trilinear volume interpolation, we replaced the pyramid upsampling of Kraus' procedure by a triangle filter and determined the symmetric, 4-tap downsampling filter that minimizes $\varepsilon$ and $\varepsilon_0$. While the two errors achieve their minima for different filters, the best compromise appears to be the 4-tap box filter with $\varepsilon = 0.057$ and $\varepsilon_0 = 0.0093$. For comparison, the 2-tap box filter results in considerably stronger deviations from a convolution filter, which are characterized by $\varepsilon = 0.34$ and $\varepsilon_0 = 0.14$. On the other hand, the best downsampling filter for quadratic B-spline sampling achieves $\varepsilon = 0.0276$ and $\varepsilon_0 = 0.0027$.

It should be noted that this procedure is not well suited to design downsampling filters for scalar data as discussed in Section 2.

# 5 Conclusions

This work demonstrates that extinction coefficients and extinction-weighted colors are more appropriate for interpolation and downsampling of RGBA volume data than opacity and opacity-weighted colors. Since modern GPUs support floating-point RGBA volume textures, the former should be preferred even for hardware-based implementations. Moreover, we propose a procedure to design appropriate downsampling filters for RGBA volume grids in dependency of the employed volume sampling and present results for trilinear interpolation and triquadratic B-spline filtering

We are confident that our contributions help to improve the image quality achieved by direct volume rendering of RGBA data. Rendering of RGBA volume data itself is likely to gain more interest in the near future due to several trends in volume rendering, e.g., multi-resolution volume data and multi-dimensional transfer functions, as well as new GPU features such as floating-point textures and texture render targets.

# References

[1] I. Bitter, N. Neophytou, K. Mueller, and A. E. Kaufman. Squeeze: Numerical-precision-optimized volume rendering. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware*, pages 25–34, 2004.

[2] J. Danskin and P. Hanrahan. Fast algorithms for volume ray tracing. In *Proceedings of*

*the 1992 Workshop on Volume Visualization*, pages 91–98, 1992.

[3] R. A. Drebin, L. Carpenter, and P. Hanrahan. Volume rendering. In *Proceedings of SIGGRAPH '88*, pages 65–74, 1988.

[4] M. Kraus. Quasi-convolution pyramidal blurring. In *Proceedings GRAPP 2008*, pages 155–162, 2008.

[5] M. Kraus and T. Ertl. Topology-guided downsampling. In K. Mueller and A. Kaufmann, editors, *Volume Graphics 2001*, Springer Computer Science, pages 223–234. Springer Verlag, Wien, New York, 2001.

[6] E. LaMar, B. Hamann, and K. I. Joy. Multiresolution techniques for interactive texture-based volume visualization. In *Proceedings of IEEE Visualization '99*, pages 355–361, 1999.

[7] D. Laur and P. Hanrahan. Hierarchical splatting: A progressive refinement algorithm for volume rendering. In *Proceedings of SIGGRAPH '91*, pages 285–288, 1991.

[8] M. Levoy and R. Whitaker. Gaze-directed volume rendering. *ACM Computer Graphics (Proceedings 1990 Symposium on Interactive 3D Graphics)*, 24(2):217–223, 1990.

[9] N. Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, 1995.

[10] M. Meißner, J. Huang, D. Bartz, K. Mueller, and R. Crawfis. A practical evaluation of popular volume rendering algorithms. In *Proceedings of the 2000 Symposium on Volume Visualization*, pages 81–90, 2000.

[11] K. Mueller, T. Möller, and R. Crawfis. Splatting without the blur. In *Proceedings of IEEE Visualization '99*, pages 363–370, 1999.

[12] C. Sigg and M. Hadwiger. Fast third-order texture filtering. In M. Pharr, editor, *GPU Gems 2*, pages 313–329. Addison Wesley, 2005.

[13] C. M. Stein, B. G. Becker, and N. L. Max. Sorting and hardware assisted rendering for volume visualization. In A. Kaufman and W. Krueger, editors, *Proceedings of the 1994 Symposium on Volume Visualization*, pages 83–89. ACM Press, 1994.

[14] C. M. Wittenbrink, T. Malzbender, and M. E. Goss. Opacity-weighted color interpolation for volume sampling. In *Proceedings of the 1998 IEEE Symposium on Volume Visualization*, pages 135–142, 1998.

[15] H. Younesy, T. Möller, and H. Carr. Improving the quality of multi-resolution volume rendering. In *Proceedings Eurographics/IEEE-VGTC Symposium on Visualization*, pages 251–258, 2006.

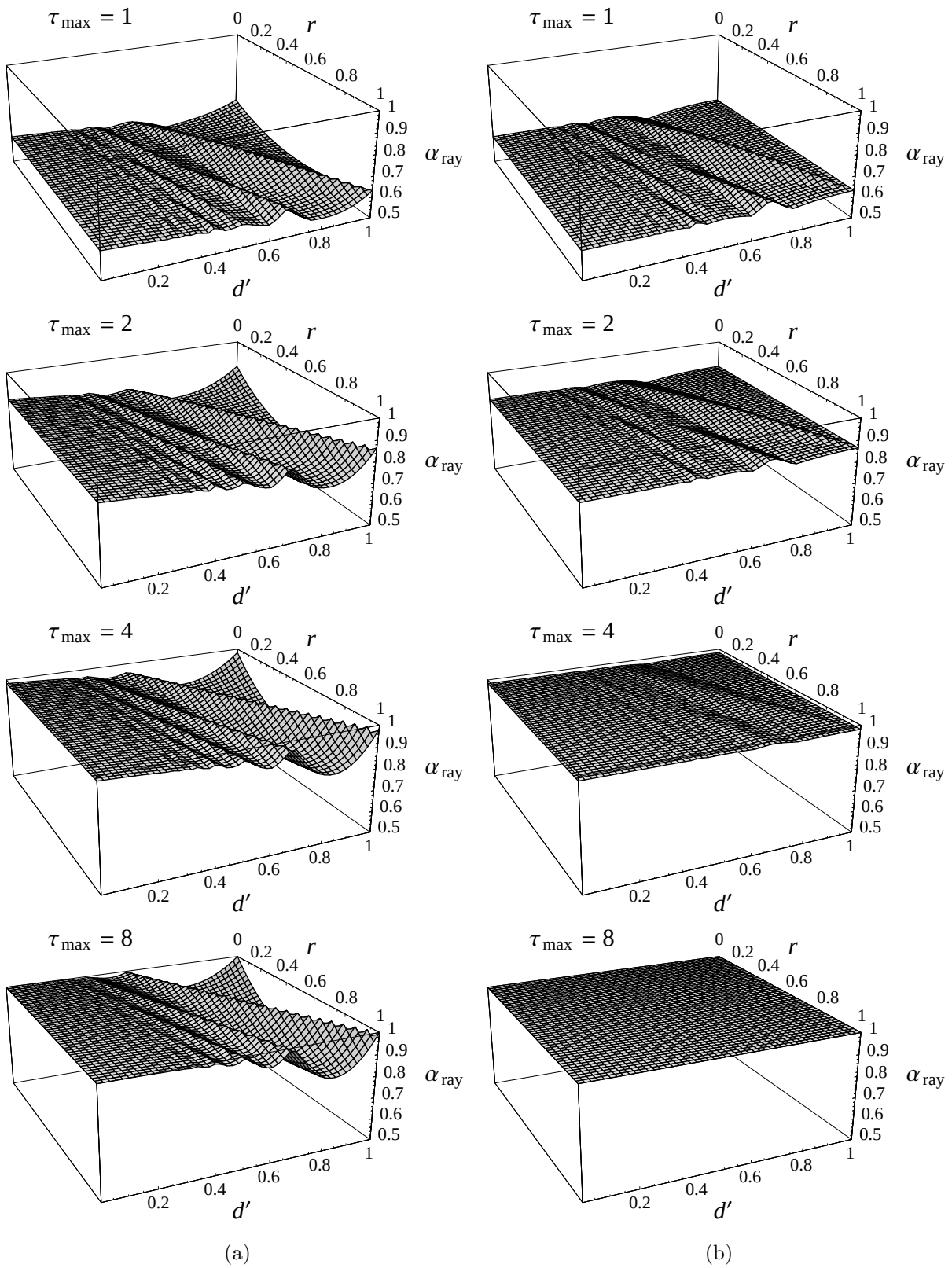


Figure 4: Ray integration results for the total opacity $\alpha_{\text{ray}}$ in dependency of the parameters $r$ and $d = d'$, which are illustrated in Figure 1, (a) for opacity-based sampling and (b) for extinction-based sampling.

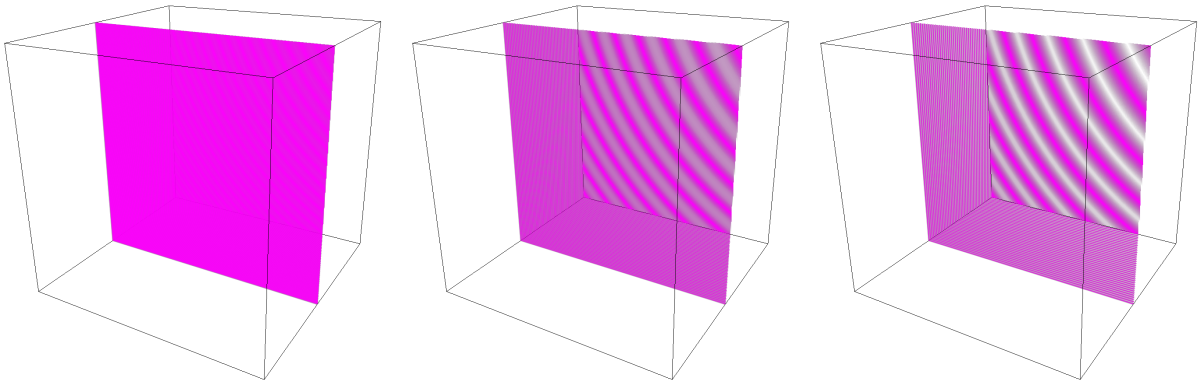


Figure 5: Opacity-based sampling of a one-voxel-thick wall of opaque voxels.

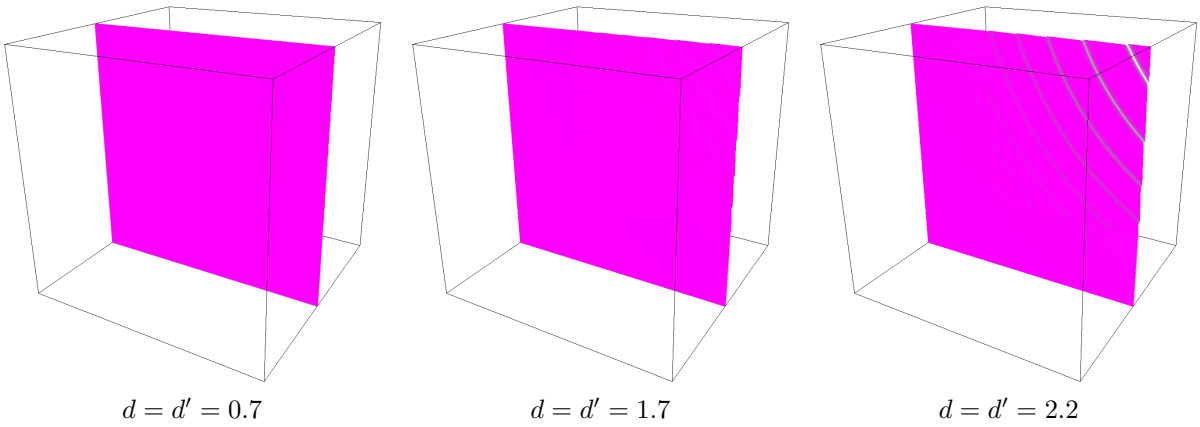


Figure 6: Extinction-based sampling of the same data set as in Figure 5.


Figure 7: Direct volume rendering with (a) opacity-based sampling and (b) extinction-based sampling.