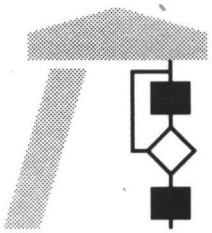
Interner Bericht

Interleaved Sampling

A. Keller, W. Heidrich

308/01



FACHBEREICH
INFORMATIK

UNIVERSITÄT
KAISERSLAUTERN

Postfach 3049 · D-67653 Kaiserslautern

Interleaved Sampling

A. Keller, W. Heidrich

308/01

Universität Kaiserslautern
AG Numerische Algorithmen
Postfach 30 49
67653 Kaiserslautern
Germany

März 2001

Herausgeber: AG Numerische Algorithmen
Leiter: Professor Dr. S. Heinrich

Interleaved Sampling

Alexander Keller[1], Wolfgang Heidrich[2]

[1)]Universität Kaiserslautern
[2)]The University of British Columbia

Abstract

The sampling of functions is one of the most fundamental tasks in computer graphics, and occurs in a variety of different forms. The known sampling methods can roughly be grouped in two categories. Sampling on regular grids is simple and efficient, and the algorithms are often easy to built into graphics hardware. On the down side, regular sampling is prone to aliasing artifacts that are expensive to overcome. Monte Carlo methods, on the other hand, mask the aliasing artifacts by noise. However due to the lack of coherence, these methods are more expensive and not well suited for hardware implementations.

In this paper, we introduce a novel sampling scheme where samples from several regular grids are a combined into a single irregular sampling pattern. The relative positions of the regular grids are themselves determined by Monte Carlo methods. This generalization obtained by interleaving yields significantly improved quality compared to traditional approaches while at the same time preserving much of the advantageous coherency of regular sampling.

We demonstrate the quality of the new sampling scheme with a number of applications ranging from supersampling over motion blur simulation to volume rendering. Due to the coherence in the interleaved samples, the method is optimally suited for implementations in graphics hardware.

CR Categories: I.3.1 [Computer Graphics]: Hardware Architecture—Graphics processors; I.3.2 [Computer Graphics]: Picture/Image Generation—Antialiasing; I.3.2 [Computer Graphics]: Picture/Image Generation—Display algorithms; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture; I.4.1 [Image Processing and Computer Vision]: Digitization and Image Capture—Sampling

Keywords: Monte Carlo Techniques, Graphics Hardware, Frame Buffer Algorithms, Antialiasing, Volume Rendering

1 Introduction

Sampling of functions is one of the most fundamental tasks in computer graphics. It has to be performed for applications as diverse as anti-aliased scan-conversion of scenes, illumination from area light sources, cameras with depth-of-field effects, motion blur, and volume rendering. The existing sampling schemes for these applications can be grouped into regular and irregular methods.

Sampling on regular grids, on the one hand, is efficient and simple, and also well suited for hardware implementations. As an example, consider the rasterization of geometric primitives to generate digital images. The samples used for this image formation process are typically arranged on a regular grid, which allows for the use of efficient algorithms for scan-conversion of lines and polygons. These algorithms can easily be implemented in graphics hardware.

Unfortunately, the signals that need to be sampled in computer graphics are typically not band limited and can often contain arbitrarily large frequencies. In the rasterization example, this is typically the case at the edges of an object. Therefore some amount of aliasing is unavoidable, and regular sampling grids emphasize these sampling artifacts.

Monte Carlo methods use randomized sample positions instead of regular sampling patterns. This irregular approach masks the aliasing artifacts by noise that is more visually pleasing. The most powerful schemes include stratified sampling [25], and Poisson disk sampling [23]. Since these points are more uniformly distributed than pure random points, they guarantee better convergence.

On the down hand, per pixel irregular sampling does not exhibit the kind of coherence known from regular sampling. Consequently, these methods are more expensive and not well suited for hardware implementations.

In this paper we introduce a new sampling scheme that we call *interleaved sampling*. In this scheme, we interleave samples taken from a number of independent regular sub-grids, and merge them into a single sample pattern. The relative positions of the sub-grids to each other are determined by an arbitrary irregular method in a preprocessing step. Within each of the regular sub-grids we can then use the efficient algorithms developed for sampling on regular grids.

In the above example of rasterization, this means that we could use any existing scanline algorithm to generate the samples for each of the sub-grids. In this case, each sub-grid can be interpreted as a low resolution image of the scene, and the pixels of all these images are interleaved to generate a high-quality, high-resolution image.

We demonstrate the quality of this new sampling scheme with a number of applications ranging from supersampling over motion blur to volume rendering. At the same time, we argue that due to the coherence in the interleaved samples, the method is optimally suited for implementations in graphics hardware. It is possible both to use the new sampling scheme with existing hardware, and to modify the hardware to provide some more direct support. Often only minimal changes to existing hardware algorithms such as the Accumulation Buffer or multisampling are required to exploit the benefits of the new sampling scheme.

The remainder of this paper is organized as follows: in Section 2 we discuss the relevant previous work in this area. In Section 3 we describe the theory behind the new sampling scheme and analyze it in terms of convergence and spectral properties, and in Section 4 we

describe applications of the scheme to a number of rendering algorithms, including supersampling, multisampling, motion blur, and volume rendering. In all applications we discuss both implementations on current hardware and how the method could be further improved by direct hardware support in the future.

2 Previous Work

The placement of samples for integration of functions has received a lot of attention in the computer graphics literature (for a survey see [11]). Besides improving the convergence by stratification as analyzed in [33, 8, 7, 25] for the setting of image synthesis, the spectral properties of sampling patterns have been investigated profoundly [5, 23, 20]. There has also been a lot of work on adaptive sampling techniques in computer graphics (e.g. [24, 19, 23]). However since our goal is to develop a sampling scheme that is well suited for hardware implementations, we do not consider locally adaptive techniques in this paper.

On the hardware side, our method is closely related to the multisampling method for anti-aliasing. In the initial description of the method by Akeley [1], each pixel has a number of sub-pixels that are placed on on a regular grid. A more recent formulation [32] allows for sampling patterns that are not necessarily on a regular grid. Each of these samples has an individual color and depth. During rendering of primitives, the color of each pixel in the image is updated every time one of its sub-pixels changes. Interleaved sampling can be implemented in a similar way, but in contrast to the original multisampling method the patterns within each pixel will not be a regular grid, and the sampling patterns of neighboring pixels will not be the same (although the patterns repeat after a number of pixels). Nonetheless we will be able to use efficient scan conversion algorithms for image synthesis, because each of the sub-pixels lies on a *globally defined* regular grid.

Also relevant to our method are the algorithms developed for the Accumulation Buffer [12]. One application of this buffer is again anti-aliasing of scenes. Like in our work, and in contrast to multisampling, the sub-pixel positions are no longer located on a regular grid. Unlike our method, however, the sampling patterns are identical for each pixel, which makes it more prone to aliasing artifacts.

Interleaved sampling can also be used for the other applications that have been described for the Accumulation Buffer, in particular motion blur, soft shadows based on the method by Brotman and Badler [2], limited depth-of-field [12, 21, 15], and Instant Radiosity [18]. In addition to these applications, we will also consider interleaved sampling for texture based volume rendering along the lines of Cabral et al. [3] and Westermann and Ertl [35].

3 Theory

In this section we briefly review how interleaved sampling is related to the method of dependent tests and review the design and desirable properties of sampling patterns suited for interleaved sampling.

3.1 The Method of Dependent Tests

The method of dependent tests is a generalization of the Monte Carlo estimation of functionals to the estimation of functions [34, 10]. Instead of estimating only a single functional, i.e. a real number defined by an integral

$$\int_{I^s} f(x)dx \approx \frac{1}{N}\sum_{i=0}^{N-1} f(x_i),$$

where the x_i are N independent realizations of one uniform random variable ξ on the s-dimensional unit cube $I^s := [0,1)^s$, we estimate a whole function g by

$$g(y) := \int_{I^s} f(x,y)dx \approx \frac{1}{N}\sum_{i=0}^{N-1} f(x_i,y). \qquad (1)$$

The method earns its name, since for any value of y, the *same* N samples $P_N := \{x_0,\ldots,x_{N-1}\}$ are used. This intrinsic coherence has been extensively exploited in computer graphics. We briefly illustrate the principle for the example of the Accumulation Buffer [12]: Instead of using a different sampling pattern for each pixel, as in e.g. software rendering, the same sampling pattern is used for each pixel. The resulting regular grids can be sampled using raster graphics hardware, which allows for very efficient implementations of crucial effects such as anti-aliasing, depth-of-field, motion blur, area light sources and even global illumination simulations [18, 13].

3.2 Interleaving the Dependent Tests

The advantage of being able to exploit scanline coherence using the method of dependent tests is paid for by a higher risk of local aliasing.

However, this disadvantage caused by repeating the same sampling pattern for every pixel can be eliminated by substructuring. This core idea is illustrated by Figure 1: Instead of repeating the same sampling pattern for every pixel, four basis patterns can be repeated on a group of pixels, in this case a 2×2 block. Thus the high coherence is preserved and can still be exploited by raster graphics hardware, but the high risk of local aliasing is reduced drastically by using a different sampling pattern in adjacent pixels.

For the scope of the paper we use a substructuring of the unit square I^2 in $s = 2$ dimensions by M arbitrary subdomains $D_j \subseteq I^2, 1 \le j \le M$ that do not necessarily need to be disjoint. Then the method of dependent tests from Equation (1) generalizes to

$$\begin{aligned} g_j(y) &:= \frac{1}{|D_j|}\int_{I^s} f(x,y)\chi_{D_j}(x)dx \\ &\approx \frac{1}{N|D_j|}\sum_{i=0}^{N-1} f(x_i,y)\chi_{D_j}(x_i), \end{aligned}$$

where

$$\chi_{D_j}(x) := \begin{cases} 1 & x \in D_j \\ 0 & \text{else} \end{cases}$$

is the characteristic function of the set D_j.

Going back to the 2-dimensional example in Figure 1b, each of the $M = 4$ domains corresponds to one of the four pixels in the basis pattern, which then is replicated. A specific 2×2 block then is addressed using the variable y. Note that depending on the stratification of the basis pattern even non-integer interleaving ratios like e.g. 1×1.5 are possible.

3.3 Sampling Patterns for Interleaved Sampling

In image synthesis the integrands are both square integrable and Riemann integrable. However no smoothness properties can be guaranteed, and in fact the integrands usually contain discontinuities such as edges of the geometry. As a consequence, sampling with a finite set of points will violate Shannon's sampling theorem. In addition the integrands can be of high dimension, finally ruling out classical tensor product quadrature. While the aliasing potential can be judged from the spectral properties of the samples, the

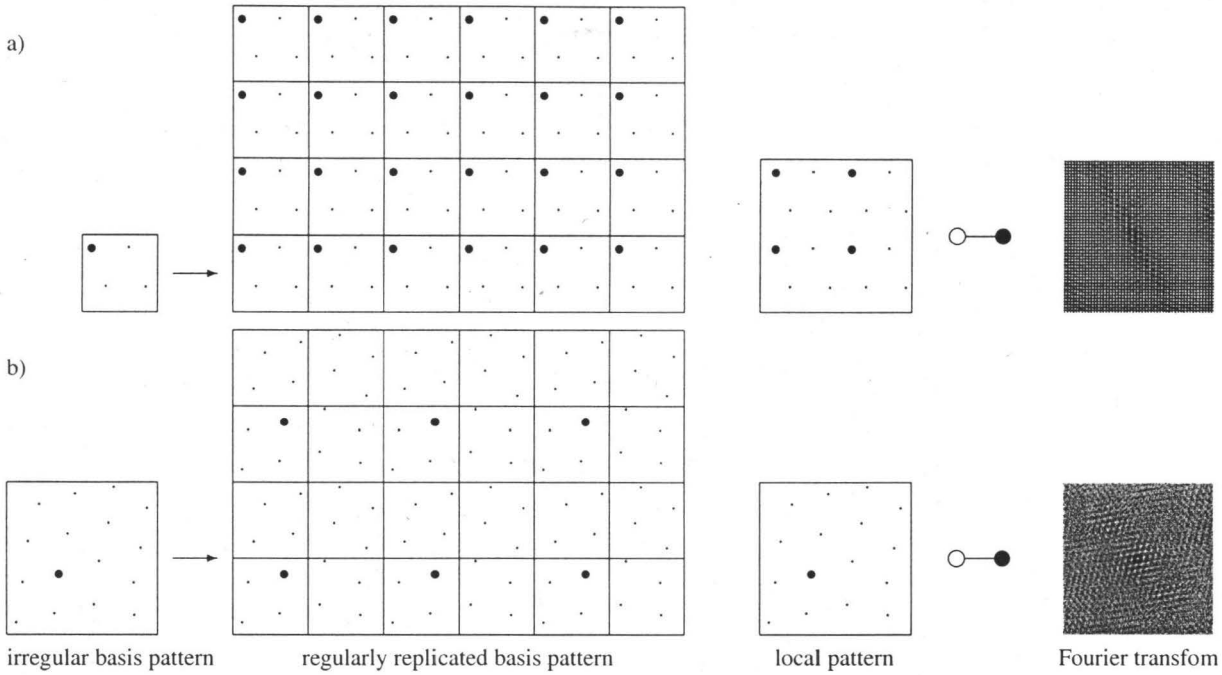


Figure 1: The principle of interleaved sampling for the example of the Accumulation Buffer: In a) the classical method of dependent tests as used in the Accumulation Buffer is illustrated. The same sampling pattern is repeated for all pixels. The Fourier transform of this repeated pattern clearly shows a high potential for aliasing. In b) a 2×2 pixel basis pattern is replicated. Thus the sampling patterns in adjacent pixels are different by interleaved sampling and the Fourier transform reveals a much reduced potential for aliasing. To illustrate the regular grids involved, one sample from the irregular basis pattern is highlighted as well as its corresponding replications.

convergence of integration can be judged from the uniformity of their distribution. We now discuss both issues and show how they combine.

3.3.1 Random Sampling

The square integrability guarantees the convergence of the Monte Carlo method, i.e. random sampling, for integrands of high dimension and/or unknown discontinuities. Due to the unknown structure of the integrand, isotropic random sampling patterns with a minimum distance property like Poisson disk patterns [5, 23, 11] are suited best. Because of the spectral properties of their irregularity, aliasing is mapped to noise, which is much less perceivable to the human observer.

In the appendix, we present a simple algorithm for the generation of sample points with blue noise spectrum that seamlessly replicate (see Figure 1). These are an even better approximation to the receptor array in the human visual system than Poisson disk samples, since both the minimum and maximum nearest neighbor distance are bounded from below and above, respectively.

3.3.2 Low Discrepancy Sampling

The uniformity of a set of sample points can be measured by discrepancy [26]. Due to [16], already the decreasing discrepancy of a sequence of sets of sample points guarantees decreasing integration error for Riemann integrable functions. Therefore, lowering the discrepancy, i.e. increasing the uniformity of the samples, yields faster quadrature rules [25]. Usual approaches include Latin hypercube sampling (also known as N-rooks sampling) and stratification. The concept of (t, m, s)-nets [26] generalizes these techniques.

By choosing a dimension $s \geq 1$ and an integer base $b \geq 2$, we define the elementary interval

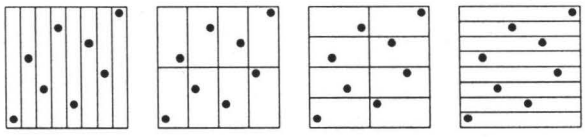
$$E := \prod_{j=1}^{s} \left[\frac{a_j}{b^{l_j}}, \frac{a_j + 1}{b^{l_j}} \right) \subseteq I^s$$

for integers $l_j \geq 0$ and $0 \leq a_j < b^{l_j}$. Consequently its volume is

$$\lambda_s(E) = \frac{1}{b^{\sum_{j=1}^{s} l_j}}.$$

Definition 1 *For two integers $0 \leq t \leq m$, a finite point set of b^m points is a (t, m, s)-net in base b in s dimensions, if every elementary volume of size $\lambda_s(E) = b^{t-m}$ contains exactly b^t points.*

It can be shown, that (t, m, s)-nets obtain the least possible order of discrepancy, i.e. are best uniformly distributed point sets [26]. Note that although sequences of refining regular grids have a decreasing discrepancy they do not obtain the discrepancy of (t, m, s)-nets, i.e. are not of low discrepancy. For the best possible quality parameter $t = 0$, the concepts of Latin hypercube sampling and stratification are united. This is illustrated by depicting all possible arrangements of elementary volumes for a $(0, 3, 2)$-net in base $b = 2$:



Usually such (t, m, s)-net constructions are deterministic and thus points are aligned on a grid structure [26]. As discrepancy is an anisotropic measure, many (t, m, s)-nets that minimize discrepancy do not perform optimal in the isotropic case and expose bad spectral properties.

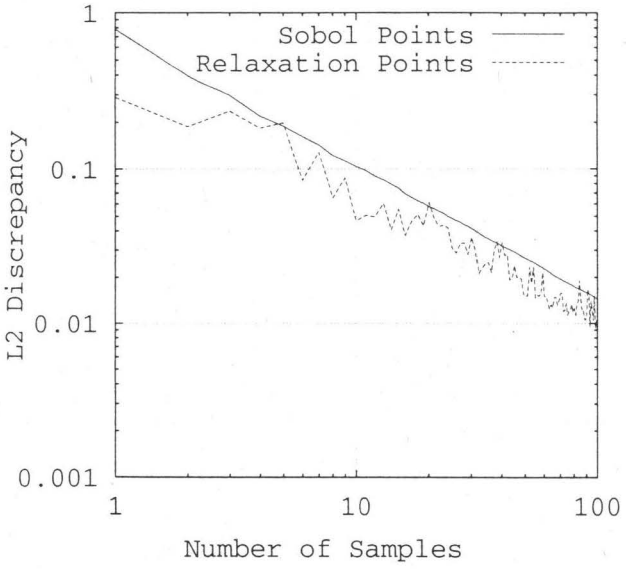


Figure 2: Comparison of the L^2-discrepancy in $s = 2$ dimensions of the low discrepancy Sobol points and the relaxation points generated by the algorithm presented in the appendix ($M = 1000$). Clearly the relaxation points obtain the same or even better discrepancy.

3.3.3 Random Low Discrepancy Sampling

Interleaved sampling is designed as a global scheme, that is, there is no local adaption as already mentioned in the introductory sections. Sampling patterns should replicate seamlessly, which is achieved by periodic boundary conditions, i.e. designing the patterns on the s-dimensional unit torus. Since usually only a small set of samples is required, the samples can be precomputed and tabulated similar to the OpenGL Accumulation Buffer implementation.

The concept of (t, m, s)-nets also applies for randomized point sets [27, 18]. These then lack the grid structure of deterministic constructions and expose much improved spectral properties.

The points obtained by the relaxation method presented in the appendix have the optimal spectral properties of blue noise samples. In addition they obtain the low discrepancy of (t, m, s)-nets as shown in Figure 2, and thus guarantee fast convergence of the integration scheme. In fact there even exist realizations of relaxation points that are $(0, m, 2)$-nets. In practice, the relaxation points are shifted on the torus I^s so that they form a point set suited for interleaved sampling.

4 Applications

Now that we have described the theoretical benefits of interleaved sampling, we consider a number of different application scenarios and describe how interleaved sampling can be applied to these.

4.1 Anti-Aliasing by Supersampling

The first application we will look at is supersample anti-aliasing along the lines of multisampling [1] and the Accumulation Buffer [12].

Existing multisampling implementations can be interpreted as a special case for interleaved sampling. If we consider only the i^{th} sample of every pixel in an image, we see that all these samples are located on a regular grid with a spacing that is identical to the spacing of the pixels (see the emphasized sample in Figure 1a). Therefore, efficient incremental scan-conversion algorithms can be applied on this grid for rasterizing geometric primitives.

With interleaved sampling we stretch out the grid spacing of the samples so that they are no longer identical to the pixel spacing. Therefore, every sample on a particular grid corresponds to a different sub-pixel position in one of the pixels (see the emphasized sample in Figure 1b). Stretching out the grid spacing also stretches out aliasing structures thus very much improving the visual quality as can be seen in Figure 3.

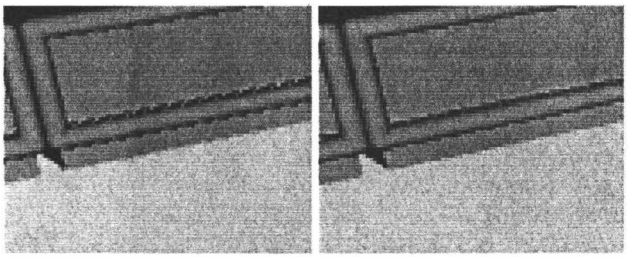


Figure 3: On the left we see the standard Accumulation Buffer approach for anti-aliasing at 4 samples per pixel as depicted in Figure 1a. Using the interleaved approach (see Figure 1b) on the right, the aliasing artifacts are spatially separated yielding a more visually pleasing appearance.

There are three different scenarios for using interleaved sampling in hardware rendering for this application: a direct implementation of a modified multisampling algorithm, adding a fast hardware path for stipple patterns, and finally an implementation on current hardware that does not require additional support.

Modification of Multisample Hardware. The most generally applicable way would be to add direct hardware support for interleaved sampling in a similar fashion multisampling is supported today. The hardware for scan conversion does not need to be modified for this. The only difference to multisampling is that not all the pixels will have the same sub-pixel masks, but the mask is one of a predetermined and implementation-specific number of masks.

The relative positions of the regular sub-grids, that also determine the masks for the individual pixels, would be an implementation-specific arrangement that could be wired into the hardware. These positions can be generated in the design phase of the hardware as discussed in Section 3.3.3.

Efficient Stipple Patterns. OpenGL [31] defines a feature called *stipple patterns*, which allows the programmer to specify a 32×32 mask that can be used to enable or disable the writing to individual pixels in the framebuffer. Using this feature, we can implement interleaved sampling as follows.

If we have a sampling pattern that repeats every n pixels in both x and y, we can generate a mask that only selects the pixels (x, y) with $x \mod n = 0$ and $y \mod n = 0$, and then render the image with the corresponding sampling pattern. We repeat this for all combinations of $x, y \mod n = 0 \ldots n - 1$, resulting in a total of n^2 rendering passes in each of which only N/n^2 pixels are set, where N is the resolution of the full image. This perfectly fits the concept of low discrepancy sampling using $(0, m, 2)$-nets as discussed in Sections 3.3.2 and 3.3.3.

Of course, this only makes sense in applications whose performance is strictly bounded by the rasterization work. Applications that are bounded by geometry transformations would be slowed down by a factor n^2. Examples for applications well suited for

this algorithm include volume rendering (see Section 4.3), and anti-aliasing of complex multipass algorithms (e.g. [14, 13, 29]) when they are implemented on modern multitexture hardware. On these platforms, many multipass algorithms can be reduced to a single pass, or at least very few passes. This reduces the geometry load dramatically because the geometry has to be rendered fewer times, but at the same time every individual pass is more expensive, mostly due to restrictions in the bandwidth to the texture and video RAM.

But even if the performance is bounded by the rasterization cost, this algorithm is not practical on most of the contemporary hardware, because stipple patterns are often not an optimized feature. Rather than terminating masked out fragments early in the graphics pipeline, i.e. during scan conversion, this masking often takes place at the end of the pipeline, after texture mapping. On these platforms, the algorithm just described is no faster than applying *all* sub-pixel masks to *every* pixels, which would of course give a much higher quality, but at a very high cost. Thus, support for early termination of fragments not set in the stipple pattern would be necessary to make this algorithm practical.

Implementation on Current Hardware. A final way to implement interleaved sampling is to individually render lower resolution images corresponding to the sub-grids, and to then interleave the samples obtained this way by hand. The straightforward way of reading back the rendered sub-grids from the framebuffer and performing the interleaving in software is usually not an option because of the the cost of reading and writing the framebuffer.

Another way, however, is either to render the sub-grids directly into a texture (this operation is supported on many contemporary platforms that have unified texture and framebuffer memory), or to copy the framebuffer to texture RAM after rendering (which is less expensive than reading back the framebuffer to main memory, because the data remains on the graphics board). With these sub-grids as textures, we can then render full-screen polygons while the stipple pattern is set in such a way that the correct sub-pixel mask is implemented.

Like the previous algorithm, this method also suffers from the fact that stipple patterns are not optimized on many platforms. However, in this case, there is only a constant overhead for rendering the full screen polygons with the stipple patterns. The cost of this process does not depend on the shading complexity of every sample in the image. Thus, if the application is limited by the rasterization performance, this algorithm has only a constant overhead over supersampling with the same sampling pattern for each pixel. It is therefore a feasible algorithm that is interesting for applications that have high shading costs for each individual sample, such as complex procedural shaders [29], or volume rendering (also see Section 4.3).

4.2 Motion Blur

The anti-aliasing by interleaving images as described in the previous Section can be extended to the simulation of motion blur in an obvious way: Each of the interleaved images is assigned a moment in time. Then the irregular basis sampling pattern is 3-dimensional, where the first two dimensions are used for interleaving and the third dimension specifies the moment in time. Interleaved sampling so enables efficient and correct motion blur simulation for e.g. the REYES architecture [4] or the photon map [17] using a very small number of time samples (i.e. the number of samples in the basis pattern). In Figure 4 the results of a motion blur simulation at 16 samples per pixel are shown. Using independent random samples no aliasing is visible, whereas the standard Accumulation Buffer technique exhibits visible artifacts. Finally a low discrepancy basis pattern of 64 samples, that are 2×2 interleaved, is used. Despite the low sampling rate the aliasing artifacts are clearly reduced.

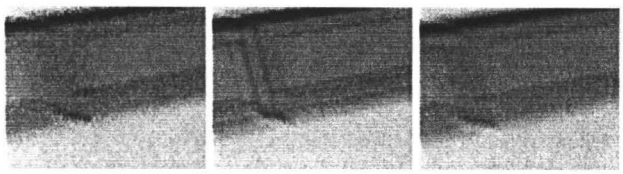


Figure 4: Comparison of different sampling schemes for motion blur. In all images we use 16 samples per pixel. From left to right: independent random samples, method of dependent tests (Accumulation Buffer), and interleaved sampling.

4.3 Volume Rendering

The next application we consider is volume rendering. We use a simple emission-absorption model like it is frequently used in medical and scientific visualization [9]. This results in a situation where a one-dimensional integral has to be computed for every pixel in the image. That is, the densities in the volume have to be mapped to emissions and absorptions that are integrated along the viewing ray for any individual pixel.

In software, the integral is usually achieved by ray-tracing. Emission and absorption of the volume are evaluated at discrete sampling points along any given viewing ray. It can be shown that regular sampling is optimal in terms of convergence for this kind of one-dimensional integral [26].

For interactive applications, we can also apply a ray-tracer, either on a high-end parallel system [28], or on dedicated volume ray-tracing hardware [30]. Another popular approach is to use more conventional texture-mapping hardware for volume rendering [3, 35]. In this latter case, which we want to concentrate on, the volume is sliced with equidistant planes. These are sorted back-to-front and blended together in the framebuffer. As a result of this algorithm, we obtain a sampling pattern that corresponds to a regular 3D grid in clip coordinates (i.e. after the projective transform).

Typically, on the order of 100 slices are used for this kind of application, and each slice yields a polygon that covers a large portion of the screen and is textured with a 3D texture with tri-linear interpolation. Thus, the fill-rate required by this method is quite high, while the cost of geometric transformations is negligible.

But even with 100 slices, there are often still some serious aliasing artifacts left. After all, modern medical imaging devices can generate data sets in the order of 512^3, and therefore 100 slices would result in an undersampling of a factor of 5. On the other hand it is not possible to increase the number of slices arbitrarily, both because of the performance implications and because of the limited depth of the framebuffer that results in quantization artifacts when using alpha blending.

Interleaved sampling can be used to improve on this situation. If we would like to keep the optimal convergence of the regular sampling along each viewing ray, the only degree of freedom is to offset the samples slightly different along each viewing ray, that is, for each pixel in the final image. We could use different, random offsets for all of the pixels, but then we would loose the coherence in the sampling that allows us to arrange the samples as a stack of planes. Interleaved sampling allows us to determine the offsets in a coherent way, so that we can still use graphics hardware for the rendering.

Using existing graphics hardware, the method then proceeds similarly to the method described in Section 4.1. First, we use the traditional texture-based volume rendering algorithm [3] to render the lower resolution sub-grids, each with the sampling planes slightly offset compared to the other sub-grids. If we use a configuration without supersampling, for example four sub-grids with half the grid spacing of the original image, then cost of this operation is

exactly as high as the original algorithm, since volume rendering is completely determined by the rasterization cost.

In a second step we then copy the sub-grids from the framebuffer to the texture RAM, and render them with the respective stipple patterns as described in Section 4.1. Figure 5 illustrates this method for the configuration with 4 sub-grids. The cost of this second step is constant, that is, it does not depend on the number of sampling planes. During our tests with an SGI Octane VPro, we found this constant cost to be about the same as rendering 8 additional planes with the original method. The quality improvements we can achieve with interleaved sampling are much higher than the improvements we get by adding this number of additional samples.

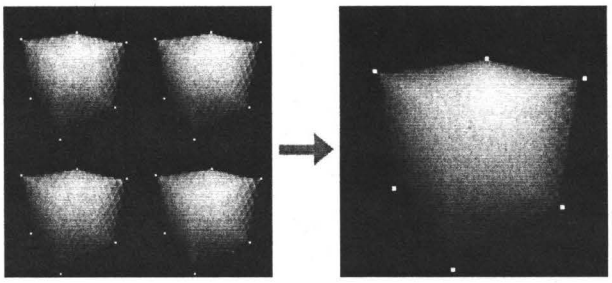


Figure 5: For volume rendering, we combine a collection of small resolution images with slightly offset sample planes into a large resolution image by interleaving the samples from the smaller images. Except for a final combining operation, the fill rate required is the same as for rendering a large image in the first place, but the rendering quality is dramatically better.

Figure 6 shows a comparison of the algorithm with and without interleaved sampling for 20, 60, and 120 sampling planes. On an Octane VPro, the frame rate for the 20 plane version is > 30 frames per second both with and without interleaving. For the 120 plane version we get around 18 frames per second in the interleaved case, and 20 frames per second in the non-interleaved case. Note that even for the 120 plane version there is still some aliasing left, and the 60 plane version with interleaved sampling looks better than the 120 plane version without. Due to the dithering in the printed version of this paper, these artifacts are only visible in a magnification, but they are quite obvious on the screen (please refer to the images on the CD-ROM). Also note that renderings with 120 sample planes already exhibit quite serious quantization artifacts due to the limited depth of the framebuffer. Thus, further increasing the number of sampling planes will not improve the quality.

Some more comparisons between the traditional volume texture-based volume rendering and interleaved sampling can be found in Figure 7 and on the CD-ROM.

4.4 Other Applications

The basic techniques of interleaved sampling presented in this paper naturally generalize to all Accumulation Buffer [12] and multi-pass frame-buffer [29] techniques while preserving the good spectral properties and the fast convergence. So it is straightforward to apply interleaved sampling to weighted sampling [12], to the simulation of extended light sources using deep shadow maps [22], and to global illumination simulations [18, 13].

5 Conclusion

In this paper we have presented a new sampling scheme we call *interleaved sampling*. The fundamental idea of our new method is to interleave samples from several regular global grids to form a locally irregular sampling pattern. The relative positions of the grids are determined by random samples of low discrepancy.

We have demonstrated that this scheme exhibits excellent spectral behavior and yields excellent quality especially for low sampling rates, while at the same time it is simple and efficient to generate.

Because the samples are interleaved from a number of regular grids, our method exhibits high coherence and thus is well suited for hardware implementations. We have discussed three different strategies for using interleaved sampling in hardware-accelerated applications. Firstly, a slightly modified implementation of the multisampling capabilities in some existing graphics systems. Secondly, a fast path for stipple patterns, a hardware feature implemented but not optimized in most contemporary graphics systems, and finally a strategy that can be used for applications limited by fill rate on current graphics hardware.

The feasibility of these methods has been demonstrated with a number of examples ranging from supersampling over motion blur to volume rendering.

6 Acknowledgments

For the image in Figure 7 we used an MR data set kindly provided Christoph Rezk-Salama, University of Erlangen-Nürnberg.

Part of this work was done in August 2000, while both authors stayed at the Max-Planck-Institute for Computer Science in Saarbrücken, Germany. The first author is especially thankful to Hans-Peter Seidel for granting the research stay and also was supported by mental images, Germany.

References

[1] K. Akeley. RealityEngine graphics. In *Computer Graphics (SIGGRAPH '93 Proceedings)*, pages 109–116, August 1993.

[2] L. Brotman and N. Badler. Generating Soft Shadows with a Depth Buffer Algorithm. *IEEE Computer Graphics and Applications*, 4(10):71–81, October 1984.

[3] B. Cabral, N. Cam, and J. Foran. Accelerated Volume Rendering and Tomographic Reconstruction Using Texture Mapping Hardware. In *1994 Symposium on Volume Visualization*, pages 91–98, October 1994.

[4] R. Cook, L. Carpenter, and E. Catmull. The Reyes Image Rendering Architecture. In Maureen C. Stone, editor, *Computer Graphics (SIGGRAPH '87 Proceedings)*, pages 95–102, July 1987.

[5] R. Cook, T. Porter, and L. Carpenter. Distributed Ray Tracing. In *Computer Graphics (SIGGRAPH 84 Conference Proceedings)*, pages 137–145, 1984.

[6] O. Deussen, S. Hiller, C. van Overveld, and T. Strothotte. Floating Points: A Method for Computing Stipple Drawings. *Computer Graphics Forum*, 19(3):C41–C50, 2000.

[7] D. Dobkin, D. Eppstein, and D. Mitchell. Computing the Discrepancy with Applications to Supersampling Patterns. *ACM Transactions on Graphics*, 15(4):354–376, October 1996.

[8] D. Dobkin and D. Mitchell. Random-Edge Discrepancy of Supersampling Patterns. In *Proceedings of Graphics Interface '93*, pages 62–69, May 1993.

[9] R. Drebin, L. Carpenter, and P. Hanrahan. Volume Rendering. In *Computer Graphics (SIGGRAPH '88 Proceedings)*, pages 65–74, August 1988.

[10] A. Frolov and N. Chentsov. On the calculation of certain integrals dependent on a parameter by the Monte Carlo method. *Zh. Vychisl. Mat. Fiz.*, 2(4):714 – 717, 1962. (in Russian).

[11] A. Glassner. *Principles of Digital Image Synthesis*. Morgan Kaufmann, 1995.

[12] P. Haeberli and K. Akeley. The Accumulation Buffer: Hardware Support for High-Quality Rendering. In *Computer Graphics (SIGGRAPH '90 Proceedings)*, pages 309–318, August 1990.

[13] W. Heidrich, K. Daubert, J. Kautz, and H.-P. Seidel. Illuminating Micro Geometry Based on Precomputed Visibility. In *Computer Graphics (SIGGRAPH '00 Proceedings)*, pages 455–464, July 2000.

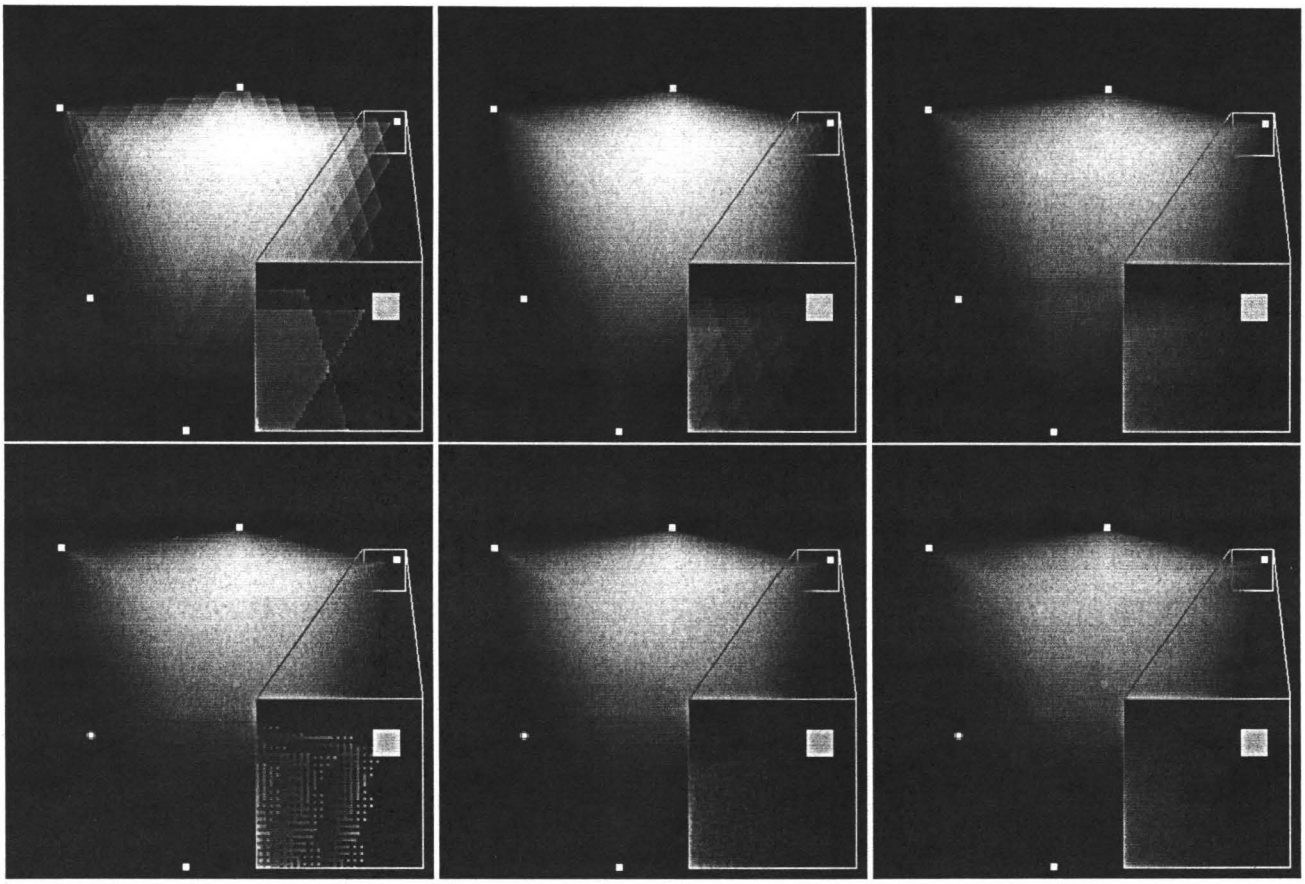


Figure 6: A comparison of interleaved sampling for volume rendering (bottom) with the traditional texture-based approach described by Cabral et al. [3] (top). From left to right: 20, 60, and 120 planes. Note that in the 120 plane example quantization artifacts due to the limited dynamic range of the framebuffer start to appear. To avoid these artifacts, smaller numbers of planes have to be used, in which case interleaved sampling produces dramatically better results than the original algorithm.

[14] W. Heidrich and H.-P. Seidel. Realistic, Hardware-accelerated Shading and Lighting. In *Computer Graphics (SIGGRAPH '99 Proceedings)*, pages 171–178, August 1999.

[15] W. Heidrich, Ph. Slusallek, and H.-P. Seidel. An Image-Based Model for Realistic Lens Systems in Interactive Computer Graphics. In *Graphics Interface '97*, pages 68–75, May 1997.

[16] E. Hlawka. Discrepancy and Riemann Integration. In L. Mirsky, editor, *Studies in Pure Mathematics*, pages 121–129. Academic Press, New York, 1971.

[17] H. Jensen and P. Christensen. Efficient Simulation of Light Transport in Scenes with Participating Media using Photon Maps. In Michael Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 311–320. ACM SIGGRAPH, Addison Wesley, July 1998.

[18] A. Keller. Instant Radiosity. In *Computer Graphics (SIGGRAPH '97 Proceedings)*, pages 49–56, August 1997.

[19] D. Kirk and J. Arvo. Unbiased Sampling Techniques for Image Synthesis. In *Computer Graphics (SIGGRAPH '91 Proceedings)*, pages 153–156, July 1991.

[20] R. Klassen. Filtered Jitter. *Computer Graphics Forum*, 19(4):223–230, 2000.

[21] C. Kolb, P. Hanrahan, and D. Mitchell. A Realistic Camera Model for Computer Graphics. In *Computer Graphics (SIGGRAPH '95 Proceedings)*, pages 317–324, August 1995.

[22] T. Lokovich and E. Veach. Deep Shadow Maps. In *Computer Graphics (SIGGRAPH '00 Proceedings)*, pages 385–392, July 2000.

[23] M. McCool and E. Fiume. Hierarchical Poisson disk sampling distributions. In *Proceedings of Graphics Interface '92*, pages 94–105, May 1992.

[24] D. Mitchell. Generating Antialiased Images at Low Sampling Densities. In *Computer Graphics (SIGGRAPH '87 Proceedings)*, pages 65–72, July 1987.

[25] D. Mitchell. Consequences of Stratified Sampling in Graphics. In *Computer Graphics (SIGGRAPH '96 Proceedings)*, pages 277–280, August 1996.

[26] H. Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods*. SIAM, Pennsylvania, 1992.

[27] A. Owen. Randomly Permuted (t, m, s)-nets and (t, s)-sequences. In H. Niederreiter and P. Shiue, editors, *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*, pages 299–315. Springer, 1995.

[28] S. Parker, M. Parker, Y. Livnat, P.-P. Sloan, C. Hansen, and P. Shirley. Interactive Ray Tracing for Volume Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 5(3):238–250, July 1999.

[29] M. Peercy, M. Olano, J. Airey, and J. Ungar. Interactive Multi-Pass Programmable Shading. In *Computer Graphics (SIGGRAPH '00 Proceedings)*, pages 425–432, July 2000.

[30] H. Pfister, J. Hardenbergh, J. Knittel, H. Lauer, and L. Seiler. The VolumePro Real-Time Ray-casting System. In *Computer Graphics (SIGGRAPH '99 Proceedings)*, pages 251–260, August 1999.

[31] M. Segal and K. Akeley. *The OpenGL Graphics System: A Specification (Version 1.2)*, 1998.

[32] SGI Inc. *Multisample Extension*, revision 1.1 edition, January 2000.

[33] P. Shirley. Discrepancy as a Quality Measure for Sample Distributions. In *Eurographics '91*, pages 183–194, September 1991.

[34] I. Sobol. The use of w^2-distribution for error estimation in the calculation of integrals by the Monte Carlo method. *U.S.S.R. Comput. Math. and Math. Phys.*, 2:717 – 723, 1962.

[35] R. Westermann and T. Ertl. Efficiently Using Graphics Hardware in Volume Rendering Applications. In *Computer Graphics (SIGGRAPH '98 Proceedings)*, pages 169–178, July 1998.

7 Appendix: Voronoi-based Relaxation

Sampling points with a blue noise spectrum are desirable, since these optimally mask aliasing caused by violating Shannon's sampling theorem by noise. In [6] a deterministic iteration scheme for Lloyd's method has been presented to generate such points. Here we introduce a new and efficient Monte Carlo version of this algorithm, however with periodic boundary conditions. Lloyd's method starts by choosing an initial set of N random points x_i. Then each iteration step computes the Voronoi diagram and moves each point x_i into the center of mass of its Voronoi cell. The fast converging iteration scheme is sketched in Figure 8. Although already running in a very short time, the algorithm can be further improved by using a sd-tree for the nearest neighbor search and by using stratified sampling for the estimation of the Voronoi diagram. For the sake of clarity these obvious optimizations have been omitted here. An example run of the algorithm can be seen in Figure 9.

The relaxation scheme reduces the variance of the nearest neighbor distance while the periodic boundary conditions guarantee that the sampling pattern can be replicated seamlessly. In addition it can be shown that the minimum and maximum nearest neighbor distance are bounded from below and above, respectively, which makes the points even more powerful than Poisson disk samples. Our new algorithm thus overcomes many drawbacks of previous Poisson disk sample generation methods as surveyed in [11] and is much simpler than the scheme described by Haeberli an Akeley [12].

```
for(int k = 0; k < Iterations; k++)
{
 for(i = 0; i < N; i++)
 {
  Δ_i = 0;
  c_i = 0;
 }

 // estimate the centers of the
 // Voronoi cells defined by the x_i

 for(i = 0; i < M * N; i++)
 {
  draw x ~ U(I^s);
  d_max = ∞;

  // find max so that x_max is
  // closest to x on the torus I^s

  for(j = 0; j < N; j++)
  {
   Δ' = (x − x_j + (1/2,...,1/2)) mod I^s
                 −(1/2,...,1/2);
   d = ||Δ'||_2^2;

   if(d_max > d)
   {
    d_max = d;
    max = j;
    Δ = Δ';
   }
  }

  c_max++;
  Δ_max += Δ;
 }

 // move x_i into the estimated
 // center of its Voronoi cell

 for(i = 0; i < N; i++)
  if(c_i > 0)
   x_i = (x_i + Δ_i/c_i) mod I^s;
}
```

Figure 8: The relaxation scheme takes N random points x_i. After the iteration N points with approximate blue noise spectrum are returned. For the Monte Carlo estimation of the Voronoi cells, M times as many samples x as there are points are drawn uniformly from the s-dimensional unit cube $I^s := [0,1)^s$. The mod I^s operation componentwise returns the positive fractional part of a real vector, e.g. $-0.3 \bmod I = 0.7$ and $2.4 \bmod I = 0.4$.

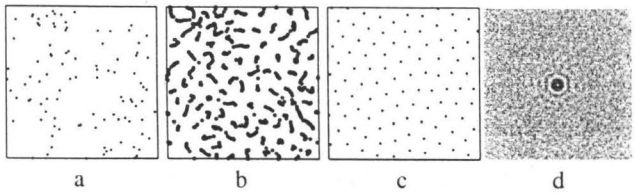


Figure 9: Example run of the stochastic point relaxation for $N = 100$ points and an oversampling rate of $M = 10000$. In a) the original random points are shown. During iteration these are moved along the trajectories in b) and after 20 iterations yield the points in c) with the Fourier spectrum d).

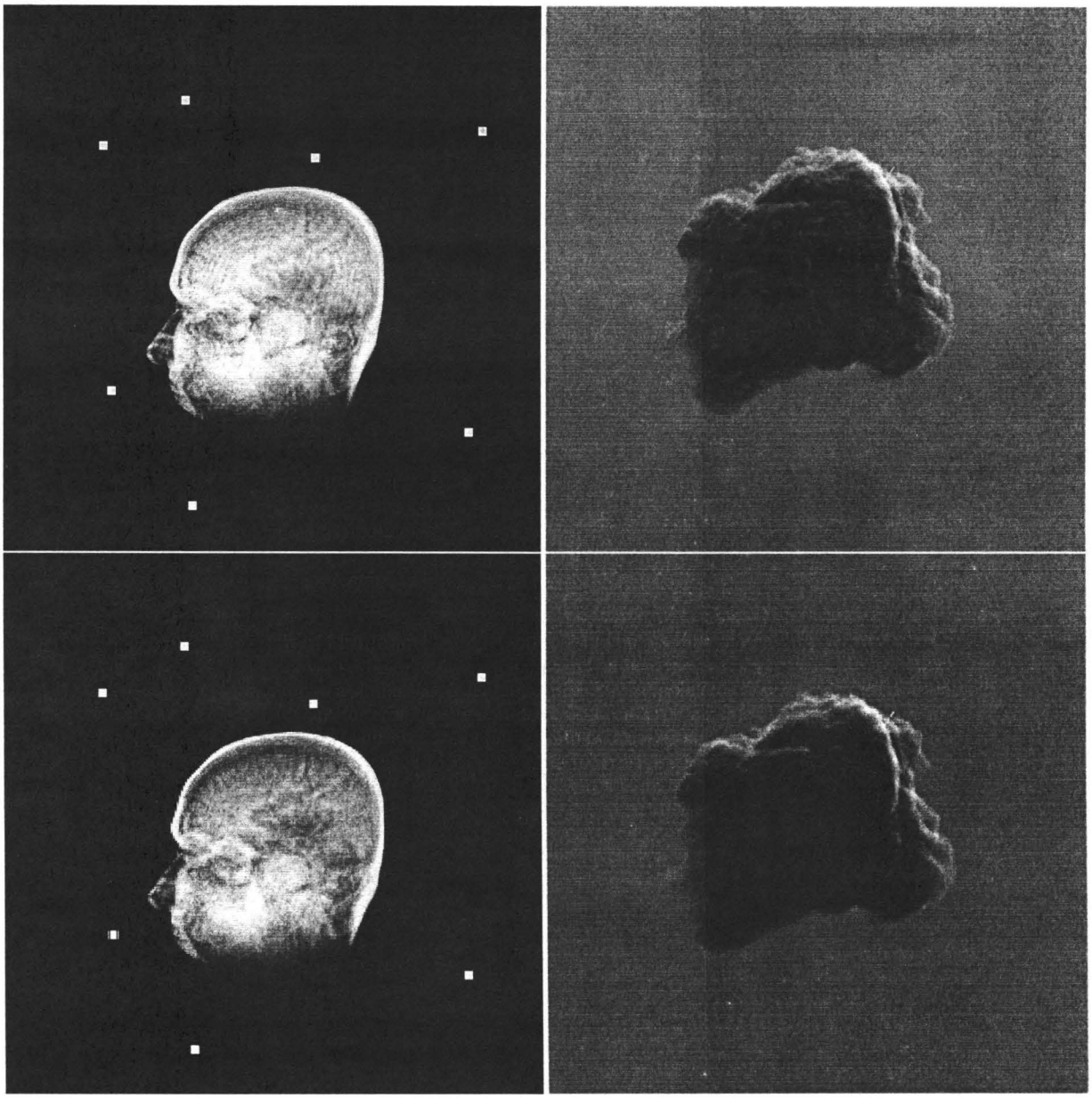


Figure 7: Examples of our volume rendering algorithm with real data sets. Top: traditional texture based method exhibiting aliasing artifacts that show up as ringing structures. Bottom: interleaved sampling with drastically reduced aliasing. All images are also on the CD-ROM.