Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

ANDREA KRATZ, JAN REININGHAUS,
MARKUS HADWIGER, INGRID HOTZ

Adaptive Screen-Space Sampling for Volume Ray-Casting

ZIB-Report 11-04 (February 2011)

Adaptive Screen-Space Sampling for Volume Ray-Casting

Abstract

This work is concerned with adaptive screen-space sampling for volume ray-casting. The goal is to reduce the number of rays being cast into the scene and, thus, the overall number of sampling points. We guarantee reliable images through explicit error control using an error estimator that is founded in the field of finite element methods (FEM). FEM theory further provides a well-founded theory to prove the efficiency of the presented algorithm via convergence analysis. We, therefore, compare the convergence behavior of our method against uniform subdivisions and a refinement scheme that was presented in the context of CPU volume ray-casting [Lev90]. Minimizing the number of sampling points is of interest for rendering large datasets where each evaluation might need an expensive decompression. Furthermore, with increasing screen resolutions high-resolution images are created more efficiently with our method.

1. Introduction and Related Work

The goal of this work is to reduce the number of sampling points via error-controlled adaptive screen-space sampling. That is, the sampling frequency is increased in image regions where color variations across space is high (e.g., edges), and it is decreased in homogeneous image regions. The heart of such methods are criteria to estimate this variance and the resulting image-space error. In contrast to previous work that also deals with adaptive image discretization, the strength of our method lies in an ubiquitous error estimator founded in the FEM theory [Bra07, AO00]. To guarantee interactivity independent from the screen resolution, we interconnect the adaptive refinement with progressive image generation: Approximations of the exact image are iteratively computed and displayed until the algorithm has converged to pixel accuracy. For volume rendering, we use ray-casting.

Whereas adaptive methods in image space are common in the area of ray-tracing, less attention has been paid to similar approaches in the volume rendering domain [HKRSW06]. In volume visualization, [Lev90] introduced adaptive image-space subdivision combined with progressive image generation for performance improvement. He uses a simple error measure, where missing features are accepted as payoff for speed. To reduce aliasing artifacts, adaptive stochastic ray tracing has been introduced [PS89]. The refinement process is driven by the variance of neighboring samples. Especially for isosurface rendering, there have been further improvements, also handling missing features [AAN05, GM07, DWL05]. Most of these methods are built on refinement criteria specific to surface rendering relying on the surface normals. Consequently, they cannot be easily adapted to general volume rendering. Further improvements of image-space techniques can be achieved by considering spatial [KSSE05, AAN05] and temporal coherence, or even both [DWL05, WLWD03].

Recently, methods that combine adaptive sampling in image and object space have been suggested. [Lju06] use a multi-resolution volume scheme to guide the sampling density along rays. The sampling density in image space is determined by projecting volume block statistics back to the image plane. [KSKE07] suggested a combined method for texture-based volume rendering. Their main motivation is to deal with large compressed data directly on the GPU. Since decompression is expensive, they apply an oracle that determines a local sampling distance being used to skip samples. Their method is successful in reducing the number of sampling points. However, their approach is restricted to texture-based volume rendering and not suited for a progressive approach.

Although almost all of these approaches are effective in improving the performance, less attention is paid to explicit error control. Error metrics are mostly based on spatial and temporal heuristics. Our method is motivated by FEM theory, which provides a well-founded theoretical framework connecting approximation schemes, error estimators, refinement schemes, and the corresponding convergence behavior of the approximative solution [AO00]. By translating common problems from the area of volume rendering into the

language of FEM (Sections 2, 3), we are able to demonstrate the efficiency of the proposed algorithm in terms of convergence behavior (Section 7). We, therefore, compare it to uniform screen-space subdivisions and a previously presented refinement scheme [Lev90].

2. Problem Defintion from the Perspective of FEM

FEM is a numerical method that is used to find approximate solutions for partial differential equations. The basic idea consists of a lattice-like partition of a specified domain Ω into geometric primitives [Bra07], the finite elements, on which approximate solutions can be computed much simpler.

To underline the motivation of using FEM methods for volume rendering, we first translate the common problem of approximating the widely used emission-absorption model [HKRSW06] to the language of FEM. If other optical models [Max95] are used, additional approximation errors (Section 3) might arise (e.g., due to the approximation of gradients), which are not covered in this work.

2.1. Emission-Absorption Model

Given a volume $\Omega \subset \mathcal{R}^3$, a scalar absorption function A, a scalar emission function E defined over Ω and a normalized vector field W that describes the propagation of light through the volume [†], the emission-absorption model leads to a partial differential equation, which determines the radiance I (see [HHS93] for a general discussion)

$$\begin{aligned} \nabla I \cdot W + AI &= E \quad \text{in } \Omega, \\ I &= 0 \quad \text{on } \partial\Omega. \end{aligned} \tag{1}$$

Let w^x denote a parametrization by arc length of the light ray of $-W$ that originates in x, ends at the boundary of Ω, and has length L^x. Then the integral form of Equation (1) is given by

$$I(x) = \int_0^{L^x} E\left(w^x\right) \exp\left(-\int_0^t A(w^x)\,ds\right) dt. \tag{2}$$

2.2. Approximation

In order to render an image, we are only interested in the values of I on a surface S (the viewport) embedded in Ω. That is, we are looking for a function

$$u := I|_S \tag{3}$$

defined in some function space $V(S)$. In general, finding a closed formula for the exact solution $u \in V(S)$ is not possible. Therefore, a discretization method is applied that computes a sequence $(n = 0, \ldots, \infty)$ of approximative solutions u_n that converge to the exact solution u. The approximation error that arises and which we seek to minimize, is measured using the L_2 error norm

$$\eta = \|u - u_n\|_S := \left(\int_S (u - u_n)^2\,dx\right)^{\frac{1}{2}}. \tag{4}$$

[†] We call W a vector field to emphasize that light rays following W might be either parallel, perspective or even curved rays.

Symbol	Short Explanation
A	Scalar absorption function
E	Scalar emission function
W	Vector field that describes the propagation of light
I	Radiance
S	Viewport
$u \in V(s)$	Exact solution of the emission-absorption model
$u_n \in V(s)$	Approximative solution of the emission-absorption model
$\mathcal{T}_n$	Set of cells of the discrete space V_n
$Q \subset S$	Set of points whose values uniquely determine a function in V_n
$q \in Q$	Evaluation point
φ_q	Basis function corresponding to the evaluation at point $q \in Q$
$\mathcal{I}_n(u) \in V_n$	Nodal interpolant
$u(q)$	Exact value of the volume rendering integral in q
$I_h(q)$	Approximation of the volume rendering integral

Table 1: *Table of mathematical symbols that are used in this work.*

3. Error Estimator

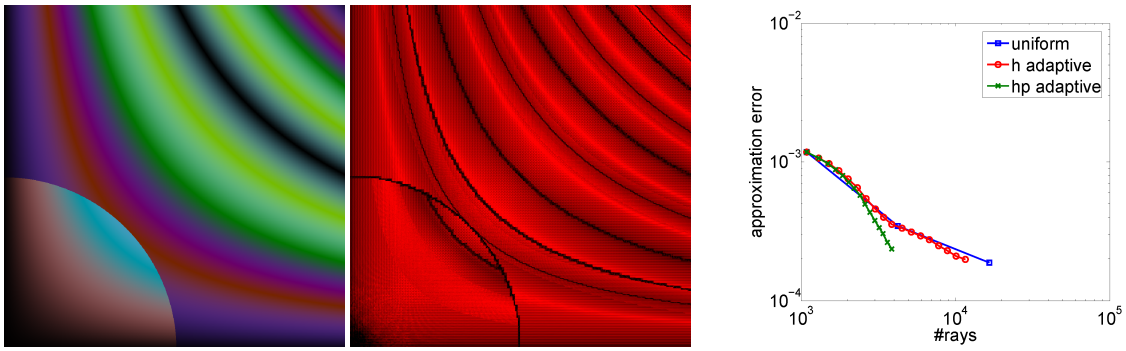
In this section we thoroughly derive the proposed error estimator. We use the following notations, which are founded in FEM theory (see also Table 3): Let $\mathcal{T}_n$ be the set of cells of the discrete space V_n and $Q \subset S$ the set of points whose values uniquely determine a function in V_n. Then, the nodal interpolant is given by

$$\mathcal{I}_n(u) = \sum_{q \in Q} u(q)\,\varphi_q \tag{5}$$

where $u(q)$ is the exact value of the volume rendering integral (Equation (2)) and φ_q the basis function corresponding to the evaluation point q. Given an approximation $I_h(q)$ of the volume rendering integral, the function u_n is finally given by

$$u_n = \sum_{q \in Q} I_h(q)\,\varphi_q \tag{6}$$

Given Equations (5) and (6), the total approximation error



(a) (b) (c)

Figure 1: *A piecewise analytic function (a) and the corresponding estimated regularity (red indicates high regularity, while black indicates low regularity) (b). Convergence graph comparing uniform vs. h-adaptive vs. hp-adaptive approximations. For analytical functions such as (a), hp-adaptivity leads to a faster convergence than h-adaptivity only.*

η can be bounded by:

$$\eta = \|u - u_n\|_S \leq \|u - \mathcal{I}_n(u)\|_S + \|\mathcal{I}_n(u) - u_n\|_S. \quad (7)$$

The first term on the right hand side represents the error of the viewport discretization, which can be reduced by refining the function space V_n by subdividing the cells. The second term represents the error due to the approximation of the volume rendering integral. It can be minimized by increasing the number of sample points employed in the quadrature formula.

To control the adaptive refinement, we consider the local error contributions on each cell $T \in \mathcal{T}_n$

$$\eta_T = \|u - u_n\|_T \leq \underbrace{\|u - \mathcal{I}_n(u)\|_T}_{:=\eta^T_{V_n}} + \underbrace{\|\mathcal{I}_n(u) - u_n\|_T}_{:=\eta^T_{I_h}}. \quad (8)$$

The error quantity $\eta^T_{I_h}$ can be controlled, for example, via adaptive ray integration. In this work, we focus on the estimation of the viewport error $\eta^T_{V_n}$. To localize image regions with large error contributions, we estimate $\eta^T_{V_n}$ using an hierarchical a-posteriori error estimator [AO00]. As we do not have any knowledge of the exact solution u, we compare two subsequent approximations:

$$\eta^T_{V_n} \approx \|u_{n-1} - u_n\|_T = \left(\int_T (u_{n-1} - u_n)^2 \, dx\right)^{\frac{1}{2}}. \quad (9)$$

The integral on the right hand side is a 2D polynomial of degree $k = 4$. It can be solved numerically, for example, using a Gaussian quadrature rule

$$\eta^T_{V_n} \approx \sum_{i=1}^{N} \phi(x_i) w_i, \quad (10)$$

with N being the number of points x_i for which the polynomial is evaluated and w_i being the weights. For the computation we separate the polynomial into two 1D polynomials and choose $N = 3 \leq 2k - 1$.

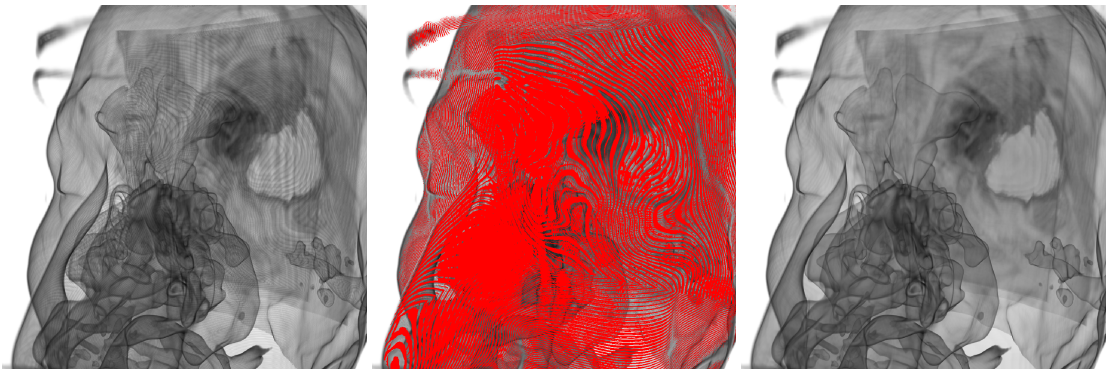
4. *h*- vs *p*- vs. *hp*- FEM

We distinguish three versions of the finite element method: the *h*-, *p*- and *hp*-version. The most popular is *h*-FEM, where the convergence of the approximative solution is achieved by increasingly finer grids. In *p*-FEM convergence is achieved by increasing the polynomial degree on a uniform grid of fixed-size finite elements. The basic idea of *hp*-FEM is to combine the advantages of *h*- and *p*-adaptivity to improve the convergence: In regions of high frequency, a fine grid and low polynomial degrees are chosen. In regions where u is smooth, a coarse grid with high polynomial degrees is preferable.
Depending on the regularity of the function u, which gives us a hint about the smoothness of the function, this leads to a method that converges algebraically, or even exponentially, to the exact solution. For analytical functions with high regularity *p*-refinement is dominant so that *hp*-adaptivity leads to a faster convergence than *h*-adaptivity only (Figure 1).

In the case of volume rendering, the order of convergence is constrained by the smoothness of the signal that results from the input data mapped to color and opacity values by a transfer function [‡]. Real data often has many discontinuities and, thus, hardly leads to any *p*-refinement. More sophisticated reconstruction kernels [HVTH02] can improve the convergence behavior as they increase the smoothness of the input data. However, often high-frequency transfer functions are desired to distinguish different materials in the final image. Consequently, *hp*-adaptivity would result in marginal benefit for highly inhomogeneous data.

In this work, we focus on *h*-adaptivity subdividing image regions of high frequency and bilinear interpolation for image reconstruction.

[‡] Assuming the emission-absorption model.



(a) (b) (c)

Figure 2: *Comparison of uniform vs. adaptive ray integration. Using a constant sampling distance results in visible image artifacts (a). These are identified by our adaptive ray integrator (b). Our simple step doubling scheme for the ray integration results in an improved image quality (c).*

5. Adaptive Screen Space Sampling

The algorithm that is presented in this section iteratively computes approximations u_n of the exact solution u. In this work, the exact solution is defined by the emission-absorption model (Equation (1)). We start with a coarse uniform grid. The adaptive refinement into quadratic cells is stored in a quadtree data structure. Each iteration computes a new discrete approximation u_n that converges against u following a standard loop from adaptive finite element theory [AO00]:

1. Estimate η^T (Equation (8)) for each *new* quadtree leaf.
2. Mark a fixed number m of those cells that have the biggest impact on the total error η (Equation (4)).
3. Refine the marked cells, i.e., create four new quadtree leafs for each marked cell. Together with the new quadtree leafs, new evaluation points arise.
4. Compute an approximation of the volume rendering integral (Equation (2)) for the new evaluation points.
5. Reconstruct the new approximation u_n (Equation (2)) and render the solution to the screen.

The loop runs, until the algorithm converges to pixel accuracy, or a user-specified error bound is reached. It is interrupted whenever the camera moves or the transfer function changes. To guarantee interactivity independent from screen resolutions, we interconnect the adaptive with progressive refinement. The image quality, thus, progressively improves.

6. Implementation Details

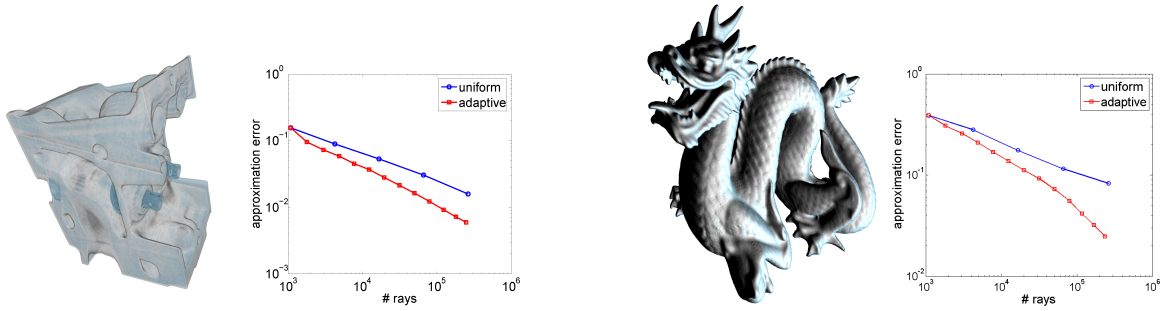
Our current prototype is implemented in C++ and OpenGL/GLSL requiring a Shader Model 4.0 GPU. The work of the algorithm is distributed between CPU and GPU leaving the data structures on the CPU. Error estimation, ray-casting and reconstruction of the final image are carried out on the GPU.

The estimator computes the approximation error η_T only on the set of *new leafs* that were generated in the previous iteration. Output is a 1D single-channel floating-point texture with m entries (Section 5) that is read back to main memory for the following steps.

To mark those m cells that contribute the most to the total error, we employ a heap data structure on the quadtree leafs that is updated each time new leafs are created, which leads to a cost of $O(m \log n)$ for sorting the quadtree leafs. The parameter m can be adjusted by the user.

The refinement creates new leafs and with each new leaf a maximum of five evaluation points is generated: one at the cell's center and four at the midpoints of the cell's edges. In either case, a new evaluation point for the cell's center arises, which leads to a minimum of one sample per new leaf. Note, that a sample is never computed twice.

For each new evaluation point standard ray-casting is performed to compute an approximation of the volume rendering integral. To control the error $\eta_{I_h}^T$, we employ a simple step doubling. Similar to the error measure for the viewport refinement, we compare two subsequent refinements. In each integration step, two color values are taken into account: one is computed using the full stepsize (h) according to the current sampling distance, and one is computed using two half steps ($h/2.0$). The squared difference of both color values gives us a hint about the accuracy associated with the current stepsize. If the difference falls below the requested error bound, the evaluation is accepted and the stepsize is increased. Otherwise the stepsize is decreased and again two integration steps are performed to measure the difference between the two results. The error bound for the ray integration



(a) (b) (c) (d)

Figure 3: *High-resolution direct volume rendering ($2k \times 2k$) of a CT scan of an engine block (a) and iso-surface rendering of a distance field (c). In each case, the corresponding convergence graphs are given on the right (b,d). The y-axis represents the approximation error (log-scale) while the x-axis denotes the number of computed rays. Red depicts the convergence behavior of our approach and blue depicts the convergence behavior of standard uniform discretizations.*

can be chosen by the user, and it is independent from the error bound that is used for screen-space sampling.

In a last step, the approximation is rendered to the screen by reconstructing u_n on a uniform grid. In this work, we use a fixed polynomial degree of $p = 2$ for the interpolation (Section 4).

7. Results

To determine the efficiency of our algorithm, we compared the convergence behavior of our method to a uniform subdivision of the image space and to the adaptive refinement scheme proposed by [Lev90] (Section 7.3).

7.1. Datasets

We have applied our method to the following datasets:

Head: The dataset depicted in Figure 2 is a CT scan of human head. The byte data is given on a uniform grid with a resolution of $256 \times 256 \times 225$.

Engine: The dataset depicted in Figure 3 (a) is a CT scan of an engine block. The byte data is given on a uniform grid with a resolution of $256 \times 256 \times 256$.

Dragon: The dataset depicted in Figure 3 (b) is a distance field. The float data is given on a uniform grid with a resolution of $256 \times 256 \times 256$.

Mixing Layer: The dataset depicted in Figure 4 describes the mixing of two fluids. The float data is given on a uniform grid with a resolution of $682 \times 61 \times 132$. Figure 4 shows a direct volume rendering of the magnitude of the velocity field that emerges during mixing.

7.2. Setup

As the goal of this work is the reduction of sampling points while simultaneously keeping the approximation error low, we have chosen the following setup for the convergence analysis: As we do not have any knowledge about the exact solution u, we replaced it with an extremely oversampled version, i.e. an image resolution of $2k \times 2k$ and a sampling distance of $1/4096$ along the ray.
The convergence graphs then compare the number of rays that are shot for each evaluation point against the approximation error (y-axis) given by the relative L_2 error:

$$\eta_{rel} = \|u - u_n\|_S / \|u\|_S. \tag{11}$$

7.3. Convergence

To determine the convergence behavior, we first compared our adaptive approach to a uniform discretization of the image plane (Figure 3).

Comparing convergence graphs of adaptive and uniform image-space subdivions (Figures 3, 4), it can clearly be seen that our adaptive approach leads to a faster convergence in all our examples. That is, fewer rays are needed for a desired image quality.

We further compared our adaptive refinement scheme to Levoy's adaptive image-space subdivision [Lev90]. The basic idea of the error metric proposed in [Lev90] is to consider the variance of the values at the cells' corner pixels. To incorporate this error estimator into our method, we modified the *estimation-stage* of our algorithm. This allows for a direct and fair comparison of the error estimators, although the resulting overall algorithm is not the same as the one proposed in [Lev90].

The results of the comparison are summarized in Figure 4.

The convergence graph shows the relative error η_{rel} (Equation (11)) of the different refinement schemes: uniform vs. [Lev90] vs. this paper. As expected, both adaptive methods outperform the uniform refinement. Levoy's error estimator produces images with lower error in the beginning (less than 128^2 rays). For more than 256^2 rays, considerably better results are achieved by our estimator.

The better convergence behavior of Levoy's approach in the beginning is because it is more sensitive to edges. Whereas our approach distributes rays more evenly, Levoy's approach resolves edges more quickly.
In linear image regions with strong slope, however, our approach is more efficient. Consider a linear function with slope 1, which is perfectly represented (with zero error) by a linear interpolant regardless of the underlying discretization. Levoy's error estimator, which is based on the variance of the corner pixels, will always indicate an approximation error regardless of the grid and, thus, would refine those regions without reducing the error.

This behavior can be seen in Figure 4 (middle row). The color of the red block on the left fades slowly into the background. This is well approximated by the bilinear interpolation using rather coarse cells. The variance of the corner pixels is rather large, which leads to an unnecessarily strong refinement in this region applying Levoy's error metric (Figure 4(f)). The hierarchical error estimator presented in this paper notices the good approximation behavior in this rather linear region of the image and focuses the refinement on the interior edge of the red block (Figure 4(g)).

7.4. Performance

The initial grid has a resolution of 256×256 and the number of cells to refine in one iteration is $m = 1000$. Using these parameters, 7 iterations are needed to produce an image with an approximation error of 10^{-2} for the dataset shown in Figure 4. In our current implementation a single iteration takes about 20 up to 40 ms, for a screen resolution of 512^2 and 2048^2, respectively [§].

8. Conclusion and Future Work

We introduced an adaptive algorithm that is used in the field of finite element methods to GPU-based volume rendering. We have shown that we are able to reduce the number of rays being cast into the scene and, thus, the overall number of sampling points while minimizing the approximation error. In the future, we will use the adaptive screen-sampling in combination with out-of-core techniques. In that case, a minimization of the number of sample points is of special importance, since each evaluation needs an expensive decompression.

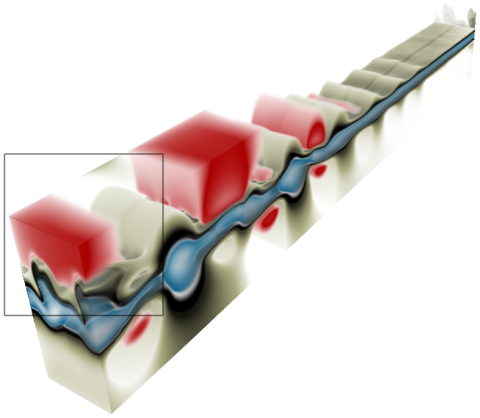
Our work focused on explicit error control. Performance issues were not covered yet. Future work, therefore, will comprise an improved implementation of the algorithm using CUDA. Thus, all steps of the algorithm can be executed on the GPU and we do not have to distribute the workload between GPU and CPU, which currently reduces the performances.

Finally, in this work we focused on h-adaptivity. Even better convergence behavior might be achieved with hp-adaptivity in combination with higher-order reconstruction kernels.

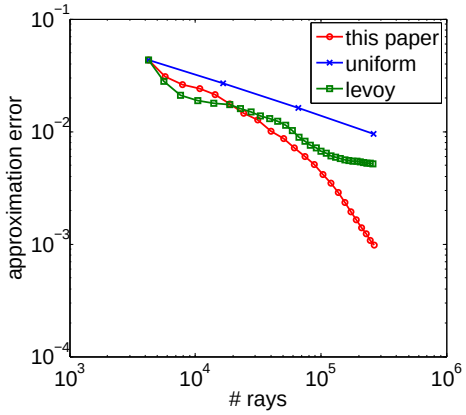
Acknowledgements

This project was funded by the DFG - Emmy Noether Research Group. The pseudo-spectral direct numerical simulation that produced the mixing layer dataset was carried out by Pierre Comte.

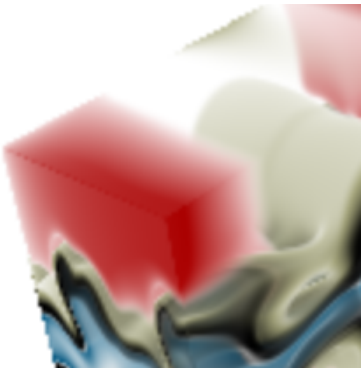
§ The performance was measured on a system equipped with an Intel Core 2 Duo CPU with 3.0 GHz and an NVIDIA GeForce 8800GT GPU.



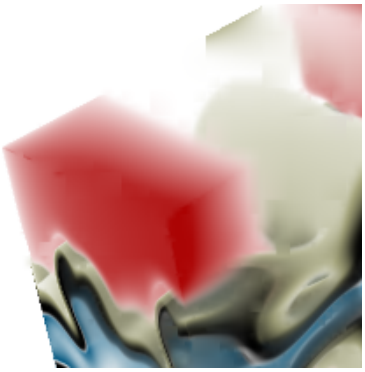
(a)



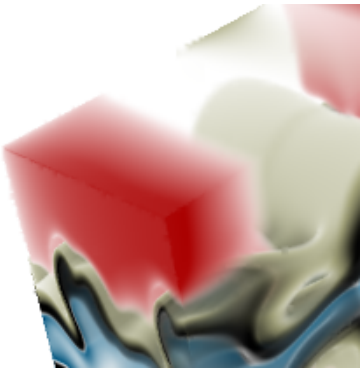
(b)



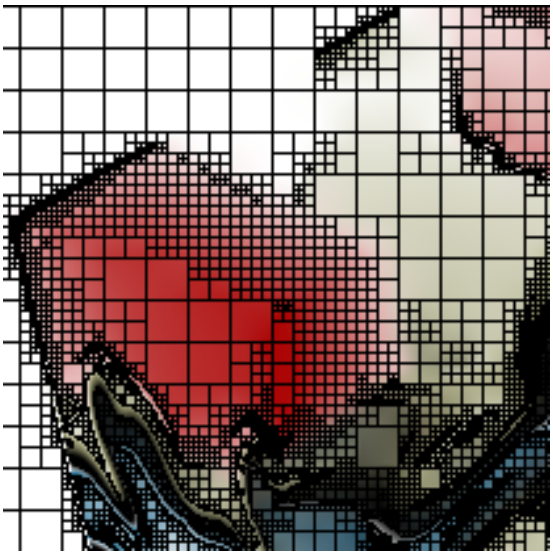
(c)



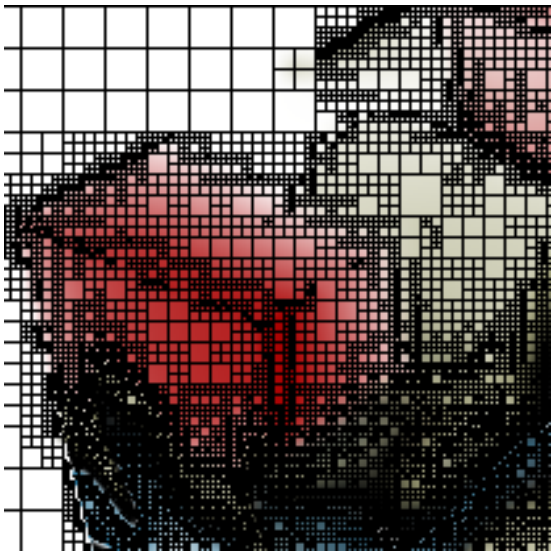
(d)



(e)



(f)



(g)

Figure 4: *Top: Direct volume rendering of a dataset from computational fluid dynamics* ($682 \times 61 \times 132$, *float dataset) (a) and corresponding convergence graph (b). Middle: Cutout of the image showing a uniform* (256×256) *viewport discretization (c), an adaptive discretization based on Levoy's algorithm [Lev90] using* 256^2 *rays (d) and our adaptive discretization with* 256^2 *rays (e). Bottom: Quadtree of Levoy's discretization (f) and our algorithm (g). Whereas edges at the border of the volume are detected by Levoy's algorithm, edges inside the volume are missed. A more consistent approximation is achieved by our adaptive discretization. Data courtesy of Pierre Comte.*

References

[AAN05] ADAMSON A., ALEXA M., NEALEN A.: Adaptive sampling of intersectable models exploiting image and object-space coherence. In *Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2005), ACM, pp. 171–178. 2

[AO00] AINSWORTH M., ODEN J. T.: *A Posterori Error Estimation in Finite Element Analysis*. John Wiley & Sons, 2000. 2, 4, 5

[Bra07] BRAESS D.: *Finite Elements: Theory, Fast Solvers, and Applications in Solid Mechanics*. Cambridge University Press, 2007. 2, 3

[DWL05] DAYAL A., WATSON B., LUEBKE D.: Adaptive frameless rendering. In *Eurographics Symposium on Rendering* (2005), Bala K., Dutre P., (Eds.). 2

[GM07] GAMITO M. N., MADDOCK S. C.: Progressive refinement rendering of implicit surfaces. *Comput. Graph. 31*, 5 (2007), 698–709. 2

[HHS93] HEGE H.-C., HÖLLERER T., STALLING D.: *Volume Rendering - Mathematical Models and Algorithmic Aspects*. Tech. Rep. TR 93-7, Zuse Institute Berlin (ZIB), 1993. 3

[HKRSW06] HADWIGER M., KNISS J. M., REZK-SALAMA C., WEISKOPF D.: *Real-time Volume Graphics*. AK Peters, Ltd., 2006. 2, 3

[HVTH02] HADWIGER M., VIOLA I., THEUSSL T., HAUSER H.: Fast and flexible high-quality texture filtering with tiled high-resolution filters. In *Vision, Modeling and Visualization'02* (2002), pp. 155–162. 4

[KSKE07] KRAUS M., STRENGERT M., KLEIN T., ERTL T.: Adaptive sampling in three dimensions for volume rendering on gpus. In *Proceedings APVIS '07* (2007). 2

[KSSE05] KLEIN T., STRENGERT M., STEGMAIER S., ERTL T.: Exploiting frame-to-frame coherence for accelerating high-quality volume raycasting on graphics hardware. In *IEEE Visualization '05* (2005), pp. 223–230. 2

[Lev90] LEVOY M.: Volume rendering by adaptive refinement. *Vis. Comput. 6*, 1 (1990), 2–7. 2, 3, 6, 7, 8

[Lju06] LJUNG P.: Adaptive sampling in single pass, gpu-based raycasting of multiresolution volumes. In *Proceedings of Volume Graphics '06* (2006), pp. 39–46. 2

[Max95] MAX N. L.: Optical models for direct volume rendering. *IEEE Trans. Vis. Comput. Graph. 1*, 2 (1995), 99–108. 3

[PS89] PAINTER J., SLOAN K.: Antialiased ray tracing by adaptive progressive refinement. In *SIGGRAPH '89* (1989), vol. 23, pp. 281–288. 2

[WLWD03] WOOLLEY J., LUEBKE D., WATSON B., DAYAL A.: Interruptible rendering. In *ACM Symposium on Interactive 3D Graphics* (2003), pp. 143–152. 2