Second Order Pre-Integrated Volume Rendering

Jean-François El Hajjar* XLIM, University of Limoges
Stephane Marchesin† LSIIT, University of Strasbourg
Jean-Michel Dischler‡ LSIIT, University of Strasbourg
Catherine Mongenet§ LSIIT, University of Strasbourg

Abstract

In the field of Volume Rendering, pre-integration techniques for arbitrary transfer functions has certainly led to the most significant and convincing results both quality and performance wise on standard PC consumer graphics. By showing that the ideal scalar signal along the cast rays is better approximated by a succession of polynomial curves as opposed to linear segments, we propose a new method for pre-integrated volume rendering. This method is based on a second order polynomial interpolation of the scalar values, allowing it to converge more rapidly towards the integration of a volume reconstructed by a trilinear filter. This approach manages to capture the smoothness of the volume's details without the need of further ray resampling, and consequently succeeds in reducing the visual artefacts in comparison to previous techniques. Futhermore, we adapt an existing technique to compute our pre-integration tables using the GPU, thus making our approach suitable for transfer function manipulations.

Keywords: Volume Rendering, Pre-Integration, Newton-Cotes Formulas, Trilinear Filtering, GPU

Index Terms: I.3.3 [Computer Graphics]: Picture/Image Generation—Viewing Algorithms; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture; I.3.1 [Computer Graphics]: Hardware Architecture—Graphics Processors

1 Introduction

Thanks to tremendous improvements in consumer graphics hardware over the recent years, volume rendering can now be made available on standard, off the shelf, personal computers. In particular, the range of functionalities brought by programmable graphics processors yielded a number of improvements in the volume rendering field : given the flexibility of such processors, higher quality volume rendering techniques can now take advantage of hardware acceleration.

In this context, many efforts have been spent on approximating the integration of a volume in a discrete way. For arbitrary transfer functions, pre-integration of the transfer function [3, 18] has proven to be an efficient way to improve the reconstruction both performance and quality wise. In another perspective, the impact of filtering kernels on the rendered images has also been studied [5], subsequently leading to higher quality rendering techniques. However, its correlation with the pre-integration of the transfer function has not yet been investigated to the authors' knowledge.

In this paper, we study this correlation in the case of a trilinear filtering kernel, currently the only filter available natively on standard consumer graphics. By observing inherent properties of a trilinearly reconstructed signal along segments, we derive a method based on a 2^{nd} order polynomial for pre-integration. This technique corresponds to a 2^{nd} order numerical integration scheme, and we show that higher quality results are achieved without the need for further ray resampling in comparison to standard pre-integrated volume rendering.

*jean-francois.el-hajjar@xlim.fr
†marchesin@icps.u-strasbg.fr
‡dischler@dpt-info.u-strasbg.fr
§mongenet@icps.u-strasbg.fr

2 Related Works

The first hardware accelerated volume rendering technique was introduced more than a decade ago by Cullip *et al* [2]. It accumulates object-aligned slices of a volume dataset to perform volume integration. The algorithm was improved by Cabral *et al* [1] who proposed to accumulate world-aligned slices of a 3D texture, thereby improving the quality of the rendered images. Thanks to the increasing flexibility of graphics hardware, GPU-based ray casting approaches have been made possible. These methods can be classified into two main categories. The first one needs multiple rendering passes to accumulate the contributions of the fragments [8, 17, 22], while it uses occlusion queries to identify termination. The second one exploits the availability of loops and fast conditionals in Shader Model 3.0 compatible GPUs. These techniques use a single rendering pass [6, 20], where one ray is cast for each pixel using a loop that iterates through the volume and accumulates color values into a floating point register.

The concept of pre-integrated volume rendering emanates from the early works on tetrahedral meshes [10, 21]. Roettger *et al* [18] extended pre-integration to account for arbitrary transfer functions in the context of the PT (Projected Tetrahedra) algorithm. The precomputation of the 3D table has later been accelerated with a GPU implementation using an iterative scheme [16]. Engel *et al* [3] futhermore adapted the technique to the slicing approach of regular grids. By neglecting the distortion of the segments due to perspective projection, they used a 2D pre-integration table allowing higher quality visualization while preserving interactive frame rates. In order to speed up the computation process of the pre-integration tables, polynomial approximations of the color values [4] or incremental subrange integration scheme [9] have been proposed. Other interesting approaches define the transfer function as being gaussian functions [7] or Bernstein polynomials [19], allowing the analytical resolution of the volume rendering integral while preserving interactive frame rates.

In the context of isosurface rendering, some methods proposed to take into account the interpolation scheme related to the discrete scalar field in order to find accurate ray-isosurface intersections. Regular grids with trilinear reconstruction have been considered in [13], second-order splines in [15], second-order tetrahedras in [23–25] as well as spectral/*hp* elements in [11].

In the field of numerical integration, numerous schemes exist. Among these, the Newton-Cotes formulas [14] are a group of formulas based on Lagrange polynomials for integrating discrete functions. In an early work [12], three methods have been proposed for controlling the quality of the approximation of the volume rendering integral, two of which being based on the Newton-Cotes formulas at respectively order 1 and 2. Given a ray segment contained in a voxel as well as a user defined error threshold, the remainder term of the chosen Newton-Cotes formula is used to solve for an appropriate subdivision step, the latter allowing to locally integrate the color and opacity values at the desired precision when reinjected in the Newton-Cotes formula. However, the coarse estimation of the derivative bounds as well as the heuristic used for estimating the global error on the ray does not guarantee optimal results, ease of implementation and low computational cost having been prefered by the authors. More importantly, this method requires preclassification of the volume data which is commonly known to be subject to artefacts and loss of details. Finally, since no practical study in terms of visualization has been proposed in this paper, it is difficult to relate the error reduction to the quality improvements of the rendered datasets.

In this paper, we propose to use a 2^{nd} order Newton-Cotes formula to achieve post-classified volume rendering with arbitrary transfer functions. We furthermore provide an explicit analysis of the integration error to support the theory behind our method, visual results confirming an improvement over past approaches.

3 Analysis of Existing Rendering Techniques

Various hardware accelerated volume rendering techniques have been developed. In this Section, we classify these techniques according to the order of integration after reviewing a non exhaustive theoretical background in signal reconstruction for volume rendering.

3.1 Signal Reconstruction

Visualizing a discrete volume consists in solving for every pixel the commonly called volume rendering integral :

$$\mathbf{C} = \int_0^D \mathbf{c}(\mathscr{F}(\mathbf{r}(t)))\tau(\mathscr{F}(\mathbf{r}(t))) \times \exp^{-\int_0^t \tau(\mathscr{F}(\mathbf{r}(t')))dt'} dt \quad (1)$$

where :

- $\mathbf{C}$ is the final pixel color.
- $\mathbf{r}(t)$ is the cast ray of length D parametrized by the distance to the front intersection with the volume.
- $\mathscr{F}$ is a reconstruction filter to be convolved with the discrete dataset in order to retrieve a continuous scalar field.
- scalar values s are mapped to chromaticity vectors $\mathbf{c}(s)$ and to extinction coefficient $\tau(s)$, thus defining the transfer function.

Even though it has been proven that more complex filters provide better volume reconstruction [5], due to their computational expense, they are inadequate for real-time reconstruction. On the other hand, the trilinear interpolation filter is widely used in interactive volume rendering techniques as it is currently the only filtering kernel fully supported in hardware by standard graphic cards.

We consider the case of regular grids with a trilinear filter. Thus, the scalar value at any position in the volume is determined by uniquely combining the values of the 8 voxels whose centers are the closest. More generally, the grid induced by the centers of the voxels will be called the dual grid, and will hence define dual voxels. For a ray segment $r(t)$ included in a dual voxel, it can be easily proved that the trilinear reconstruction of its scalar values comes down to a cubic polynomial parametrized by t. This property has been previously used in isosurface rendering [13].

We observe two interesting mathematical properties in the case of a trilinearly reconstructed volume :

- the global scalar signal of a ray traversing a volume holds the property of $\mathscr{C}^0$ continuity, the first derivative differing at the transition points between two dual voxels.
- the scalar signal relative to a segment lying in a dual voxel is a $\mathscr{C}^\infty$ continuous cubic curve. The derivative of a cubic polynomial being a quadratic polynomial and the latter admitting at most two real roots, the cubic curve has at most two turning points.

As no assumptions are made on the volume dataset and on the transfer function, solving equation 1 analytically is impossible. We thus have to resort to numerical integration techniques in order to find an approximation of the solution. We now review how the volume rendering integral has previously been computed, focusing on the quality of the scalar approximation with respect to a trilinear filter.

3.2 Numerical Integration

Let $\mathbf{r}(t)$ be a cast ray from the viewer's position whose length is D, and which has been sampled at each intersection with the borders of the crossed dual voxels. We futhermore sample the ray at the midpoints of each successive pair of samples that have been previously defined. n is the total number of discrete samples and t_i denotes the distance from the i^{th} sample to the ray origin. Such a sampling scheme will allow the comparison between a well defined function (a cubic polynomial) with different numerical approximations, which accounts for the ideal sampling case. Notice that the case of constant length sampling will be studied in Section 5. Furthermore, in order to estimate the quality of the approximations, we will consider the numerical error term ξ between an ideal signal and its relative approximation as being the integral of the absolute value of their differences, which we compute using a 0^{th} order integration scheme (for its ease of implementation) at a very high sampling rate (for high precision). The green areas in Figures 1, 2, 4, 5 depict the signed differences between the ideal signals and the corresponding approximations, possibly magnified if so described in the legends.

3.2.1 Zero Order Integration

The most straightforward way of evaluating the volume rendering integral is by decomposing it into a Riemann sum and by applying for each successive pair of samples the rectangle rule. Each slab $(\mathbf{r}(t_i), \mathbf{r}(t_{i+1}))$ of length $t_{i+1} - t_i$ is considered as being either of scalar value $\mathscr{F}(\mathbf{r}(t_i))$ (left Riemann approximation), of scalar value $\mathscr{F}(\mathbf{r}(t_{i+1}))$ (right Riemann approximation) or of scalar value $\mathscr{F}\left(\frac{\mathbf{r}(t_i)+\mathbf{r}(t_{i+1})}{2}\right)$ (midpoint approximation).

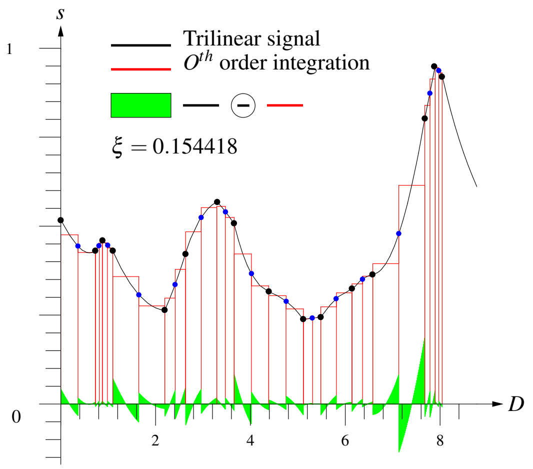


Figure 1: Trilinearly interpolated scalar signal (in black) along a cast ray approximated by the midpoint rule (in red).

Such a process is known as a zero-order hold reconstruction, the approximated scalar density being constant within a ray segment. Thus, the color and opacity values are also constant for a ray segment and no further integration is needed relatively to the transfer function. As shown in Figure 1, the reconstructed signal is not even $\mathscr{C}^0$ continuous, and the approximation error tends to be significant. In the field of volume visualization, this leads to important visual artefacts where the discontinuities are clearly visible between two consecutive samples.

3.2.2 First Order Integration

In order to capture the high frequencies of the transfer function and thus improve the overall rendering quality, pre-integration based volume rendering methods have been proposed [3,18]. The numerical integration performed on the continuous scalar field corresponds to the trapezoid rule, the approximated scalar field within a ray segment being the linear interpolation between its two endpoints. As the resulting scalar field is no longer constant and has to be composed with the transfer function, further integration is performed relatively to the chromaticity vector and the extinction coefficient.

Consequently, the sampling distance must be taken into account as it affects the computation of the color and opacity values, and thus a 3D table indexed by (s_f, s_b, l) is needed to store the precomputed integrals (s_f being the front sample, s_b the back sample and l the length of a ray segment).

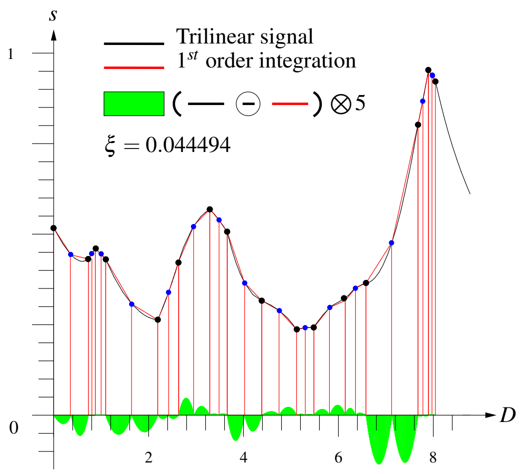


Figure 2: Trilinearly interpolated scalar signal (in black) along a cast ray approximated by the trapezoid rule (in red).

Figure 2 illustrates a 1st order approximation of a cast ray where the scalar field has been trilinearly interpolated. Even though better convergence is achieved in comparison to a 0th order approximation (see the differences between the error terms), the reconstructed signal does not have $\mathscr{C}^1$ continuity inside a dual voxel (between two black points). However, the global approximation is $\mathscr{C}^0$ continuous, which is the reason behind the capture of the transfer function's high frequencies. Notice how within a ray segment the error term grows proportionally to the distance between the current position on the ray and the nearest sample.

3.3 Convergence Analysis

In the field of numerical integration, a function f is integrated by estimating piecewisely the swept area of successive curves, the curves being built from a finite number of discrete samples of f, the latter determining the order of the integration (the higher, the better). An approximation of the integrated function f is then obtained by summing up the swept areas. In general, the goal of numerical integration techniques is the direct estimation of the area swept by a curve using a definite formula, the equation of the curve being usually not needed. However, when dealing with the volume rendering integral, the transfer function is further composed with the scalar field. As the variables t and t' over which we integrate determine the scalar field (see Equation 1), the chromaticity vector and the extinction coefficient cannot be factored out of the integrals, implying further integration relatively to the transfer function.

Thus, when precomputing the rendering integrals as in Section 3.2.2, two distinct numerical integrations must be performed, the first for the scalar field, the second for the transfer function which operates on the approximated scalar field. Consequently, what accounts the most when pre-integrating the volume rendering integral is the shape of the approximated scalar field as opposed to its swept area. The interpolation scheme used for approximating the trilinear signal will determine the correctness of the captured scalar high frequencies, and thus the correctness of the captured high frequencies in the transfer function. If the scalar interpolation scheme used for pre-integrating the volume rendering integral converges poorly towards the ideal trilinear signal when used piecewisely along a ray, this leads to visual artefacts as well as omission of important features of the volume.

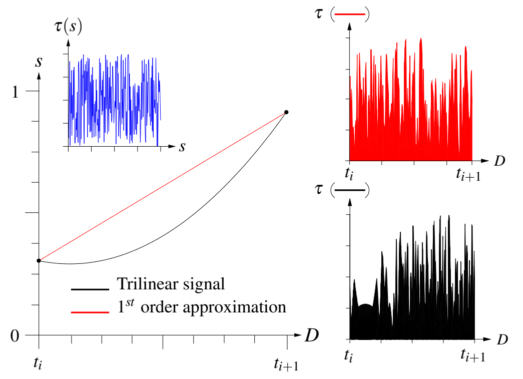


Figure 3: Comparison between a trilinearly interpolated signal (black curve) and its 1st order approximation (red segment) after composition with the extinction component $\tau(s)$ of a random transfer function (blue diagram on top left). The two resulting curves are depicted on the right with their respective colors.

Figure 3 outlines the latter. This figure shows an example of a cast ray segment bounded by only two discrete samples. The trilinear signal is approximated by a 1st order interpolation. We compare both curves after convoluting them with the extinction component of a transfer function having high frequencies. As one can see, the two resulting curves show great differences. The resulting error in terms of visualization would be further increased because of self attenuation within the segment and by the fact that a pixel color is defined by the combination of a triplet of primary colors whose values are computed using such an approximative interpolation scheme. Even though standard pre-integrated volume rendering claims to capture the high frequencies of the transfer function, the approximated pixel colors may greatly differ from what would have ideally been integrated. Indeed, in the case of high frequencies present in the continuous scalar field, if the sampling rate along the rays is not high enough, wrong high frequencies of the transfer function would be captured. This would yield to misleading visual artefacts.

We show in the rest of this paper that a higher order integration scheme can greatly improve the approximation of the volume rendering integral without the need of further resampling the cast rays. For the purpose of direct volume rendering, we also detail how to build a higher order pre-integration table.

4 Second Order Integration

As mentioned in Section 3.1, the trilinear reconstruction of a ray $r(t)$ traversing a volume possesses interesting properties that are omitted by the previously described techniques. In order to converge faster towards the accurate signal, a pre-integrated table should take these properties into account.

Despite the fact that we have analyzed the scalar signal along cast rays for many volumes whose scalar field has been generated semi-randomnly (in order to obtain high frequencies in the continuous scalar field), we have not encountered the case where a ray segment included in a dual voxel has a scalar field with two turning points. These experiments have led us to the conclusion that such situations seem extremely rare. In fact, in the worst case, only one derivative's sign change has been observed. Such a property is inherent to a 2nd order Lagrange polynomial, which is a polynomial of degree 2 defined by three equispaced points, the resulting curve passing through each of these points. Due to its polynomial nature, the shape of the interpolation over a single interval is that of a $\mathscr{C}^{\inf}$ continuous curve, and when used piecewisely on numerous samples, $\mathscr{C}^0$ continuity is achieved at the joint points as the end point of an interval is the start point of its successor.

In the field of numerical integration, this corresponds to a 2nd order Newton-Cotes integration (also known as Simpson's rule) where the area under the curve is computed in $O(1)$. However, when applied to volume rendering techniques, scalar interpolation takes precedence over area computation as we have seen in Section 3.2.2, and we will thus focus on the scalar curve obtained from the 2nd order Lagrange polynomial.

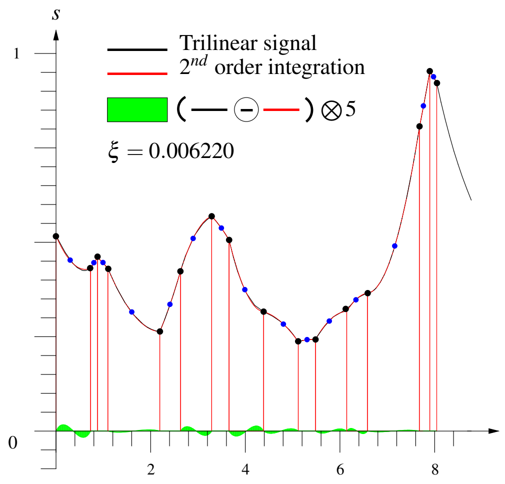


Figure 4: Trilinearly interpolated scalar signal (in black) along a cast ray approximated by Simpson's rule (in red).

As we have chosen to sample the rays at each intersection with the crossed dual voxels' borders and by futhermore considering the midpoints of each previous pair of sample, we will approximate the trilinear interpolation of a segment bounded by a dual voxel with a 2nd order Lagrange polynomial. Figure 4 illustrates this piecewise interpolation and compares it to the trilinear interpolation of the volume. We can observe that in comparison to a 1st order approach, the approximation error is significantly reduced as the original signal is more closely matched. The main reason behind the quality of the approximation is that inside a dual voxel, the trilinear interpolation is analytically defined as a cubic polynomial and thus is smooth and well behaved. As noted previously, one turning point at most has been observed for these bounded segments despite the presence of high frequencies in the generated datasets, which is a property of 2nd order polynomials. Faster convergence may therefore be achieved without the need of strong ray resampling.

In order to benefit from a 2nd order integration scheme, we precompute the volume rendering integral where the scalar field is defined as a 2nd order Lagrange polynomial. Three equispaced scalar samples (namely s_f, s_m and s_b) must be considered (respectively holding for the front, the middle and the back sample). Since we take self attenuation into account, the length l of the ray segment over which we integrate must be considered. This leads conceptually to a 4D table, as opposed to a 3D table in the case of a 1st order integration. Equation 2 details how color and opacity values are computed.

$$\begin{aligned}
\mathbf{c_2}(s_f,s_m,s_b,l) = \; & \int_0^1 \mathbf{c}(P(t))\tau(P(t))l \\
& \times \exp^{-\int_0^t \tau(P(t'))ldt'} dt \\
\tau_2(s_f,s_m,s_b,l) = \; & 1-\exp^{-\int_0^t \tau(P(t))ldt}
\end{aligned} \tag{2}$$

$$\text{where} \left\{ \begin{array}{ccl} A & = & 2s_f - 4s_m + 2s_b \\ B & = & -3s_f + 4s_m - s_b \\ C & = & s_f \\ P(t) & = & At^2 + Bt + C \end{array} \right.$$

Thus, the volume rendering integral (equation 1) is approximated by the following formula where the summation is stepped by 2 since we now consider triads of sample values instead of pairs of values :

$$\begin{aligned}
C \approx \; & \sum_{\substack{i=0 \\ i+2}}^{n-3} \mathbf{c_2}\left(s_f(i), s_m(i), s_b(i), l(i)\right) \\
& \times \prod_{\substack{j=0 \\ j+2}}^{i-1} \left(1 - \tau_2\left(s_f(j), s_m(j), s_b(j), l(j)\right)\right)
\end{aligned} \tag{3}$$

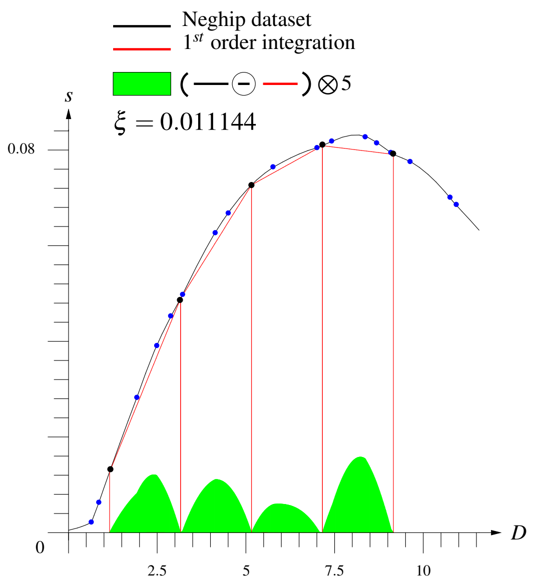
$$\text{where} \left\{ \begin{array}{ccl} s_f(k) & = & \mathscr{F}\left(r\left(t_k\right)\right) \\ s_m(k) & = & \mathscr{F}\left(r\left(t_{k+1}\right)\right) \\ s_b(k) & = & \mathscr{F}\left(r\left(t_{k+2}\right)\right) \\ l(k) & = & t_{k+2} - t_k \end{array} \right.$$

However, the adaptive sampling scheme that has been used in the previous examples (see Figures 1, 2 and 4) is more complex than most of the sampling algorithms used in popular direct volume rendering methods. In fact, constant length sampling of the rays (as with raycasting or as commonly admitted with slicing) is more efficient with respect to performance. Also, it is simpler to implement. Consequently, we have to analyze how a 2nd order pre-integrated table affects the quality of the scalar approximation when samples along the rays are considered equispaced. This is the purpose of the next Section.

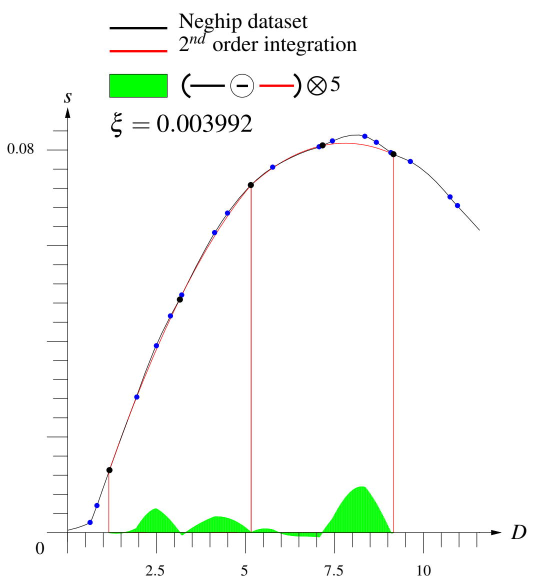
5 Case of Constant Length Ray Sampling

In comparison to the previous sampling approach of the cast rays, constant length sampling implies that the trilinear interpolation between two endpoints of a ray segment is no longer defined analytically (*i.e.* a cubic polynomial). First derivatives may vary abruptly within a segment as we do not know where the samples will be collected. Theoretically, this could deteriorate the quality of the 2nd order integration if a point lying on a given ray segment intersects a dual voxel's border and has its left first trilinear derivative differing in sign with its right first trilinear derivative.

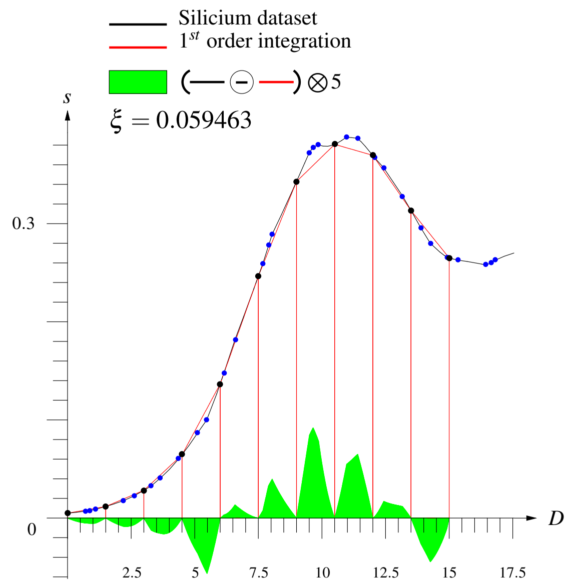
Since it is impossible in this case to analytically estimate the approximation error of a 2nd order integration of the volume rendering integral (no assumptions being made on the datasets and on the transfer function), we have conducted a serie of tests on common datasets in order to determine the quality of the interpolation. We show two of the worst cases that we have encountered in practice, comparing the trilinear interpolation to respectively its 1st and 2nd order approximation.



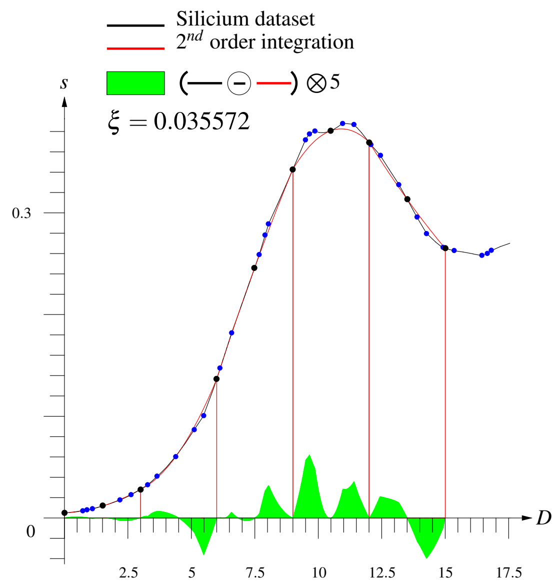
(a) 1^{st} order approximation



(b) 2^{nd} order approximation



(c) 1^{st} order approximation



(d) 2^{nd} order approximation

Figure 5: Trilinear signals (in black) obtained from cast rays in the Neghip (top row) and the Silicium (bottom row) datasets. The red curves illustrate the 1^{st} order approximations (left column) and the 2^{nd} order approximations (right columns) with constant sampling length of the ray.

Figures 5(a) and 5(b) show a cast ray in the neghip dataset, black dots representing the chosen samples along the ray, blue dots representing every intersection with the borders of the crossed dual voxels. We can see that the 2^{nd} order approximation 5(b) better converges towards the trilinear signal than the 1^{st} order approximation 5(a) does, which is reflected by their respective error terms. Figures 5(c) and 5(d) illustrate the same situation for the Silicium dataset, with the 2^{nd} order approximation 5(d) also minimizing the error term in comparison to the 1^{st} order approximation 5(c).

Even though better convergence is achieved when the cast rays are sampled at the intersections with the dual voxels [22], we have verified in practice (numerically as well as in terms of visualization) that even with equispaced samples, a 2^{nd} order integration of the scalar field still improves the quality of the rendered volume in comparison to a 1^{st} order integration. We explain the better results by the fact that a trilinearly interpolated scalar field along a ray has the shape of a curve where the magnitude of its curvature is generally non zero. Intuitively, such a curve is better fitted piecewisely with 2^{nd} order polynomials (which can have non zero curvature) than with segments (which corresponds to a 1^{st} order approach, and where the magnitude of the curvature is always zero within the segment). Consequently, even in the case of constant length ray sampling, the discrepancy with the ideal signal is minimized when using a 2^{nd} order approach in comparison to a 1^{st} order approach, thus yielding better visual results when the same number of samples are considered. Regarding pre-integration, the precomputed table (2) may be reduced from 4D to 3D in this case, the distance between the end points of a segment being constant. This in turn is a desirable feature for direct volume rendering as 3D textures are handled natively by standard consumer graphics.

6 Implementation

We now detail the implementation of our volume renderer which benefits from a 2nd order integration scheme. The cast rays being sampled regularly, a 3D 2nd order pre-integration table is enough for storing the color and opacity values. Both renderer and 3D table computation make use of the GPU programmability features and are intended for Shader Model 3.

6.1 Rendering Algorithm

Our volume rendering implementation is inspired by a 3D texture based raycasting approach with the sampling length d being fixed and common to all the cast rays.

In a first pass, given a perspective matrix transform, both front and back facing quadrilaterals of the bounding box are projected to two distinct 32 bit floating point auxiliary buffers where a fragment shader writes the interpolated 3D texture coordinates. We make use of the OpenGL extension GL_ARB_draw_buffers for rendering multiple targets, the back-face register allowing to determine wether a fragment has been generated by a front face or by a back face.

In a second pass, the two previous auxiliary buffers are bound to two distinct texture units, and a quadrilateral filling the viewport is projected. For each pixel of the viewport, the main fragment shader fetches in the back facing texture if valid texture coordinates are present. If not, the fragment is discarded, otherwise the front facing texture coordinates are retrieved in order to compute the step vector along the ray according to the predefined step length d as well as the total number of iterations to proceed.
The raycasting phase then begins :
Each slab of thickness $2d$ is defined by three successive samples (s_f, s_m, s_b), the next slab being indexed by (s'_f, s'_m, s'_b) with the correlation $s'_f = s_b$. After retrieving the samples (s_f, s_m, s_b) scalar values in the 3D volume, the color and opacity values are given by a 3D texture dependant fetch in the 3D pre-integration table. A front to back compositing algorithm blends the successive slabs to finally obtain the pixel color.

6.2 3D Tables Computation on GPU

We consider transfer functions defined on an 8 bit scalar domain. As the pre-integration table will be of size 256^3, an orthogonal projection matrix as well as a 256^2 viewport are set.

In order to compute every 2D slice along the depth of the pre-integration table, we render 256 times a quadrilateral parallel to the projection plane which fills the viewport. For every pixel, the fragment shader relates x screen coordinate to the front sample s_f, y screen coordinate to the middle sample s_m as the back sample s_b is given by a uniform variable.

The 2nd order polynomial coefficients A, B and C are thus determined as in (2) before entering the main loop. The shader then operates a fixed number of iterations over the polynomial's interval $[0,1]$ to compute the successive scalar values using Horner's rule for better numerical accuracy and better perfomance. At each step, color and opacity values of the current scalar are fetched in the transfer function, which are then composited using a back to front algorithm with the previous accumulated color and opacity. Once the loop has ended, the final color and opacity of the pixel is written to the frame buffer. Notice that the number of iterations used in the shader determines the quality of the pre-integration and is user defined.

After the viewport has been filled, we perform a copy to texture operation on the texture unit that is storing the 3D pre-integration table for the current depth index. Notice also that since we accumulate values into a floating point register, the composition of the color and opacity values is achieved in 32 bit floating point accuracy, only the final value being clamped to the framebuffer's precision.

7 Results

All results have been reported on an Athlon64 3200+ processor coupled with a *n*VIDIA GeForce 7800GTX graphics card. A 2nd order 3D table with 8 bits encoded color and opacity components occupies 64 MBytes in GPU memory. As our method requires two times less access to the pre-integration table in comparison to a 1st order approach, two times more compositing operations are needed in color space in the latter case. In order to avoid artefacts due to color quantization errors and thus enforce the comparison between the two different methods, 16 bits floating point 1st order pre-integrated tables have been used. The 2nd order pre-integrated tables have been computed using 256 iterations in order to satisfy the Nyquist sampling theorem. However, even though a low number of iterations has led to visible artefacts, we have observed in practice that independently of the dataset and of the transfer function, 64 iterations were enough to achieve the same visual results as an upsampled pre-integrated table.

Figure 7 shows the neghip dataset rendered twice with a 1st order approach (Figures 7(a) and 7(b)) and once with a 2nd order approach (Figure 7(c)). In both cases, a sampling length of 2 voxels' width has been used for the cast rays on Figures 7(a) and 7(c). We can clearly observe that more rendering artefacts are present in the case of a 1st order approach (blurry regions, deformed isosurfaces), most of the latter being successfully removed by the 2nd order pre-integration method. In order to confirm the correctness of the convergence, we futhermore compare our results to a 1st order approach where the sampling rate of the cast rays has been doubled (see Figure 7(b)).

In a same fashion, we show results for two other datasets; namely the fuel 8 and the bucky ball 9. First, considering the same sampling length of the rays, 1st order pre-integration based images (respectively Figures 8(a) and 9(a)) are compared evenly to 2nd order pre-integration based images (respectively Figures 8(c) and 9(c)). Upsampling of the cast rays for the 1st order approach is futhermore given in Figures 8(b) and 9(b) to estimate the quality of our method's convergence.

Iterations	8	16	32
Computation times	28/0.53	50/0.93	90/1.52
Iterations	64	128	256
Computation times	165/2.74	328/5.14	663/10.88

Figure 6: Pre-integration table computation times for the CPU and the GPU approaches (in seconds).

Figure 6 presents the computation times for our pre-integrated table using different number of iterations. A comparison is made between the CPU approach (left of the cells) and its GPU adaptation (right of the cells) for increasing numbers of iterations. The GPU approach performs up to 120 times faster than the CPU one and thus allows fast transfer function modifications when using a sensible number of steps. We believe that in the near future, standard graphics hardware will make possible interactive transfer function manipulation.

Another result worth noticing is that, given the same sampling rate of the rays, second order interpolation is consistently faster than first order interpolation despite redundant 3D dependent texture accesses (which are known to produce bad cache coherency). We explain this performance benefit by the fact that first order rendering requires a loop over N iterations in the fragment shader while second order rendering requires a loop over N/2 iterations, *i.e.*, half as much. Indeed, branching is still a penalizing instruction on standard consumer graphics. In addition, only half of the compositing operations are needed, thus lowering the computational cost. Notice that independantly of the order used for pre-integration, the number of texture accesses to the volume data is always N, the back sample being cached for further use in the next iteration.

8 Conclusions and Future Works

We have presented a novel direct volume rendering method based on the pre-integration of the transfer function using a 2^{nd} order integration of the scalar field.

We have shown that independently of the sampling scheme employed for the cast rays (constant or adaptive sampling), this higher order integration scheme significantly minimizes the discrepancy between a trilinearly interpolated volume and its piecewise approximation. Scalar high frequencies in the volume are more accurately captured in comparison to standard pre-integration based methods, which is reflected on the correctness of the captured transfer function high frequencies and thus on the volumes' visualization. For scientific data or critical datasets where high frequencies are often encountered, less ray sampling is needed to capture the finest details, thus implying improvements with respect to both performance and quality. However, a direct consequence of this 2^{nd} order integration approach is the memory occupation when pre-integrated. A 4D table must be used when adaptively sampling the rays whereas a 3D table when the rays are sampled regularly. Consequently, we plan on reducing the memory occupation of the pre-integrated tables as well as speeding up the precomputation process. Finally, we believe that a volume renderer which smartly samples the cast rays while based on a 2^{nd} order pre-integration table could greatly improve the quality of the rendered images, the challenges being to maintain at least interactive frame rates and to fit the 4D pre-integration table into GPU memory.

From a conceptual point of view, we have shown that depending on the reconstruction filter, pre-integration techniques should consider the scalar interpolation scheme used for precomputation in order to reflect as correctly as possible the behaviour of the ideal scalar field. Thus, we believe that better reconstruction filters (in comparison to a trilinear filter) should be studied in correlation with the pre-integration of the volume rendering integral as the quality of the visualization could greatly benefit from it.

References

[1] B. Cabral, N. Cam, and J. Foran. Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In *VVS '94: Proceedings of the 1994 symposium on Volume visualization*, pages 91–98, New York, NY, USA, 1994. ACM Press.

[2] T. J. Cullip and U. Neumann. Accelerating volume reconstruction with 3d texture hardware. Technical report, Chapel Hill, NC, USA, 1994.

[3] K. Engel, M. Kraus, and T. Ertl. High-quality pre-integrated volume rendering using hardware-accelerated pixel shading. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, pages 9–16. ACM Press, 2001.

[4] S. Guthe, S. Roettger, A. Schieber, W. Strasser, and T. Ertl. High-quality unstructured volume rendering on the pc platform. In *HWWS '02: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 119–125, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association.

[5] M. Hadwiger, I. Viola, T. Theußl, and H. Hauser. Fast and flexible high-quality texture filtering with tiled high-resolution filters. In *Vision, Modeling and Visualization 2002*, pages 155–162. Akademische Verlagsgesellschaft Aka GmbH, Berlin, Nov. 2002.

[6] T. Klein, M. Strengert, S. Stegmaier, and T. Ertl. Exploiting Frame-to-Frame Coherence for Accelerating High-Quality Volume Raycasting on Graphics Hardware. In C. Silva and E. Gröller and H. Rushmeier, editor, *Procceedings of IEEE Visualization '05*, pages 223–230. IEEE, 2005.

[7] J. Kniss, S. Premoze, M. Ikits, A. Lefohn, C. Hansen, and E. Praun. Gaussian transfer functions for multi-field volume visualization. In *Proceedings of IEEE Visualization 2003*, pages 497–504, 2003.

[8] J. Kruger and R. Westermann. Acceleration techniques for GPU-based volume rendering. In *VIS '03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, page 38, Washington, DC, USA, 2003. IEEE Computer Society.

[9] E. Lum, B. Wilson, and K.-L. Ma. High-quality lighting and efficient pre-integration for volume rendering. The Joint Eurographics-IEEE TVCG Symposium on Visualization 2004, 2004.

[10] N. Max, P. Hanrahan, and R. Crawfis. Area and volume coherence for efficient visualization of 3d scalar functions. In *VVS '90: Proceedings of the 1990 workshop on Volume visualization*, pages 27–33, New York, NY, USA, 1990. ACM.

[11] B. Nelson. Ray-tracing polymorphic multidomain spectral/hp elements for isosurface rendering. *IEEE Transactions on Visualization and Computer Graphics*, 12(1):114–125, 2006. Member-Robert M. Kirby.

[12] K. Novins and J. Arvo. Controlled precision volume integration. In *VVS '92: Proceedings of the 1992 workshop on Volume visualization*, pages 83–89, New York, NY, USA, 1992. ACM Press.

[13] S. Parker, P. Shirley, Y. Livnat, C. Hansen, and P.-P. Sloan. Interactive ray tracing for isosurface rendering. In *VIS '98: Proceedings of the conference on Visualization '98*, pages 233–238, Los Alamitos, CA, USA, 1998. IEEE Computer Society Press.

[14] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 1992.

[15] C. Rossl, F. Zeilfelder, G. Nurnberger, and H.-P. Seidel. Reconstruction of volume data with quadratic super splines. *IEEE Transactions on Visualization and Computer Graphics*, 10(4):397–409, 2004.

[16] S. Röttger 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, Piscataway, NJ, USA, 2002. IEEE Press.

[17] S. Röttger, 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, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[18] S. Röttger, 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, Los Alamitos, CA, USA, 2000. IEEE Computer Society Press.

[19] O. Sadowsky, J. D. Cohen, and R. H. Taylor. Rendering tetrahedral meshes with higher-order attenuation functions for digital radiograph reconstruction. In *IEEE Visualization*, page 39. IEEE Computer Society, 2005.

[20] S. Stegmaier, M. Strengert, T. Klein, and T. Ertl. A Simple and Flexible Volume Rendering Framework for Graphics-Hardware–based Raycasting. In *Proceedings of the International Workshop on Volume Graphics '05*, pages 187–195, 2005.

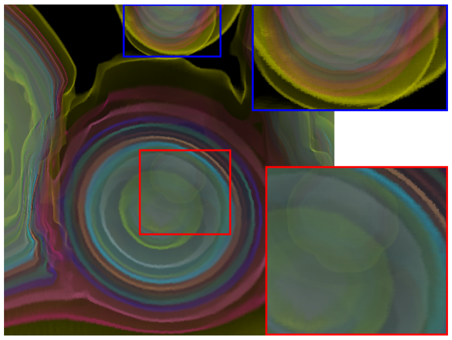
[21] C. M. Stein, B. G. Becker, and N. L. Max. Sorting and hardware assisted rendering for volume visualization. In *VVS '94: Proceedings of the 1994 symposium on Volume visualization*, pages 83–89, New York, NY, USA, 1994. ACM.

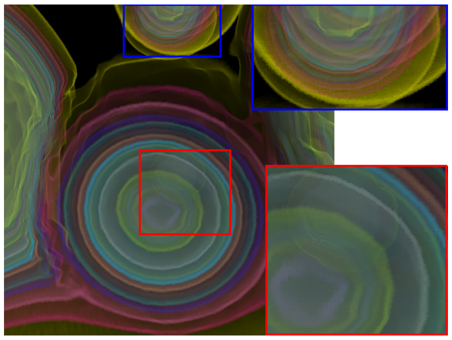
[22] M. Weiler, M. Kraus, M. Merz, and T. Ertl. Hardware-Based Ray Casting for Tetrahedral Meshes. In *Procceedings of IEEE Visualization '03*, pages 333–340. IEEE, 2003.

[23] D. F. Wiley, H. R. Childs, B. F. Gregorski, B. Hamann, and K. I. Joy. Contouring curved quadratic elements. In *VISSYM '03: Proceedings of the symposium on Data visualisation 2003*, pages 167–176, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[24] D. F. Wiley, H. R. Childs, B. Hamann, and K. I. Joy. Ray casting curved-quadratic elements. In *VisSym 2004, Symposium on Visualization, Konstanz, Germany, May 19-21, 2004*.

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


(a) 1st order, 2 voxels' width sampling length, 54.0 FPS


(b) 1st order, 1 voxel's width sampling length, 29.4 FPS

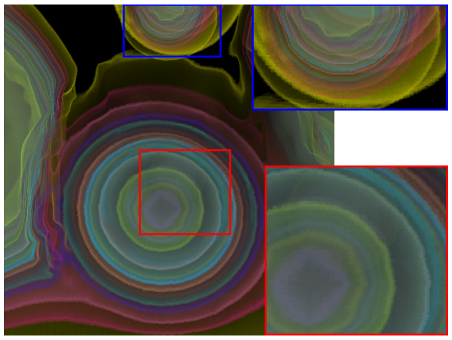
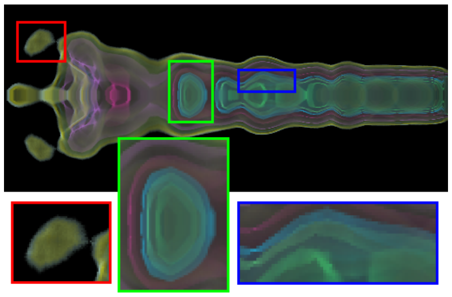
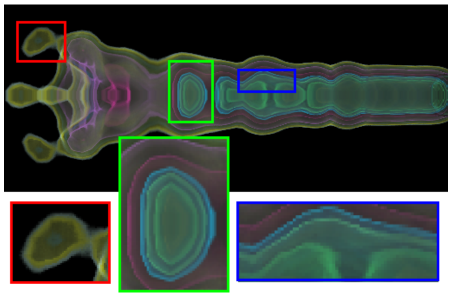

(c) 2nd order, 2 voxels' width sampling length, 84.2 FPS

Figure 7: Neghip 64^3 dataset rendered at different integration orders and different sampling rates


(a) 1st order, 2 voxels' width sampling length, 44.7 FPS


(b) 1st order, 1 voxel's width sampling length, 23.8 FPS

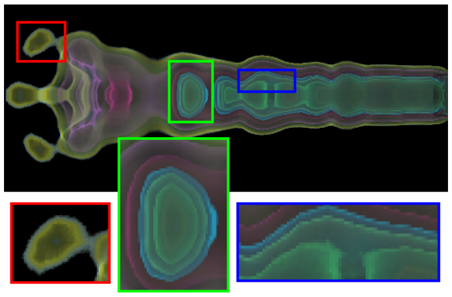
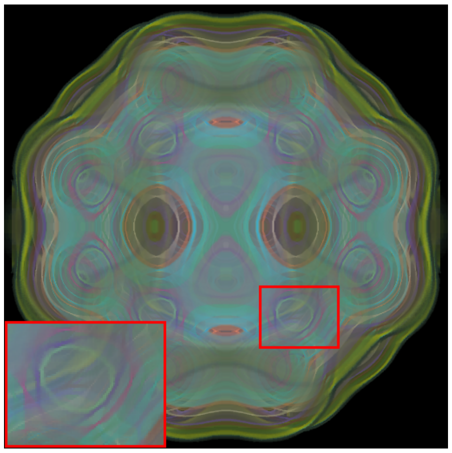
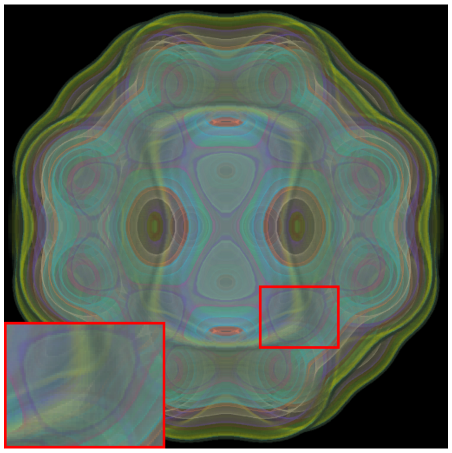

(c) 2nd order, 2 voxels' width sampling length, 75.0 FPS

Figure 8: Fuel 64^3 dataset rendered at different integration orders and different sampling rates


(a) 1st order, 4 voxels' width sampling length, 80.0 FPS


(b) 1st order, 1 voxel's width sampling length, 24.6 FPS

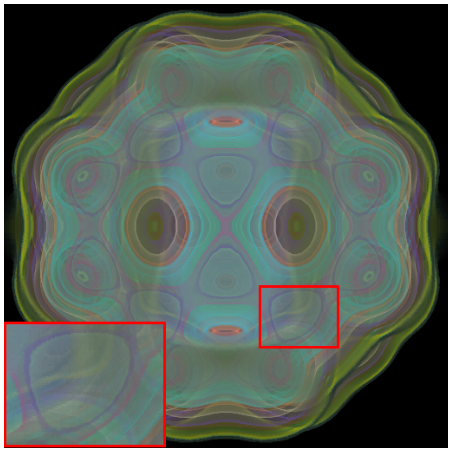

(c) 2nd order, 4 voxels' width sampling length, 111.3 FPS

Figure 9: Bucky Ball 64^3 dataset rendered at different integration orders and different sampling rates