Volume Raycasting Sampling Revisited

P. Steneteg, D. Jönsson, M. Falk and I. Hotz

Department of Science and Technology, Linköping University, Sweden
E-mail: peter.steneteg@liu.se

***Abstract*—We investigate the effects of practical sample placement strategies when solving the volume rendering integral for interactive volume raycasting with fixed step lengths for each ray. Different sample placements have been used in previous work but they have not been compared with respect to their correctness or visual quality. In this work, the different sampling strategies are presented visually and practical implementation details are provided using algorithmic descriptions of each strategy. A thorough analysis based on comparisons with analytic solutions and real-world data shows that visual artifacts, especially at volume borders, can appear if samples are not placed correctly. Our analysis and comparison results in a sample placement strategy that easily can be integrated into existing implementations, has no impact on performance, and decreases visual artifacts of the rendered image compared to other fixed step size sample strategies.**

***Index Terms*—Volume Rendering, Raytracing, Rasterization**

I. Introduction

Volume raycasting has developed into one of the most common techniques for visualizing three dimensional scalar fields with a large variety of applications ranging from medical imaging data to engineering simulations. However implementing an accurate volume raycaster is a tedious task requiring the consideration of many intricate details to achieve correct results. The difficulty of getting all aspects correct were demonstrated by Etiene et al. [1]. They showed that widely available volume raycasting implementations have flaws producing visual artifacts. In this paper, we focus on visual artifacts related to different ray sampling schemes.

Interactive volume raycasting implementations typically use a discretized solution of the volume rendering equation where the ray is partitioned into fixed length segments [2]. This solution is fast and accurate but artifacts can occur at volume boundaries or across pixels depending on the sampling pattern. Due to the importance of interactivity it is often not possible to use sampling methods that avoid these artifacts, i.e. Monte Carlo sampling. Practical solutions that reduce or remove such sampling artifacts without increasing the computational costs or complexity of the implementation can thus be very useful. With this goal in mind we have analyzed typically used ray-sampling schemes and introduce a novel strategy that removes artifacts at borders without having a negative impact on performance. In addition, it can be easily integrated into existing implementations. Our main contributions are:

- A thorough quantitative and qualitative analysis of the effects of different ray sample placement schemes in volume raycasting
- A study and comparison of existing ray sample placement implementations
- A practical solution, including source code, that can be easily integrated into existing implementations

LEVIA'19: Leipzig Symposium on Visualization in Applications 2019

II. Related Work

Volume raycasting, first presented in 1988 [3]–[5], usually relies on the emission-absorption model or density-emitter model [5]. The volume rendering integral (VTI) proposed by Max [2] describes such an model for each viewing ray and is discretized using a Riemann sum. Here, the work by Kruger et al. [6] and the book Real-Time Volume Graphics by Engel et al. [7] among others provides valuable practical examples on how to make efficient implementations of the discretized VRI. Engel et al. [7], and other typical implementations of the VRI, use a sampling rate derived from the resolution of the volume to solve the Riemann sum in the discretized VRI. We perform an analysis of different sampling schemes including the one mentioned above, and their effect when used on multiple co-located rays.

The correctness of the implementation of VRI has been analyzed in several works. Williams and Max [8] analyzed the VRI with respect to piecewise-linear transfer function and derived analytic solutions for such a case. Lee and Newman [9] proposed a new opacity correction method for improved accuracy when oversampling the volume. Kratz et al. [10] presented an adaptive screen-space sampling technique to reduce errors. Pre-integrated transfer function were introduced by Engel et al. [11] which can significantly decrease errors, especially for thin structures. Etiene et al. [1] analyzed the discretization error of the VRI and used it to verify volume rendering implementations by comparing their convergence with the expected convergence. Our analysis and proposed solution is orthogonal to these works on correctness since we are focusing on determining the step length and the sample positions.

Adaptive step lengths can also be used to improve quality and performance. However, they require additional information and data structures to determine step sizes. Examples include the early work of Levoy [12] who used an octree data structure. These additional data structures are not always feasible to compute in interactive scenarios, for example when dealing with time-varying data. Importance-based sampling strategies [13] can also be employed but they are less suited to hardware-accelerated methods due to their irregular behavior across neighboring rays, which prevents caches from being

used efficiently. Our analysis is therefore performed on uniform grids without adaptive structures.

III. Problem Statement

In this work, we assume the low albedo emission-absorption model proposed by Max [2]:

$$I(D) = I_0 \mathrm{e}^{-\int_0^D \tau(t)\,\mathrm{d}t} + \int_0^D g(s) \mathrm{e}^{-\int_s^D \tau(t)\,\mathrm{d}t}\,\mathrm{d}s, \qquad (1)$$

with $I(D)$ being the accumulated light intensity along a ray of length D. The first term describes the light intensity in the background, I_0, attenuated by the transmittance through the medium with the extinction coefficient $\tau(t)$ along the ray. The second term describes the accumulated emission, $g(s)$, attenuated by the transparency along the ray. Again, following Max [2], the volume integral is approximated numerically by a Riemann sum over n equidistant segments of length Δx yielding

$$I(D) \approx I_0 \prod_{i=1}^{n} t_i + \sum_{i=1}^{n} g_i \prod_{j=i+1}^{n} t_j. \qquad (2)$$

Here, the transparency of the ith segment is $t_i = \mathrm{e}^{-\tau(i\Delta x)\Delta x}$ and the emission is $g_i = g(i\Delta x)\Delta x$. While this solution provides a starting ground for implementing a volume renderer, it does not deal with determining n or where to start and end sampling along a ray.

A practical and common solution to determining n is to allow the user to specify a sampling rate, which in turn is used to determine the step size depending on the resolution of the volume. That is, a sampling rate of two applied to a volume of two voxels would require four samples. However, naively applying this to an implementation will cause issues at borders and visually disturbing effects due to aligned samples over pixels.

In this work, we analyze where to best place the first and last sample and how big the step size Δx should be, depending on view direction and volume resolution.

IV. Sampling Position Approaches

There are a range of options to consider when placing samples using fixed step lengths along a ray entering and exiting a volume. We will describe these options together with practical details supplied in the form of OpenGL shading language code. We assume that the mid-point rule is used in the Riemann sum, which is the case for most implementations of volume raycasting.

As a first attempt, following Fig. 2 by Kruger et al. [6], we calculate the first sample point in the volume as seen from the eye, described in Listing 1. This will create a sampling pattern across pixels that form concentric lines as seen in Fig. 1b. We denote this as "Eye Align".

We can see that the Riemann sum mid-point estimate will over- or underestimate contributions at both the first and last sample points as indicated by the overshooting red sections of the samples along the ray in the figure. To remedy this, we can align the sample points, shifted half a step, with the entry of the volume, described in Listing 2. Similar to the depictions in the Real-Time Volume Graphics book by Engel et al. [7]. We call this method "Entry Align".

```
// ray in texture coordinates
vec3 ray = exitPoint - entryPoint;
float samples = length(ray * dim) * samplingRate;
float tEnd = length(ray);
float tIncr = tEnd / samples;
float numSteps = length(entryPoint - eyePos) /
  ↪ tIncr;
// first point along the ray after the entryPoint
float tStart = (1-fract(numSteps)) * tIncr;
```

Listing 1: **Eye align** sampling. Inputs are the *entryPoint* and *exitPoint*, given in texture coordinates, the *sampleRate*, and the volume dimensions *dim*. The outputs are *tStart*, *tIncr*, and *tEnd*.

```
vec3 ray = exitPoint - entryPoint;
float samples = length(ray * dim) * samplingRate;
float tEnd = length(ray);
float tIncr = tEnd / samples;
float tStart = 0.5 * tIncr;
```

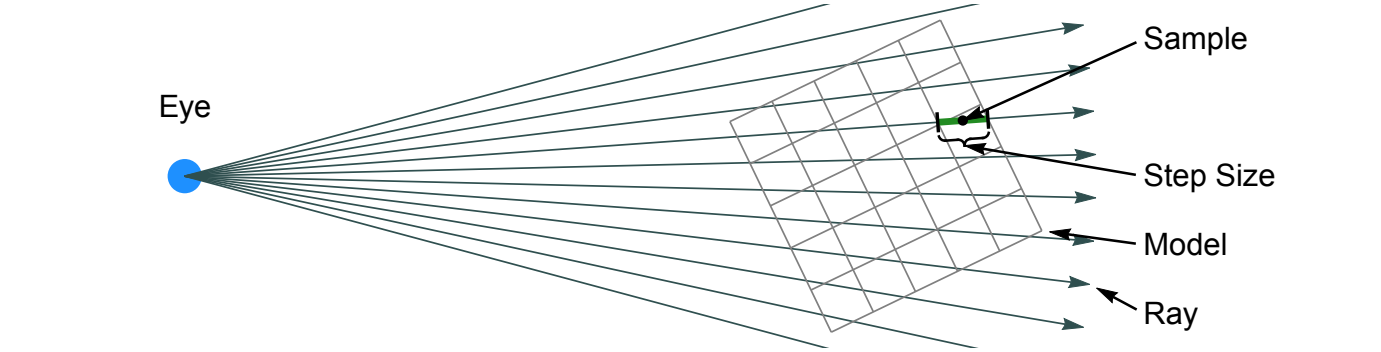
Listing 2: **Entry Align** sampling. Inputs are the *entryPoint* and *exitPoint*, given in texture coordinates, the *sampleRate*, and the volume dimensions *dim*. The outputs are *tStart*, *tIncr*, and *tEnd*.

Now, the first samples will be aligned with the bounding geometry as seen in Fig. 1c. Still, the last part of the volume, i.e. on the back side, will be under- or overestimated depending on how close the last sample is to the back side. Ensuring that both the first and the last samples are not under- or overestimated requires that the first sample is taken half a step from the entry point and the last sample is taken half a step from the exit point, as in Fig. 1d. Subsequently, the step size needs to be adjusted for each ray, while still being fixed along the ray, as seen in Listing 3. We denote this as "Entry Exit Align".

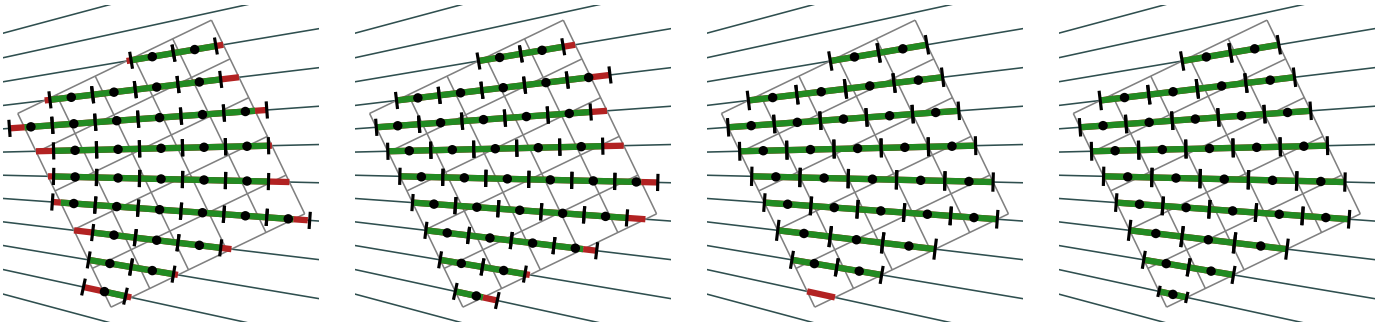
```
vec3 ray = exitPoint - entryPoint;
float samples = floor(length(ray * dim) *
  ↪ samplingRate);
float tEnd = length(ray);
float tIncr = tEnd / samples;
float tStart = 0.5 * tIncr;
```

Listing 3: **Entry Exit Align** sampling. Inputs are the *entryPoint* and *exitPoint*, given in texture coordinates, the *sampleRate*, and the volume dimensions *dim*. The outputs are *tStart*, *tIncr*, and *tEnd*.

The approaches above will exclude contributions at the corners where the length of the ray is smaller than the step size. These rays can be included by adjusting the step size to the length of those rays and placing a sample at their mid-points, as in Listing 4. This is illustrated in Fig. 1e when



(a) *Overview of the raycasting scene*



(b) *Eye Align* **(c)** *Entry Align* **(d)** *Entry Exit Align* **(e)** *Entry Exit Align Corner*

Fig. 1. Graphical depiction of the four discussed sampling patterns. Black points represent the sampling positions, and the ticks the beginning and end of the sampling step attributed to each sampling point. The green lines represent a valid sampling distance whereas the red lines represent either a distance inside the volume that is not sampled or a distance outside the volume that is sampled but should not have been considered.

combined with the "Entry Exit Align" sampling method. We call this "Entry Exit Align Corner".

We note that implementations using the pre-integrated volume rendering technique reduced to two dimensions, i.e. scalar values at start and end of each ray segment [11], require all rays to use the same step length and therefore must choose one of the sampling strategies which under- or overestimates the integral. This issue is also discussed in the work by Stegmaier et al. [14], who chose to neglect the last sample and accept the error it introduced.

```
vec3 ray = exitPoint - entryPoint;
float samples = ceil(samplingRate*length(ray * (dim
  ↪  - vec3(1))));
float tEnd = length(ray);
float tIncr = tEnd / samples;
float tStart = 0.5 * tIncr;
```

Listing 4: **Entry Exit Align Corner** sampling. Inputs are the *entryPoint* and *exitPoint*, given in texture coordinates, the *sampleRate*, and the volume dimensions *dim*. The outputs are *tStart*, *tIncr*, and *tEnd*.

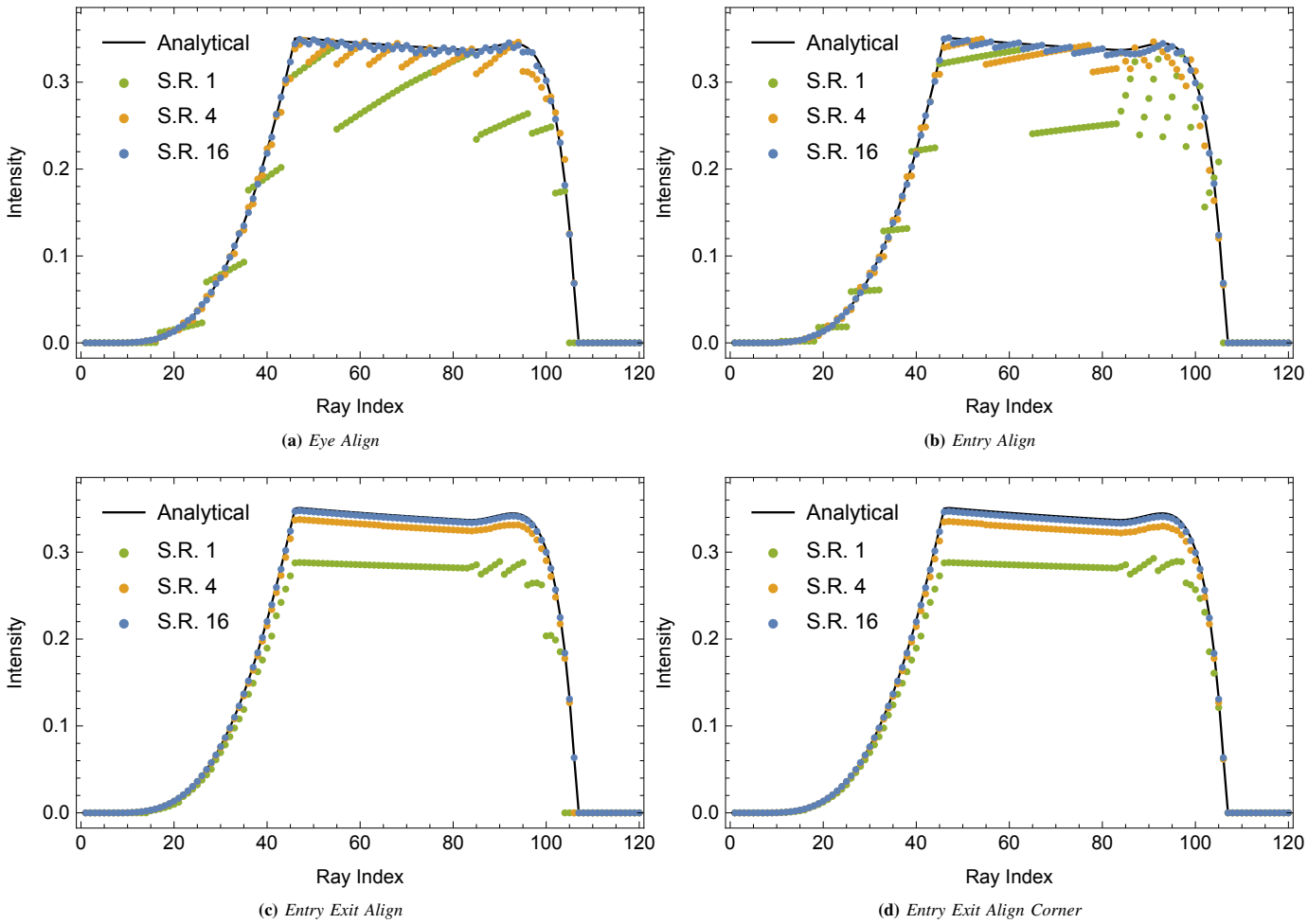
V. Analysis

We analyzed the impact of the different sample placement strategies with respect to the total ray intensity error by comparing their results to the analytic solutions as well as using real data. We also present visual comparisons to see how the patterns across pixels affect the perceived quality. Furthermore, we study the convergence rate with respect to step size for the different methods. We were not able to detect any significant difference with respect to rendering performance for the four different sample patterns. This is to be expected since the total number of sampled points is very similar for the different methods.

A. Numerical analysis

Without loss of generality, our analytic solution is evaluated in 2D using the setup seen in Fig. 1a. Here, the eye is placed at (-9,0) units and the data model is 3 units square centered at the origin and rotated by an angle $\alpha = \pi/7$ with respect to the x-axis. The data model is then filled using the analytic function illustrated in Fig. 3 and discretized into a 5×5 grid. A set of 120 rays with a field of view of 30° and equal angular spacing is then created. The Volume rendering integral is solved for each ray either directly, using a closed-form solution of the continuous equation (1), or using a set of sample points generated by one of the four described methods using linear interpolation.

We parameterize the volume rendering equation as follows. A zero background intensity, $I_0 = 0$, is assumed. The ray r is parameterized using $\chi_r(\lambda) : \mathcal{R} \rightarrow \mathcal{R}^2$ where the ray enters the volume at $\lambda = 0$ and exits at $\lambda = D$. The data model is sampled using $s(\chi)$. For evaluation of the different sampling methods we define the transfer function as $g(s) = s$ and $\tau(s) = s/2$. Hence the intensity for ray r is in the



(a) *Eye Align*

(b) *Entry Align*

(c) *Entry Exit Align*

(d) *Entry Exit Align Corner*

Fig. 2. Calculated intensities for the different sampling patterns using the linear data set with a set of 120 rays. The solution of the continuous Equation (3) for each ray is drawn as a black line, the solutions of the discrete Equation (5) for a sampling rate (S.R.) of 1, 4, and 16 are shown as dotted data series. The effects of over and under estimating the first and last samples points can be seen in (a) and (b) whereas the effect of missing corner samples can be seen as underestimated intensities at around ray 105 in (c).

continuous case described by:

$$I_r = \int_0^D g(s(\chi_r(\lambda)))\mathrm{e}^{-\int_0^\lambda \tau(s(\chi_r(\lambda')))\,\mathrm{d}\lambda'}\,\mathrm{d}\lambda. \quad (3)$$

$$= \int_0^D f(\chi_r(\lambda))\mathrm{e}^{-\int_0^\lambda f(\chi_r(\lambda'))/2\,\mathrm{d}\lambda'}\,\mathrm{d}\lambda. \quad (4)$$

where f is the sampled data. And for the discrete case

$$I_r = \sum_{i=1}^{n} g(s(\chi_{r,i}))\Delta x \prod_{j=0}^{i-1} \mathrm{e}^{-\tau(s(\chi_{r,j}))\Delta x} \quad (5)$$

$$= \sum_{i=1}^{n} f(\chi_{r,i})\Delta x \prod_{j=0}^{i-1} \mathrm{e}^{-f(\chi_{r,j})\Delta x/2} \quad (6)$$

where $\chi_{r,i}$ is the ith sample position along ray r and Δx is the step length.

The results of applying this solution are shown for the linear dataset $f(x,y) = \frac{x+1.5}{3}$ in Fig. 3. Here, the effects of over or under estimating the first and last samples points is clearly visible in Fig. 2a. Fig. 2b has less pronounced errors since the strategy is not over- or underestimating the first sample point along the ray, only the last point. The behavior in Fig. 2c is clearly more stable as it neither over- or underestimates any of

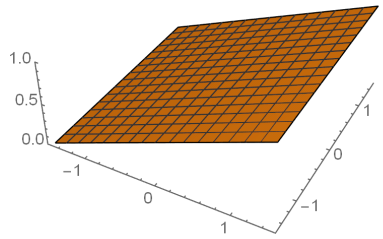
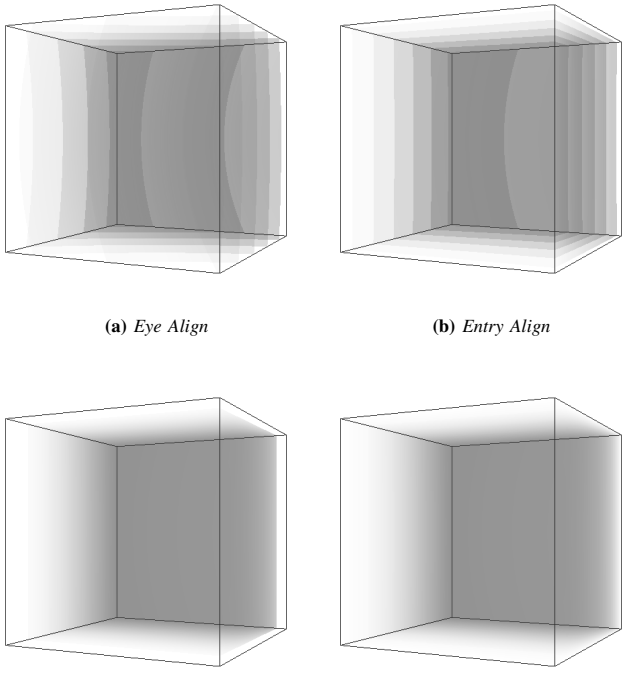


Fig. 3. Analytic function used for testing: $f(x,y) = \frac{x+1.5}{3}$



(a) *Eye Align* **(b)** *Entry Align*

(c) *Entry Exit Align* **(d)** *Entry Exit Align Corner*

Fig. 4. Results of raycasting (500×500 rays) using the different sample patterns on the linear function test dataset. The effects of the sampling patterns can clearly be seen. In (a) and (b) the effect of over- or underestimating the first and/or last samples points is visible as a stepping artifact originating from the corners. In (c) the missing corner samples are evident by the gap between the bounding box and the content. (d) has none of the above mentioned artifacts

the sample points. However, around ray 105, we can see that the intensities drop to zero before the analytical solution. This is a result of missing corner samples, as is illustrated in the bottom part of Fig. 1d, where the ray length is shorter than the step size. Fig. 2d no longer shows this sharp drop-off since the corners are accurately sampled.

B. Visual analysis

While the analytic solution is good for studying numerical effects it can only be done for trivial functions and does not provide insight into how the sampling strategies are interpreted visually. We therefore implemented the different sampling patterns within the Inviwo visualization framework [15]. Fig. 4 shows the results of using this implementation on the same data and setup as in the analytic solution, albeit extruded into three dimensions (5^3 voxels) and with a denser set of rays. The intensities in the middle horizontal pixel row corresponds to the intensities in Fig. 2 (darker means higher in this case). This low resolution grid clearly shows the type of artifacts one can expect for each sampling strategy. In Fig. 4a and Fig. 4b the effects of over- or underestimating the first and/or last samples points are visible as steps in the intensities. Fig. 4c no longer has any of the visible steps in the intensities as should be expected since the ray no longer over- or undershoots the

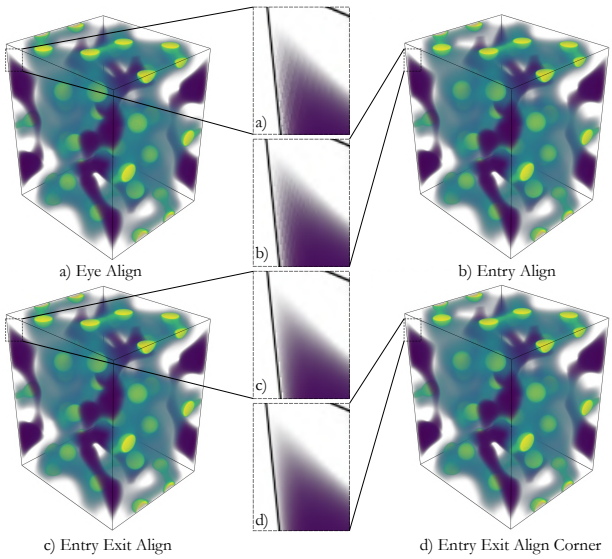


Fig. 5. Renderings of charged density field of a γ-Boron unit cell using the different sample placement strategies. (a) Eye aligned sampling causes ringing artifacts. (b) Entry aligned sampling exhibits less but still noticeable ringing artifacts. (c) Values along border are not taken into account. (d) Border displays correctly.

volume. However, a small inset between the data and the bounding box can be seen corresponding to missing corner sample points in Fig. 1d. The gap is easily noticeable when compared to Fig. 4d where the gap is gone.

To illustrate the characteristics of the sampling strategies on real-world data we chose to visualize a charged density field of a γ-Boron unit cell with a resolution of 150^3 voxels. The Boron data is distributed along with the Inviwo framework [15] and contains information at borders where the sampling strategies differ in appearance. These differences are highlighted in Fig. 5. It can be seen that the artifacts can appear also in real-world data.

VI. Discussion

Our analysis only considers the basic cases where the entire volume is rendered using a step size which is fixed per ray. Yet the analysis should still hold true with the introduction of cut planes in the volume since it will only introduce new entry and exit points, which we still can use as input for the sample placement strategies. If more elaborate sample techniques with adaptive step lengths are applied, the same overall observations about the over- and underestimation of the first and last sample points and missing corner sample hold. But a more extensive analysis would have to be carried out for the adaptive step length method in question.

It is also evident that the errors presented in the sampling pattern in Fig. 1 and the corresponding errors in intensities in Fig. 2 is a result of over- and underestimation of the samples at the border voxels in the volume. For low resolution volumes, or volumes with a large fraction of the significant information in that area, these samples may represent a significant part to the total ray intensity and the errors may be of significance.

But for high resolution volumes the fraction of voxels at the border decreases quickly and, hence, also the observed errors. Likewise, if all of the interesting information is in the interior of the volume the errors at the border will not be visible.

VII. Conclusion

Implementing a volume raycasting algorithm can be tedious and there are many aspects that need to be considered in order to obtain correct results. We have presented, analyzed, and discussed four different sampling strategies that can be used for fixed-length sampling. The results showed that, for small volumes with data at the borders, the errors due to sample placement can be significant. However, for real world use cases with higher resolution volumes the effect is often marginal. Certain situations, such as when zooming in on borders, can still make artifacts appear. Thus, since there is no performance impact and no other downside to using a sample strategy which minimizes the errors, we recommend using the suggested "Entry Exit Align Corner" method from Listing 4.

Acknowledgment

This work was supported through grants from the Excellence Center at Linköping and Lund in Information Technology (ELLIIT), the Swedish e-Science Research Centre (SeRC), and the DigiPat project by VINNOVA grant 2014-04257. The presented concepts have been developed and evaluated in the Inviwo framework [15] (www.inviwo.org).

References

[1] T. Etiene, D. Jönsson, T. Ropinski, C. Scheidegger, J. Comba, L. G. Nonato, R. M. Kirby, A. Ynnerman, and C. T. Silvia, "Verifying volume rendering using discretization error analysis," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 1, pp. 140–154, 2014.

[2] N. Max, "Optical models for direct volume rendering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 1, no. 2, pp. 99–108, 1995.

[3] R. A. Drebin, L. Carpenter, and P. Hanrahan, "Volume rendering," *Computer Graphics (Proceedings of SIGGRAPH 1988)*, vol. 22, no. 4, pp. 65–74, 1988.

[4] M. Levoy, "Display of surfaces from volume data," *IEEE Computer Graphics and Applications*, vol. 8, no. 3, pp. 29–37, 1988.

[5] P. Sabella, "A rendering algorithm for visualizing 3D scalar fields," *Computer Graphics (Proceedings of SIGGRAPH 1988)*, vol. 22, no. 4, pp. 51–58, 1988.

[6] J. Kruger and R. Westermann, "Acceleration techniques for gpu-based volume rendering," in *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, ser. VIS '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 38–. [Online]. Available: http://dl.acm.org/citation.cfm?id=1081432.1081482

[7] K. Engel, M. Hadwiger, J. Kniss, C. Rezk-Salama, and D. Weiskopf, *Real-time volume graphics*. CRC Press, 2006.

[8] P. L. Williams and N. Max, "A volume density optical model," in *Workshop on Volume Visualization*, 1992, pp. 61–68.

[9] J. K. Lee and T. S. Newman, "New method for opacity correction in oversampled volume ray casting," *Journal of WSCG*, 2007.

[10] A. Kratz, J. Reininghaus, M. Hadwiger, and I. Hotz, "Adaptive screen-space sampling for volume ray-casting," Zuse Institute Berlin, Germany, Tech. Rep., 2011. [Online]. Available: https://opus4.kobv.de/opus4-zib/frontdoor/index/index/docId/1244

[11] K. Engel, M. Kraus, and T. Ertl, "High-quality pre-integrated volume rendering using hardware-accelerated pixel shading," in *ACM SIGGRAPH/Eurographics Workshop on Graphics Hardware*, 2001, pp. 9–16.

[12] M. Levoy, "Efficient ray tracing of volume data," *ACM Transactions on Graphics*, vol. 9, no. 3, pp. 245–261, 1990.

[13] J. Danskin and P. Hanrahan, "Fast algorithms for volume ray tracing," in *Workshop on Volume Visualization*, 1992, pp. 91–98.

[14] S. Stegmaier, M. Strengert, T. Klein, and T. Ertl, "A simple and flexible volume rendering framework for graphics-hardware–based raycasting," in *Eurographics/IEEE VGTC Workshop on Volume Graphics*, 2005, pp. 187–195.

[15] D. Jönsson, P. Steneteg, E. Sundén, R. Englund, S. Kottravel, M. Falk, A. Ynnerman, I. Hotz, and T. Ropinski, "Inviwo - a visualization system with usage abstraction levels," *IEEE Transactions on Visualization and Computer Graphics*, 2019.