Pre-Integrated Volume Rendering with Non-Linear Gradient Interpolation

Amel Guetat, Alexandre Ancel, Stephane Marchesin, and Jean-Michel Dischler

Abstract—Shading is an important feature for the comprehension of volume datasets, but is difficult to implement accurately. Current techniques based on pre-integrated direct volume rendering approximate the volume rendering integral by ignoring non-linear gradient variations between front and back samples, which might result in cumulated shading errors when gradient variations are important and / or when the illumination function features high frequencies. In this paper, we explore a simple approach for pre-integrated volume rendering with non-linear gradient interpolation between front and back samples. We consider that the gradient smoothly varies along a quadratic curve instead of a segment in-between consecutive samples. This not only allows us to compute more accurate shaded pre-integrated look-up tables, but also allows us to more efficiently process shading amplifying effects, based on gradient filtering. An interesting property is that the pre-integration tables we use remain two-dimensional as for usual pre-integrated classification. We conduct experiments using a full hardware approach with the Blinn-Phong illumination model as well as with a non-photorealistic illumination model.

Index Terms—Ray casting, pre-integration, Phong shading, volume rendering.

1 Introduction

Volume rendering is a well-known method for exploring 3D scalar fields by using a simplified light transport theory. It expresses the color of a pixel as an integral along the viewing ray traversing the volume. However rendering should not only be fast to ensure interactivity, it must also be of good visual quality to avoid artifacts while making internal features well perceptible. In the worst case, artifacts might lead to wrong interpretation of the data. High regular or adaptive sampling rates lead to accurate results but drive accordingly very high computational costs, often limiting framerates, especially for large datasets. By exporting some computations to a pre-processing step, pre-integration is a widely used technique for improving numerical accuracy without increasing the computational requirements (see related works). Since shading is generally considered as an important issue, Lum *et al.* [8] suggested to further improve pre-integration by linearly interpolating intensity values between the front and back sample points, thus guaranteeing continuity of shading from one slab to the next. This approach can be considered as a volumetric extension of Gouraud shading on surfaces. Gouraud shading, however, is known to overblur surface aspects, while discontinuities in the derivative are visible. In the case of volume rendering, such effects are less perceptible on iso-surfaces, because gradients are computed densely on a per-voxel basis (when gradients are pre-computed and stored as 3D texture, the GPU also applies a tri-linear filtering, thus interpolating the gradient inside the voxel). In addition, contributions are mixed and accumulated along the ray depending on the opacity, which potentially smoothes results further. Yet, errors in the form of a loss of luminous energy are introduced in two cases: high frequency gradient variations and high frequency shading functions (for example the specular part of lighting models). Unfortunately, it is not possible to perform accurate pre-integration by considering gradients on front and back samples because of practical issues: a table taking the front and back normals into account would require four additional dimensions (two dimensions per normal when representing the vector as Euler angles).

- *A. Guetat, University of Strasbourg.*
- *A. Ancel, University of Strasbourg.*
- *S. Marchesin, UC Davis.*
- *J.M. Dischler, University of Strasbourg.*

Manuscript received 31 March 2010; accepted 1 August 2010; posted online 24 October 2010; mailed on 16 October 2010.
For information on obtaining reprints of this article, please send email to: tvcg@computer.org.

Our technique is based on a non-linear shading interpolation between front and back samples, and thus significantly reduces shading-related errors. It uses a quadratic curve to interpolate the gradients between the front and back samples. Such an interpolation leads to a cubic polynomial under the integral, which allows accurate computation of two-dimensional pre-integration tables. Our experimental study demonstrates that an interpolation of the gradients along a curve (close to a circle) between samples results in more accurate rendering, especially in the presence of important local gradient variations and / or of high frequency lighting functions. In the case of the Blinn-Phong lighting model, we process specular highlights by replacing the usual cosine lobes by cosine cones. We show that the latter do well preserve visual cues, but are computationally much simpler. Because of improved overall accuracy, our technique can be used to process visual effects related to gradient filtering more efficiently, especially techniques that amplify local gradient variations, for instance to improve small scale details perception. Our approach is furthermore compatible with many other local illumination models, including non-photorealist techniques, as far as these are related to gradients (normal vectors on surfaces).

The remaining parts of this paper are structured as follows. We first present related works concerning pre-integrated classification, shading models and gradient filtering for detail enhancement. Section 3 describes some background and motivations. Then, in Section 4, we describe our approach for non-linear interpolation of gradient vectors between two consecutive samples along the ray. In particular, we propose to analyze errors introduced when gradients are just linearly interpolated between front and back samples. Next, we show how our interpolation can be used along with the Blinn-Phong illumination model as well as pre-integrated classification. We also discuss extensions to other local illumination models. Finally, before concluding, results are given in Section 6.

2 Related Works

2.1 Pre-integrated classification

Different strategies for numerically solving the volume rendering integral were discussed by Max et al. [10]. Their idea was to precompute and tabulate the indefinite rendering integral between values defined at the vertices of the polyhedra for the cell projection method, assuming that the density scalar function varies linearly along the ray. Improvements were further discussed by Williams *et al.* [20]. They calculate the volume rendering integral cell by cell assuming that the transfer function varies piecewise linearly along a ray segment within each cell. Later, simplifications were derived by Stein *et al.* [21]. They de-

velop a faster but less accurate method, where they assume the opacity varies linearly along the ray segment and the color is constantly equal to the average of the color at the front and the back of the ray segment. This was an approximation, since the opacity along a ray segment hides the far color more than the near one, but it was much quicker to evaluate. Roettger *et al.* [15] applied this technique as an enhancement to the Projected Tetrahedra algorithm using 3D texture hardware for enabling the use of arbitrary transfer functions and for rendering isosurfaces without reconstructing them geometrically. Compared to Stein, Roettger *et al.* develop a scheme to approximate the volume rendering integral for different sampling distances which allows one to pre-integrate the contribution of the volume rendering integral within the considered interval by a 2D look-up table instead of a 3D look-up table. Engel *et al.* [2] name this technique pre-integration and extend it by computing the colors in the same way as Roettger *et al.*, but considering their modulation by diffuse and specular shading. For the normals, they average the normals of the two subsequent sampling points under consideration. However, to accelerate the computation of the pre-integration tables, they hold the distance between sample points constant and neglect the attenuation within the segment intervals. Later improvements [13] consider an optimized look-up table-generation and final rendering using 2D texture hardware. Roettger *et al.* [14] improve the algorithm by introducing super-sampling and accurate clipping in order to achieve a better image quality around high frequency transfer function and data variations. This can also significantly improve shading by doing super-sampling where normal variations are important. An application of pre-integration to Shear Warp algorithms was suggested by Schulze *et al.* [18]. They render slabs between adjacent slices instead of individual slices using a buffer slice to store interpolated scalar values of the back slice, and store the result into a 2D lookup table considering a constant distance. They improve image quality at the expense of the algorithm performance. Lum *et al.* [8] improved pre-integration for shading by linearly interpolating the lighting in the front and back sample points. Further suggestions were given for computing the pre-integration look-up tables. [11] restricts transfer functions to piecewise linear functions which allows one to precompute a pre-integration table independent of classification. Recently, El Hajjar *et al.* [5] derived a method based on a second order polynomial for pre-integration at constant sampling steps. But they need to use a 3D table even though considering equidistant samples. Providing a better approximation of shading has not been considered by this approach.

With regard to shading combined with pre-integration, two main strategies can be differentiated: approaches based on super-sampling to compute more samples on high frequency variations and approaches based on illumination interpolation. The former can lead to excessive computations and needs to consider variable sampling distances, e.g. 3D tables, and the latter may introduce numerical errors on regions where the gradient variations are locally important. Our motivation is to provide a method that avoids both drawbacks by considering a true gradient interpolation for computing the tables.

2.2 Local illumination models

Local illumination simulates the interaction of light with surfaces. The empirical Phong [12] illumination model is the most frequently used in volume rendering. It computes the light reflected by a surface as a combination of ambient, diffuse and specular terms where each term is represented as an RGB triplet. The specular term depends on the viewing vector $\mathbf{V}$ and on $\mathbf{R}$ the direction of a perfect reflection. The Blinn-Phong model proposes to compute more efficiently the specular term by introducing a vector $\mathbf{H}$, which is the halfway between $\mathbf{V}$ and $\mathbf{L}$, where the latter represents the light source direction. More recently other local illumination models have been proposed, for instance in the field of non-photorealistic rendering, to provide new graphical effects. Gooch *et al.* 's model [3] for instance adds a subtone linearly ranging from a cool (blue) to warm (yellow) color. Such illumination models can be used with direct volume rendering as done in [1] for instance.

2.3 Gradient computation and filtering

For computing gradient approximations a popular technique consists in applying the Sobel operator, e.g. a discrete differentiation of neighboring voxels, but higher order interpolation polynomials can be used instead. However, these are often time consuming and can hardly be computed on the fly in real-time. Gradients are therefore generally pre-computed. Fast hardware compatible higher order interpolation techniques have been explored in [4]. Other techniques attempt to reinforce the perception of details by adjusting light directions, which can be considered as an adjustment of normal / gradient vectors instead [16].

3 Background and Motivation

3.1 Volume Rendering Integral

The transport theory of light [6, 17, 7] is the basis for many volume rendering methods and ends up in following volume rendering integral:

$$I(\rho) = \int_0^{\rho} q(t)\, e^{-\sigma(0,t)} dt. \tag{1}$$

I is the intensity at position ρ along a ray, $\rho \in [0,B]$ (in three-dimensional space $\rho = B$ is the location of the background). q is the scattering function that can be identified with different models, and σ is the *optical depth* defined as

$$\sigma(t_1,t_2) = \int_{t_1}^{t_2} \kappa(t')\, dt', \tag{2}$$

where κ is the *absorption function*. For numerical evaluation, this integral defined on the interval $[0,B]$ is further subdivided into small not necessarily equidistant subintervals $[t_k,t_{k+1}],\ k = 0,\ldots,N-1$, where $t_0 = 0$ and $t_N = B$.

For the visualization of a continuous scalar field $g(\mathbf{x})$ ($\mathbf{x}$ is any point inside the data volume), a classification is further introduced to map scalar values to color $c(g)$ and to absorption $\kappa(g)$. The scattering function q generally uses $c(g)$, along with a local shading model to compute the color contribution of a point in space according to light sources.

3.2 Shading with a local surface illumination model

Shading introduces a color variation according to given light sources. When using the Blinn-Phong lighting model the previous Equation 1 can be written as:

$$\begin{aligned} I(\rho) \quad = \quad & \int_0^{\rho} [c(g(t))(K_a + K_d(\frac{\nabla g(t)}{\|\nabla g(t)\|} \cdot \mathbf{L})) \\ & + c_s(g(t)) K_s(\mathbf{H} \cdot \frac{\nabla g(t)}{\|\nabla g(t)\|})^{n_\delta}] e^{-\sigma(0,t)} dt \end{aligned} \tag{3}$$

K_a, K_d, K_s and n_δ respectively represent the ambient, diffuse, specular and shininess scalar coefficients of the Blinn-Phong model. In this equation, c_s is a specular color. A frequent choice is to assume it is white, thus removing it from the equation. But any specific specular color can be used if desired. An important point is the fact that Equation 3 is generally not used as such because of practical issues. The dot product between the light direction and the normalized gradient vector might become negative. Two solutions are generally possible. The first one consists in applying a maximum operator with zero (single-sided illumination). The second one consists in mirroring the gradient (double-sided illumination).

One condition is that the shading varies continuously to avoid visible discontinuity bands. Unfortunately, true pre-integrated gradient-based lighting is not realizable due to the high table dimension (the table requires 4 additional entries if we represent the normals as a pair of Euler angles, on top of the two scalar values). Lum *et al.* [8] proposed a solution consisting in linearly interpolating intensity values computed only on the front and back sample points. For this purpose, two tables are used: one fades front sample intensity out, while the

other fades the back sample intensity in. Each table represents shading weighted respectively towards front and back samples by a linear ramp. The rendered result is computed as a sum of the contribution of the two samples. Two more tables are necessary if specular highlights are further considered, as well as the usual pre-integrated table for the ambient term. Using a linear ramp is equivalent to interpolating the gradient along a segment between front and back samples. If the angle between two consecutive samples is important, this can introduce shading errors. Our approach aims at avoiding these errors. In what follows, we consider only directional lighting, e.g. the lighting vector $\mathbf{L}$ is constant over the whole scene. For illumination, we use the normalized gradient of the scalar density function $g(\mathbf{x}(t))$ on point $\mathbf{x}$ at position t upon the ray. To simplify notations because of subdivision into subintervals $[t_k, t_{k+1}]$, we further denote $g(\mathbf{x}(t_k+t))$ by $g(t)$. We also assume the absorption function in Equation (2), depending on $g(t)$. In what follows, we show how gradients can be non-linearly interpolated between front and back samples.

4 Non-linear gradient interpolation between samples

Phong normal interpolation on surfaces is a well studied problem [19]. As discussed in [19], a common solution for interpolating between two normals consists in applying a linear interpolation (a weighted sum) followed by normalization. In the case of volume rendering, gradients are generally interpolated and normalized on the fly using the available hardware trilinear filter. With pre-integrated volume rendering, there are two gradients to consider at the front and back of the slab. A solution that consists in using only one of them (or an average of both) to compute shading produces shading discontinuities, which is not desirable. A better solution consists in applying a linear interpolation: $\mathbf{N}(t) = \mathbf{N}_b t/D + \mathbf{N}_f(1-t/D)$ with $t \in [0,D]$ ($t=0$ means front and $t=D$ back) to smoothly vary from the front gradient $\mathbf{N}_f$ towards the back gradient $\mathbf{N}_b$ for a slab of size D. The resulting equation is:

$$\begin{aligned}\tilde{C} &= \int_0^D [c(g(t))(K_a + K_d \cdot \frac{(\mathbf{N}_b t + \mathbf{N}_f(D-t))}{D} \cdot \mathbf{L}) \\ &\quad + K_s(\mathbf{H} \cdot \frac{(\mathbf{N}_b t + \mathbf{N}_f(D-t))}{D})^{n_\delta}] e^{-\sigma(0,t)} dt\end{aligned}$$

However, since $\mathbf{N}_b$ and $\mathbf{N}_f$ are both normalized, the resulting norm of the interpolated vector $\mathbf{N}(t)$ is always lower than one. It is therefore important to introduce a normalization term $\eta(t) = \|\mathbf{N}(t)\|$, as suggested in [19], during the integration from front to back, e.g. to divide $\mathbf{N}(t)$ by $\eta(t)$. The equation is then:

$$\begin{aligned}\tilde{C} &= \int_0^D [c(g(t))(K_a + K_d \cdot \frac{(\mathbf{N}_b t + \mathbf{N}_f(D-t))}{D \cdot \eta(t)} \cdot \mathbf{L}) \\ &\quad + K_s(\mathbf{H} \cdot \frac{(\mathbf{N}_b t + \mathbf{N}_f(D-t))}{D \cdot \eta(t)})^{n_\delta}] e^{-\sigma(0,t)} dt\end{aligned}$$

Applying a linear interpolation of illumination between front and back samples instead of applying an illumination along an interpolated normalized vector, is equivalent to assuming the term $\eta(t)$ is always equal to 1 in-between the front and back samples. Subsequently, such an approach necessarily introduces a loss of energy during integration. This loss of energy is proportional to the angle between $\mathbf{N}_b$ and $\mathbf{N}_f$. Indeed, the committed error increases as the dot product between $\mathbf{N}_f$ and $\mathbf{N}_b$ becomes smaller, e.g. the angle between both vectors becomes greater. An example of illumination error introduced by assuming $\eta(t) = 1$ between two consecutive samples is shown in Figure 1. Note that the error is further increased when the illumination model has a high frequency component, like specular lobes. This figure compares the illumination evolution along the ray if correct normalization is used (a), e.g. $\eta(t) = \|\mathbf{N}(t)\|$ and if no normalization is used (b), e.g. $\eta = 1$. We show the diffuse and specular components, as well as both together. Part (c) is our own solution, as will be explained below. Part (b) corresponds to the approach of Lum *et al.* [8]. As can be seen, the error might become in some cases very high, compared to a correct reference result on (a).

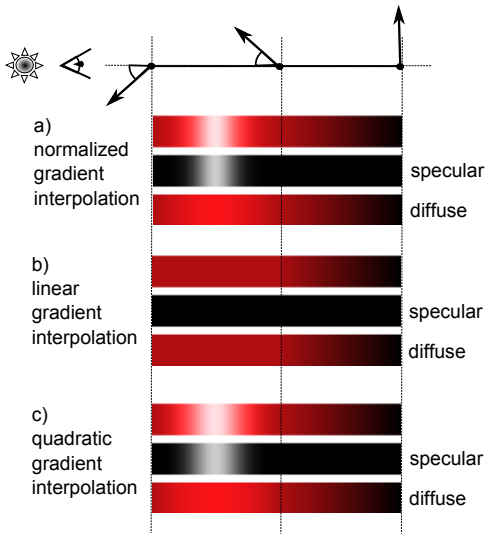

Fig. 1. Comparing results according to the type of gradient interpolation used between front and back samples. (a) interpolation with normalization (ground truth result). (b) linear interpolation without normalization. (c) non-linear polynomial interpolation (our solution).

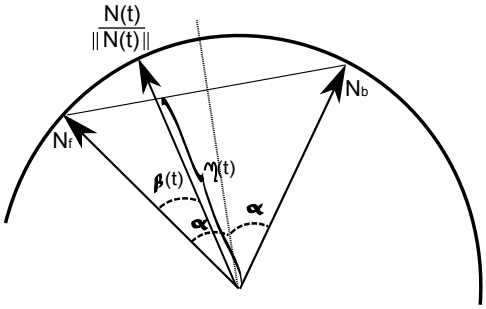

Fig. 2. Evaluating the normalization term $\eta(t)$ between front and back gradients.

By dividing $\mathbf{N}(t)$ by $\eta(t)$, the resulting vector no longer moves along a straight line but along an arc of circle, as shown by Figure 2. This corresponds to a non-linear interpolation of the gradient vector between front and back samples. As can be seen, the maximum value for $\eta(t)$ is reached at half way between $\mathbf{N}_f$ and $\mathbf{N}_b$. $\eta(t)$ can be expressed as follows:

$$\eta(t) = \frac{cos(\alpha)}{cos(\beta(t) - \alpha)} \tag{4}$$

where $\alpha = acos(\mathbf{N}_f \cdot \mathbf{N}_b)/2$. Because of linear interpolation, $\beta(t)$ is an angle that varies linearly from 0 to 2α, e.g. $\beta = 2\alpha t/D$. We note that for an angle of $\alpha = 45°$, assuming $\eta = 1$ introduces an error that reaches 41% at halfway between $\mathbf{N}_b$ and $\mathbf{N}_f$, since at half-way $1/\eta = 1/cos(\pi/4) = 1.414$.

To estimate if it is worth introducing a non-linear interpolation between front and back samples for visualizing volume datasets, we first have to evaluate how large angles between gradients of consecutive sample points are effectively, for a given ray traversal. Figure 3 shows such an evaluation for the CT head dataset. The angle variation mainly depends on the technique used to compute the gradients, as well as on the sampling distance (the greater the distance, the more the gradients are likely to be different between front and back samples). On this figure the three rows show results for three different gradient computation techniques. The left column shows the rendered dataset. The middle

and right visualizes the gradient variation along the ray for respectively 1 voxel sampling step and 0.5 voxel sampling step, for angles α greater than 30 degrees and 45 degrees respectively (this means an angle of 60, respectively 90 degrees between the front and back sample gradients). Dark zones depict the amount of samples that have such angles. That is, the darker the zones are, the more there are consecutive samples with angle greater than 60 and 90 degrees respectively along the ray. The first row uses gradients that were pre-computed using a Sobel operator. The second row illustrates the use of a cubic spline interpolation to compute gradients. Finally, the last row illustrates exaggerated shading as inspired by [16]. In our case, to exaggerate the perception of details, we have increased for each sample the angle between the actual gradient on this sample with respect to a mean gradient around this sample for a given neighborhood of voxels. That is, calling $\mathbf{N}_m$ the average gradient around sample (i,j,k) in a given neighborhood and $\mathbf{N}$ the gradient on (i,j,k), we replace the latter by $\mathbf{N'} = \mathbf{N} + \varepsilon(\mathbf{N} - \mathbf{N}_m)$, where $\varepsilon >= 0$ is the amplitude of exaggeration (0 means no exaggeration). For the example of Figure 1, we used $\varepsilon = 1$ and a neighborhood of 5^3 voxels. Exaggerated shading improves the perception of small scale details and increases the gradient variation between front and back samples.

All of these examples show that datasets can contain locally important gradient variations between front and back samples (further increased when the sampling step is high, or when details are exaggerated). For such cases, it seems important to apply a correct gradient interpolation to avoid introducing shading errors.

To do pre-integrated classification with non-linear gradient variation between front and back samples, a brute force approach could consist in using tables that would be further depending on α. In practice this means using 3D pre-integration tables instead of two-dimensional tables, the third index being α. However, 3D tables are not very practical, because they can only hardly be updated at real time rates during transfer function changes, especially if more than a single table must be pre-computed (as for the diffuse and specular terms). In addition, 3D tables are texture memory consuming, while datasets with pre-computed gradients may already require much texture memory. In addition, variable sample steps would become almost impossible to process with GPU hardware, since it would result in 4D tables. For all of these reasons it seems desirable to keep the tables two-dimensional.

Our solution consists in exploiting the fact that a correctly interpolated normal $\mathbf{N}(t)$ moves along a circle (as visible in Figure2). A circle arc can be quite well approximated by a quadratic polynomial function. We propose to use following approximation:

$$\eta(t) = \frac{cos(\alpha)}{x^2(cos(\alpha)-1)+1}, x = (\beta(t)-\alpha)/\alpha = 2t/D-1 \qquad (5)$$

Note that we have for $t=0$, $\beta=0$ and $\eta=1$, for $t=0.5D$, $\beta=\alpha$ and $\eta=cos(\alpha)$ and for $t=D$, $\beta=2\alpha$ and $\eta=1$. These three cases are matching the exact values. By using such a normalization term, the error becomes very low with respect to a real arc of circle, compared to an interpolation along a straight line (e.g. taking $\eta=1$). Indeed, for an angle of 90°, the maximal error is now about 0.3% (compared to the previous 41% without any normalization). An example of shading variation using such a normalization term is shown on bottom row of Figure1. The difference with the top row (correct interpolation) is negligible.

Using the previous approximate normalization term, we end up with following final expression for the normalized gradient between front an back samples:

$$\frac{\nabla g(t)}{\|\nabla g(t)\|} \approx \frac{(1+(2\frac{t}{D}-1)^2(cos(\alpha)-1))(\frac{\nabla g(0)}{\|\nabla g(0)\|}t + \frac{\nabla g(D)}{\|\nabla g(D)\|}(D-t))}{D\cdot cos(\alpha)} \qquad (6)$$

5 Pre-Integrated Classification including Blinn-Phong Shading

In this section, we describe how pre-integrated volume rendering can be realized by considering the previously defined gradient interpolation scheme between the front and back samples. We first consider the Blinn-Phong illumination model. As for classical pre-integration, we assume the scalar values between two consecutive sampling points change linearly. For rendering, we pre-compute gradients from the density function $g(t)$.

The ambient contribution of the volume rendering integral (equation 3) between two sample points can be computed with traditional pre-integrated volume rendering and needs no further discussion. In the following two subsections we describe respectively the diffuse and specular parts.

5.1 Diffuse term

By calling $\mathbf{N}_f$ and $\mathbf{N}_b$ the normalized gradients on the two consecutive front and back samples and by using the previous gradient interpolation scheme (equation 6), the corresponding diffuse part of equation 3 for a sampling step of length D is:

$$\begin{aligned}\tilde{C_d} &= K_d \int_0^D [c(g(t))\frac{(2t/D-1)^2(cos(\alpha)-1)+1}{cos(\alpha)} \\ &\quad \cdot \frac{(\mathbf{N}_b t + \mathbf{N}_f(D-t))}{D}\cdot\mathbf{L})]e^{-\sigma(0,t)}dt\end{aligned}$$

The variable substitution $s' := g(t)$ where $g(t) = t(S_b - S_f)/D + S_f$ and $ds' = (S_b - S_f)/D \cdot dt$ leads to the following pre-integrated formulation:

$$\begin{aligned}\tilde{C_d}(S_f,S_b) &= K_d \cdot d \int_{S_f}^{S_b} [c(s')\frac{(2s-1)^2(cos(\alpha)-1)+1}{cos(\alpha)} \\ &\quad \cdot(\mathbf{N}_b\cdot\mathbf{L}s + \mathbf{N}_f\cdot\mathbf{L}(1-s))]e^{-d\sigma(S_f,s')}ds' \qquad (7)\end{aligned}$$

with $d = D/(S_b - S_f)$ and $s = (s' - S_f)/(S_b - S_f) = t/D$.

Using the following notation:

$$I_k(S_f,S_b) = d\int_{S_f}^{S_b} c(s')\cdot s^k \cdot exp(-d\sigma(S_f,s'))ds'$$

(k is an integer exponent value for s) and by separating the constant, linear (s), quadratic (s^2) and cubic (s^3) terms, the equation becomes:

$$\tilde{C_d} = K_d(a_0(L)I_0 + a_1(L)I_1 + a_2(L)I_2 + a_3(L)I_3) \qquad (8)$$

where

$$\begin{aligned}a_0(L) &= \mathbf{N}_f\cdot L \\ a_1(L) &= \mathbf{N}_b\cdot L - (1+r)\mathbf{N}_f\cdot L \\ a_2(L) &= r(2\mathbf{N}_f\cdot L - \mathbf{N}_b\cdot L) \\ a_3(L) &= r(\mathbf{N}_b\cdot L - \mathbf{N}_f\cdot L) \\ r &= \frac{4(\cos(\alpha)-1)}{\cos(\alpha)}\end{aligned}$$

All I_k terms in this equation depend only on S_f and S_b, and thus can be calculated and tabulated into $2D$ look-up tables. During rendering the four tables can be accessed and then the constants a_k computed, and finally the sum performed. This yields the diffuse component of shading taking into account an interpolation of the gradient with negligible error compared to an interpolation along a circle even if α is important, e.g. even if $\mathbf{N}_b$ and $\mathbf{N}_f$ are very different (for example 90°).

As mentioned in section 3.2, Equation 7 is generally not used as such because of negative dot product values. We must either apply a maximum operator with zero or a mirroring of the gradient to make sure values are positive. For the first case, we have to further introduce a $max(\_,0)$ function into the integral, but such a function is not easy to integrate. To circumvent this difficulty, we distinguish three cases:

- if $\mathbf{N}_f\cdot\mathbf{L} > 0$ and $\mathbf{N}_b\cdot\mathbf{L} > 0$, the previous formulation can be used without *max* operator.

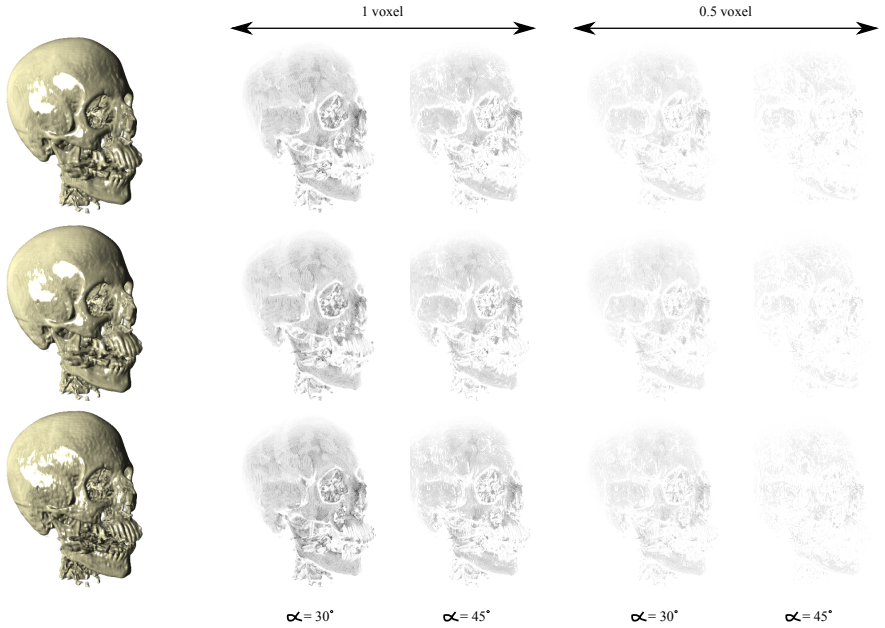

Fig. 3. Gradient variation between consecutive front and back samples for the CT head dataset using three different gradient computation techniques and two different sampling steps (1 and 0.5 voxel).

- if either $\mathbf{N}_f \cdot \mathbf{L} < 0$ or $\mathbf{N}_b \cdot \mathbf{L} < 0$, we use an approximation that consists in fading in or out the non-negative value, e.g. $(\mathbf{N}_b \cdot \mathbf{L}s + \mathbf{N}_f \cdot \mathbf{L}(1-s))/\eta(t)$ is replaced by $\mathbf{N}_b \cdot \mathbf{L}s$ or $\mathbf{N}_f \cdot \mathbf{L}(1-s)$. We thus obtain $\tilde{C_d}(S_f, S_b) = K_d \mathbf{N}_f \cdot \mathbf{L}(I_0 - I_1)$ and $\tilde{C_d}(S_f, S_b) = K_d \mathbf{N}_b \cdot \mathbf{L} I_1$ respectively.
- if $\mathbf{N}_f \cdot \mathbf{L} < 0$ and $\mathbf{N}_b \cdot \mathbf{L} < 0$, the result is just 0.

For the second case, we can just mirror the gradient responsible for the negative value and apply the interpolation as usual. But this will introduce an approximation for slabs where one gradient (whether front or back) is correctly oriented and the other one is inverted with respect to the light. Indeed, on such slabs, an accurate interpolation is no longer a single arc of circle but two arcs of circles, joining at a vector perpendicular to the light (at the location where the gradient becomes inverted, the shading must be null). Applying a straight interpolation with an inverted gradient, thus tends to add some energy in this particular case.

5.2 Specular term

Let us now consider the case of specular reflection. Under the same assumptions used to compute the diffuse reflection, the specular part can be written as:

$$\tilde{C_s}(S_f, S_b) = K_s \cdot d \int_{S_f}^{S_b} \left(\frac{(2s-1)^2 (cos(\alpha) - 1) + 1}{cos(\alpha)} \cdot (\mathbf{N}_b s + \mathbf{N}_f (1-s)) \cdot \mathbf{H} \right)^{n_\delta} e^{-d\sigma(S_f, s')} ds' \quad (9)$$

It is difficult to evaluate this integral numerically because of the power n_δ. We therefore propose to replace the usual specular lobe by a specular cone. That is, we use the approximation $(1+a)^n = 1 + na + o(x)$. This slightly modifies the shape of the specular highlight without, however, removing visual cues. Calling $\mathbf{N} \cdot \mathbf{H} = cos(\theta)$, we replace $cos(\theta)^{n_\delta}$ by $1 + n_\delta(cos(\theta) - 1)$. As shown by Figure 4, this approximation does not affect the perception of specular highlights. The left represents the classical lobe and the right the approximated cone. In particular in both cases, the derivative on $\theta = 0$ (e.g. on the maximal value of the highlight) is 0, which ensures a smooth spot. A visual difference is noticeable on a perfect sphere (second row), but this represents an extreme case. Indeed, as soon as there are local geometric variations (see noisy sphere), the visual difference becomes almost non-perceptible. Figure 5 shows an example on the skull dataset. The left is rendered with a classical Phong lobe and the right with a cone setting $n_\delta = 20$. When zooming in, slight differences are visible, but visual cues remain globally well preserved. The advantage of using a cone instead of a lobe is that it allows for an accurate straightforward integration.

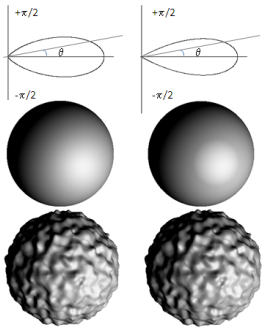

Fig. 4. Replacing the specular lobe (left) by a cone (right) for $n_\delta = 10$. The visual difference on 3D surfaces is low, and specular hightlights remain well perceptible.

By using a cone instead of a lobe for the specular term, we obtain:

$$\tilde{C_s}(S_f, S_b) = K_s \cdot d \int_{S_f}^{S_b} [1 + n_\delta (\frac{(2s-1)^2 (cos(\alpha) - 1) + 1}{cos(\alpha)} \cdot (\mathbf{N}_b s + \mathbf{N}_f (1-s)) \cdot \mathbf{H} - 1)] e^{-d\sigma(S_f, s')} ds'$$

As for the diffuse case, we introduce the notation:

$$I'_k(S_f, S_b) = d \int_{S_f}^{S_b} s^k \cdot exp(-d\sigma(S_f, s')) ds'$$

and separate the constant, linear (s), quadratic (s^2) and cubic (s^3) terms. The equation finally becomes:

$$\tilde{C_s} = K_s (I'_0 + n_\delta (a_0(H) I'_0 + a_1(H) I'_1 + a_2(H) I'_2 + a_3(H) I'_3) - n_\delta I'_0 \quad (10)$$

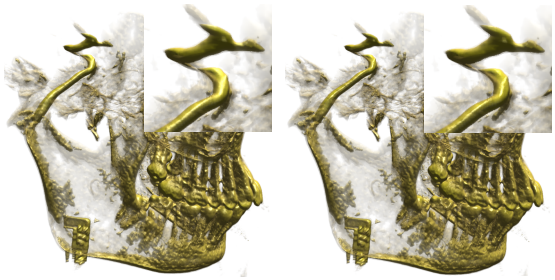

Fig. 5. Comparing a specular lobe (left) with a cone (right) in the case of the skull dataset for $n_\delta = 20$. Differences are just noticeable on the zooms, but do not affect global visual cues.

As for the diffuse case, values might become negative depending on the dot product between $\mathbf{N}(s')$ and $\mathbf{H}$ and depending on n_δ. So, we likewise have to invert gradients or introduce a $max(\_,0)$ function to avoid such negative values. The values become negative when the dot product between H and the interpolated gradient $N(s')$ is lower than $thresh = (n_\delta - 1)/n_\delta$. For the $max(\_,0)$ function, we have to distinguish following three cases:

- if $\mathbf{N}_f \cdot \mathbf{H} > thresh$ and $\mathbf{N}_b \cdot \mathbf{H} > thresh$, the previous formulation can be used, since all values are positive.

- if either $\mathbf{N}_f \cdot \mathbf{H} < thresh$ or $\mathbf{N}_b \cdot \mathbf{H} < thresh$, we use an approximation that consists in fading in or out the non-negative value, e.g. we use $\tilde{C_s}(S_f, S_b) = K_s(I_0(1-n_\delta) + n_\delta \mathbf{N}_b \cdot \mathbf{L} I_1)$ and $\tilde{C_s}(S_f, S_b) = K_s(I_0(1+n_\delta(\mathbf{N}_f \cdot \mathbf{L} - 1)) - n_\delta \mathbf{N}_f \cdot \mathbf{L} I_1$ respectively.

- if $\mathbf{N}_f \cdot \mathbf{L} < 0$ and $\mathbf{N}_b \cdot \mathbf{L} < 0$, the result is different from the diffuse case, just 0, since there might be a highlight in-between the front and back samples. To compute the relative position t' where the maximum specular value is reached, we can use the derivative of the cubic form, which is a quadratic form, and solve the corresponding equation of second degree. Note that a similar technique has been used in [9] in order to find out the maximal shading value of an analytical form. Then, we can check whether the solution t' is in $[0,1]$ or not, and compute the value of the corresponding specular term on t'. If it is positive, there is a maximum on t', and subsequently a highlight between the front and back samples. A way of doing integration would consist in fading in the specular value from 0 to t' and then fading it out towards 1. But this is an impractical solution, since there would be as many tables as positions of t'. Therefore, we propose to introduce an approximation using a single additional table I'':

$$I''(S_f, S_b) = d \int_{S_f}^{S_b} (s < 0.5 ? 2s : 2 - 2s) \cdot exp(-d\sigma(S_f, s'))ds'$$

The last approximation consists in assuming that the maximum specular value is reached at half-way between the front and back samples, that is for $t' = 0.5$. This will make sure that there will be a visible highlight between the front and back samples, but not at the exact position between the samples, which makes it either a bit too visible or a bit less visible with respect to the actual position on the concerned slab because of occlusion.

5.3 Other shading functions

The Blinn-Phong lighting model that we used in the previous subsections is not the only available local surface lighting model. Since our technique is based on a gradient interpolation between front and back samples, it can be used a priori with any other lighting model, provided it can be expressed as a polynomial form. In what follows, we take the example of the illustrative Gooch *et al.* [3] lighting model. This model modifies Equation 3 as follows:

$$\begin{aligned} I(\rho) &= \int_0^\rho [(\frac{1}{2}(c_{blue} + a \cdot c(g(t))K_d)(1 + \mathbf{N} \cdot \mathbf{L}) \\ &+ \frac{1}{2}(c_{yellow} + b \cdot c(g(t))K_d)(1 - \mathbf{N} \cdot \mathbf{L}))] e^{-\sigma(0,t)} dt \quad (11) \end{aligned}$$

where coefficients a and b control the prominence of the object color and strength of luminance shift (for instance $a = 0.2$ and $b = 0.6$) and c_{blue} and c_{yellow} determine the subtone.

In its pre-integrated form, the diffuse part of the previous equation can be written in this case as:

$$\tilde{C_d} = a_0(L) I_0^{Gooch} + a_1(L) I_1^{Gooch} + a_2(L) I_2^{Gooch} + a_3(L) I_3^{Gooch}$$

with

$$\begin{aligned} I_k^{Gooch}(S_f, S_b) &= \frac{d}{2} \int_{S_f}^{S_b} (c_{blue} - c_{yellow} + K_d c(s')(a-b)) \\ &\cdot s^k \cdot exp(-d\sigma(S_f, s'))ds' \end{aligned}$$

Note that for this shading model negative values of the dot product $\mathbf{N}.\mathbf{L}$ do not need to be processed specifically, since the final resulting shading values are always positive.

6 Results

We conducted our tests using OpenGL and GLSL with a viewport of 800x800 pixels on an Intel Core 2Quad Q9300, with 4GB RAM and with a Nvidia GeForce GTX 280 graphics board (with 1Gb memory). We implemented a GPU-based ray casting rendering scheme on the fragment shader without optimization except early ray termination. All pre-integration tables use a 16bit unsigned short representation on the GPU memory. In the following, we first discuss visual quality issues and make a comparative study with existing techniques.

Figure 6 shows a set of results obtained for different datasets and pre-integration schemes. From top to bottom, we show the bonsai dataset (256x256x256), the CT Head dataset (256x256x225) and the Knee dataset (379x229x305). The parameters for Phong lighting are $Ka = 0$, $Kd = 1$, $Ks = 2$, $n_\delta = 50$ for the bonsai dataset and $Ka = 0$, $Kd = 1$, $Ks = 3$, $n_\delta = 80$ for the CT Head and the Knee datasets. For all examples, we use a Phong cone instead of a Phong lobe. The first column (a) shows a reference image obtained without pre-integration and with a true gradient interpolation between front and back samples using a high sampling rate (20 sub-samples). The remaining columns show respectively results obtained using: (b) a constant gradient (the average between front and back gradient values), (c) a linear gradient interpolation (e.g. the approach of [8]), and (d) our approach (non-linear gradient interpolation). We also show close up views as well as difference images to the reference (a): low differences result in black color values during ray traversal. As visible in these images, (c) and (d) are systematically closer to the reference compared to (b). With our approach (d) the difference is almost everywhere null (difference images are almost completely black). However, there are a few visible differences on some locations for the Knee dataset. In fact, our approach tends to exaggerate intensity around specular highlights, which is related to the fact that the quadratic curve used for interpolation is not exactly an arc of circle. The normalization term might be too low, which, in turn, results in vectors that have a norm greater than one. Hence, some energy is added. For rendering, we have used respectively a large 2-voxel sampling step ($2/256$) for the bonsai dataset and a 1-voxel step ($1/256$) for the CT Head and Knee datasets. At far distance, differences between the three different pre-integration schemes are hardly perceptible. But, on the zooms three types of benefits resulting from using a non-linear gradient interpolation are visible. The first one (first row, bonsai dataset), is that for large sampling steps (here, 2-voxels) remaining stripes are less obvious because of the smoothness of shading. We note that, pre-integration does not necessarily remove all visible stripes, especially for large sampling steps, as it has been shown in [14]. The smoother shading reduces the visibility of

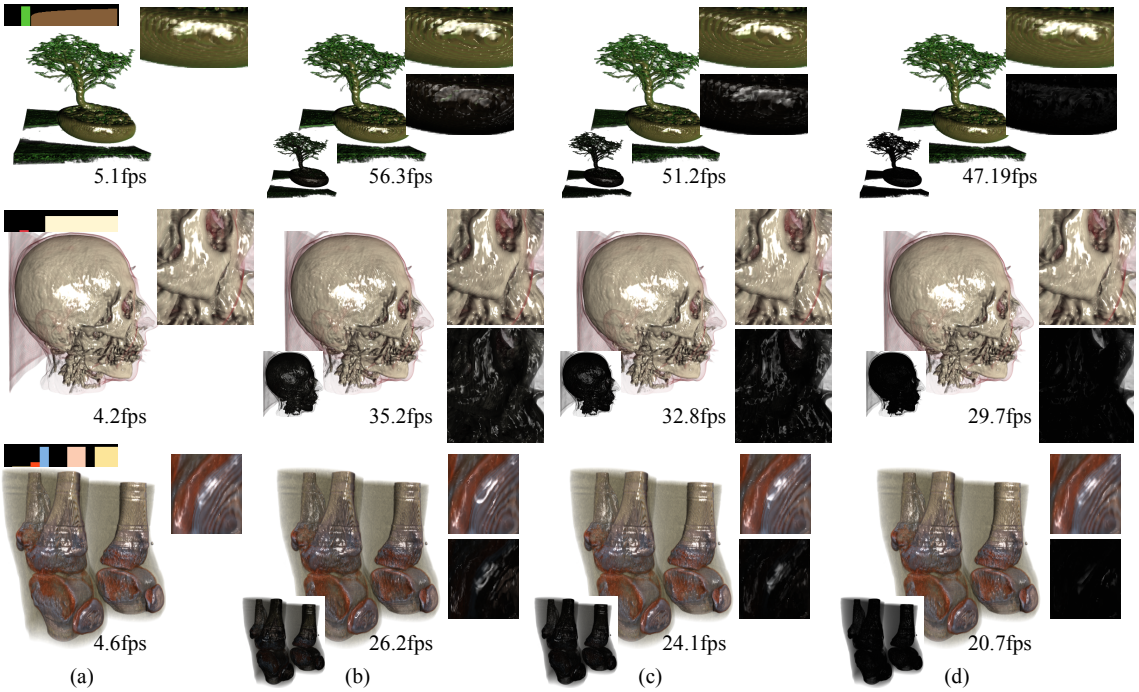

Fig. 6. Comparing different pre-integration schemes. (a) accurate gradient interpolation using a high sub-sampling rate between samples and no pre-integration (reference image), (b) constant gradient using the mean between front and back values, (c) linear gradient interpolation, (d) our approach.

these stripes, compared to constant gradient and linear gradient interpolation. The second benefit (second row, CT Head dataset) is that it reduces a kind of apparent noise that appears when the frequency of specular highlights is high (individual small bright dots), which is also related to an improved global smoothness. Finally, the last, probably most important benefit, is that the contours of specular highlights are more accurately depicted. This is an important issue for the correct visual comprehension of underlying objects shapes. Concerning timings (provided within the figure), our approach is about 10% to 20% slower than a linear gradient interpolation, which we explain by the fact that we must use two more texture accesses (only two additional tables are used, given the fact that for the specular color we use only a single color channel, since we assume the specular color is white).

In order to better perceive the difference between the three pre-integration schemes at global object scale, we further used a dataset with an object of low curvature (accordingly the lighting also changes slowly, along with committed shading errors) and with a thin layer so as to maximize the gradient variation between front and back samples. The engine dataset meets well these criteria. Figure 7 illustrates the comparative study for this dataset. The left shows a ground truth result obtained using a very low sampling step of $0.1/256$ without pre-integration. Then, we show respectively the gradient variation, our result, the approach of [8] and pre-integration with constant gradient. The first row is for a sampling step of $1/256$, the second one for a sampling step of $0.5/256$. In this case, differences between the pre-integration schemes are better perceivable. These differences decrease as the sampling rate becomes greater, since the variation of the gradients also becomes smaller (due to trilinear interpolation on voxels). Yet, even for a $0.5/256$ step, our approach remains closer to the ground truth (see upper right corner of the engine).

Figure 8 finally shows two examples of illustrative rendering using the Gooch et al. [3] local shading model using two different datasets: top is the aneurism (256x256x256) dataset and bottom is the CT head dataset. The left shows the reference and the right our result obtained for a sampling step of $0.5/256$. Nearly no difference is visible between our result and the reference. For the aneurism the reference (left) has a 1.8 fps framerate for a sampling step of $0.1/256$ and our result (right) a 16.3 fps framerate for a sampling step of $0.5/256$. For the CT head the reference (left) has a 3.3 fps framerate for a sampling step of $0.1/256$ and our result (right) a 18.7 fps framerate for a sampling step of $0.5/256$.

7 Conclusions and Future Work

In this paper, it has been shown that non-linear gradient variations along front and back samples can be integrated into the pre-integrated volume rendering framework for improved shading computation accuracy. Improvements are obtained when gradients are strongly varying and/or shading functions have high frequency components. Concerning the specular term of the Phong shading model, we have replaced the lobe by a cone. From a visual point of view the cone is close to the lobe and well preserves similar visual cues. Our technique is also compatible with other local surface shading models, like non-photorealistic ones, as long as these can be expressed as polynomials.

Our approach requires more tables than an approach based on shading interpolation (linear gradient interpolation), which requires some more texture accesses in hardware implementations. Performance degradation remains however low. An interesting property is that tables remain two-dimensional, so that they can be straightforwardly updated during transfer function editing.

Using a non-linear interpolation scheme, as we proposed, does implicitly increase the degree of the signal reconstruction for the shading part. Therefore, such approaches not only avoid visual shading discontinuity bands, but further improve the global smoothness compared to methods that use a constant gradient or a linear gradient interpolation. However, as for all pre-integration techniques, we assume the underlying signal is piecewise linear for computing the occlusion and color components. As it has been already shown in previous works, there might remain visible stripes if the integration step is set too large. Our approach reduces the visibility of these stripes, but does not remove them. To further improve results in such cases, it seems important to improve the reconstruction of the signal in-between front and back samples. However, the problem that must be solved in this case, is the resulting increase of dimensionality.

Fig. 7. Rendering of the engine dataset using different pre-integration schemes. Top is for 1 voxel sampling step. Bottom is for 0.5 sampling step.

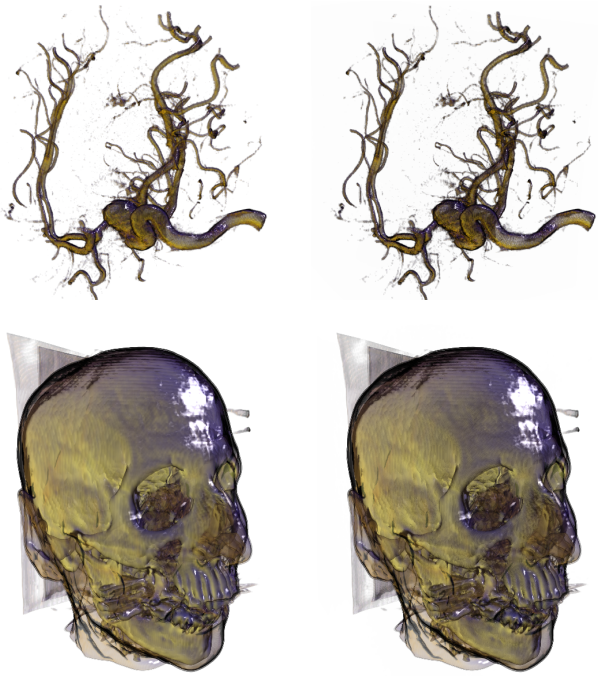

Fig. 8. Two examples of datasets rendered using an illustrative lighting model. Left is reference image (no pre-integration using a high sampling rate) and right is our result using a $0.5/256$ sampling step.

Acknowledgments

This work has been funded by the french national science fundation (ANR) under project ATROCO number ANR-07-MDCO-001.

References

[1] D. Ebert and P. Rheingans. Volume illustration: non-photorealistic rendering of volume models. In *VIS '00: Proceedings of the conference on Visualization '00*, pages 195–202, 2000.

[2] K. Engel, M. Kraus, and T. Ertl. High-quality pre-integrated volume rendering using hardware-accelerated pixel shading. In *HWWS '01: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, pages 9 – 16, 2001.

[3] A. Gooch, B. Gooch, P. Shirley, and E. Cohen. A non-photorealistic lighting model for automatic technical illustration. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 447–452, 1998.

[4] M. Hadwiger, I. Viola, T. Theussl, H. Hauser, and H. Hauser. Fast and flexible high-quality texture filtering with tiled high-resolution filters. In *In Proceedings of Vision, Modeling, and Visualization*, 2002.

[5] J.-F. E. Hajjar, S. Marchesin, J.-M. Dischler, and C. Mongenet. Second order pre-integrated volume rendering. In *Visualization Symposium, 2008. PacificVIS '08. IEEE Pacific*, pages 9–16, 2008.

[6] J. Kajiya. The rendering equation. *SIGGRAPH Comput. Graph.*, 20(4):143–150, 1986.

[7] W. Krueger. The application of transport theory to visualization of 3d scalar data fields. *Proc. IEEE Visualization*, pages 273–280, 1990.

[8] E. Lum, B. Wilson, and K. Ma. High-quality lighting and efficient pre-integration for volume rendering. In *Proceedings of the Joint Eurographics-IEEE TVCG Symposium on Visualization 2004*, pages 25–34, 2004.

[9] S. Marchesin and G. C. de Verdiere. High-quality, semi-analytical volume rendering for amr data. *IEEE Transactions on Visualization and Computer Graphics*, 15:1611–1618, 2009.

[10] N. Max, P. Crawfis, and P. Hanrahan. Area and volume coherence for efficient visualization of 3d scalar functions. *SIGGRAPH Comput. Graph.*, 24(5):27–33, 1990.

[11] K. Moreland and E. Angel. A fast high accuracy volume renderer for unstructured data. In *VV '04: Proceedings of the 2004 IEEE Symposium on Volume Visualization and Graphics*, pages 9–16, 2004.

[12] B. T. Phong. Illumination for computer generated pictures. *Commun. ACM*, 18(6):311–317, 1975.

[13] S. Roettger and T. Ertl. A two-step approach for interactive pre-integrated volume rendering of unstructured grids. In *VVS '02: Proceedings of the 2002 IEEE symposium on Volume visualization and graphics*, pages 23 – 28, 2002.

[14] S. Roettger, S. Guthe, D. Weiskopf, T. Ertl, and W. Strasser. Smart hardware-accelerated volume rendering. In *VISSYM '03: Proceedings of the symposium on Data visualisation 2003*, pages 231 – 238, 2003.

[15] S. Roettger, M. Kraus, and T. Ertl. Hardware-accelerated volume and isosurface rendering based on cell-projection. In *VIS '00: Proceedings of the conference on Visualization '00*, pages 109 – 116, 2000.

[16] S. Rusinkiewicz, M. Burns, and D. DeCarlo. Exaggerated shading for depicting shape and detail. *ACM Trans. Graph.*, 25(3):1199–1205, 2006.

[17] P. Sabella. A rendering algorithm for visualizing 3D scalar fields. *SIGGRAPH Comput. Graph.*, 22(4):51–58, 1988.

[18] J. P. Schulze, M. Kraus, U. Lang, and T. Ertl. Integrating pre-integration into the shear-warp algorithm. In *VG '03: Proceedings of the 2003 Eurographics/IEEE TVCG Workshop on Volume graphics*, pages 109 – 118, 2003.

[19] C. van Overveld and B. Wyvill. Phong normal interpolation revisited. *ACM Trans. Graph.*, 16(4):397–419, 1997.

[20] P. Williams and N. Max. A volume density optical model. In *Proceedings of the 1992 workshop on Volume visualization*, pages 61–68, 1992.

[21] P. Williams, N. Max, and C. Stein. A high accuracy volume renderer for unstructured data. *IEEE Transactions on Visualization and Computer Graphics*, 4(1):37 – 54, 1998.