

Using a GPU-based Framework for Interactive Tone Mapping of Medical Volume Data

Matthias Raspe* Stefan Müller†

Computer Graphics Working Group
Institute for Computational Visualistics
University of Koblenz-Landau

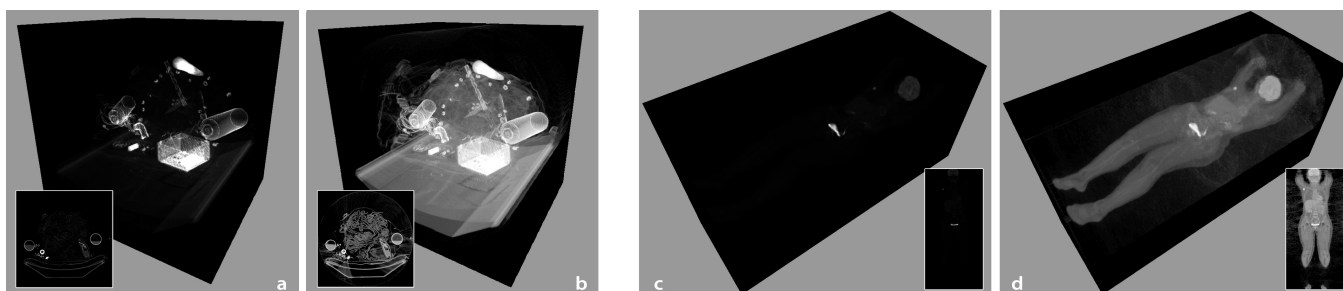


Figure 1: Direct volume rendering of high dynamic range volume data. The images on the left show a CT data set, on the right a PET scan of a human body, with insets depicting a slice of the according volumes for reference. While the linear mapping (a, c) reveals only the highest values of the data sets, tone mapping algorithms (b, d) can display the whole dynamic range in real-time.

Abstract

Medical workstations nowadays visualize large amounts of data from image acquisition systems that have dynamic ranges usually much higher than standard devices can display. In order to examine the data or control other processing steps, the user specifies windowing parameters to map the input values to the displayable output range. While this operation can be performed efficiently even on large datasets by using simple lookup tables, no acceptable performance is achieved when advanced algorithms from high dynamic range imaging are needed. Especially data from functional imaging modalities has a much higher dynamic range and requires considerable interaction for proper visualization using the traditional windowing approach. Therefore, we propose to integrate tone mapping algorithms into the visualization pipeline of volume data by exploiting modern graphics hardware. To allow for a flexible implementation and integration with other processing steps, we will present our programming framework and compare the performance to CPU implementations. In addition, we will discuss different tone mapping approaches in consideration of miscellaneous medical modalities and the role of transfer functions in the context of high dynamic range rendering.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation—Viewing algorithms; I.4.3 [Image Processing and Computer Vision]: Enhancement—Grayscale manipulation; I.4.10 [Image Processing and Computer Vision]: Image Representation—Volumetric

Keywords: graphics hardware, volume data processing, medical visualization, tone mapping

1 Introduction

During the last years, programmable graphics hardware (GPU) has become more important in different fields of applications [Owens et al. 2007]. This is mainly due to the high computing power clearly exceeding that of modern processors (CPUs) and thus offering a much better price-to-performance ratio (see figure 2). Driven by the growing entertainment market, the performance of the GPU’s inherently parallel and integrated architecture can be increased at a fast pace, making them especially attractive for data intensive tasks such as image processing, visualization, simulation, etc. Utilizing the graphics hardware not only for visualization is of particular interest, because the data being processed can be displayed without additional conversion or transfer, contrary to visualizing results of CPU algorithms.

However, there are some drawbacks with this development. First, using the graphics hardware for general purposes still involves graphics programming and thus thorough knowledge in computer graphics. In addition, complex algorithms are not easily ported to the GPU due to limitations in memory access, programming constructs, etc. This fact is even aggravated by the limited graphics memory which is usually much smaller than the host system’s memory. As a result, much communication between the GPU and the host is needed, being a typical bottleneck especially for memory intensive tasks. In order to investigate the benefit of commodity hardware for “real world” applications, we have built the cross-platform programming environment “Cascada”. The main intention of this system is realizing volume processing and visualization algorithms on GPUs, focused on (but not limited to) medical vol-

*e-mail: mraspe@uni-koblenz.de

†e-mail: stefanm@uni-koblenz.de

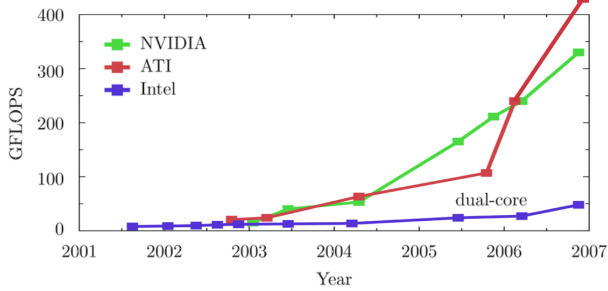


Figure 2: Development of raw performance for commodity graphics hardware compared to CPUs. ([Owens et al. 2007])

ume data. As GPUs are optimized for two-dimensional textures consisting of four channels, however, we use a compact volume representation in order to exploit the graphics hardware’s capabilities. This also allows for loading and working with volumes larger than the maximum texture size and is in addition compatible with older graphics systems. These textures are updated using an on-demand scheme, i.e., data is converted and transferred to the currently needed representation only once. Thus, interchanging GPU and CPU algorithms as part of a whole sequence processing the same data is simple, yet achieving the best performance for data transfer between the two devices. These sequences resemble one layer of our hierarchical representation of workflows.

As the data acquired and processed usually has a higher value range than standard output devices, different approaches can be applied to map the range of values to the display. For medical data, controlling a window of varying width and center that defines a linear mapping within this window is the de-facto standard in examining image data. While this is straightforward to use and allows for fast implementations, its results are limited and often require lots of manual interaction. Depending on the focus of the diagnosis this can become a tedious manual process.

In the field of computer graphics and computer vision, high dynamic range images are a common approach to represent simulated or captured illumination without introducing errors due to band-limiting the signal. These images are stored in data types of a larger numerical range and/or precision, typically 16 or 32 bit integers or floating point. However, displaying such HDR data by mapping the input data linearly to the output device’s range is not sufficient, as only the bright areas would be visible (see figure 3). To overcome this issue, a lot of research has been done in both the computer graphics and computer vision community. With such a foundation it is quite obvious to apply those techniques to medical data in order to improve the visualization. As each modality in medical image acquisition requires a different interpretation – depending on the protocol even single acquisitions – this topic has to be discussed more thoroughly to avoid misinterpreting the results.

The remainder of the paper is structured as follows: In the next section we will discuss existing approaches, focusing on both GPU implementations and tone mapping algorithms. In section 3 we will present our system by outlining the environment with details about our abstract representation of procedures, data handling, the modular concept for shader implementation, etc. Also, we will discuss different tone mapping algorithms and already look at their possible application to different modalities. The results of applying the techniques to medical volume data will be described in section 4 with a discussion of both performance and visual results. We will conclude and propose directions of further investigation in the last section.

2 Related work

In this section we will discuss existing approaches and techniques to outline the context of this paper. Therefore, graphics hardware developments are summarized, followed by a more closer look on tone mapping methods.

2.1 General Purpose GPU

Using commodity graphics hardware for non-graphic application has become an important research topic for several years. This is mainly due to the rapid advances in hardware and programming capabilities, allowing much more flexible programs (so-called “shaders”) with unprecedented computing performance. In their survey, Owens et al. give an exhaustive overview of today’s technology and its use in all kinds of applications [Owens et al. 2007]. Quite recently, graphics hardware companies have developed dedicated hardware and APIs for non-graphical applications [NVI 2007], [Hensley 2007] to overcome the graphics-only programming concepts.

Although this trend is promising and will continue at an increase rate clearly outperforming the development of current (multi-core) CPUs, as depicted in figure 2, applications often suffer from limited bandwidth for the data transfer to and especially from the graphics memory. However, Langs et al. [Langs and Biedermann 2007] have shown recently that for example advanced filtering of large video data can still be performed several orders of magnitudes faster than CPU-only implementations.

2.2 GPU programming

Aside from the hardware architecture itself, programming such devices has also become more important. High-level languages like Cg [Fernando and Kilgard 2003] or GLSL [Rost 2005] have become the de-facto standard for GPU programming. Although this already enables more flexible and rapid development of shaders, approaches aiming at another abstraction layer have been proposed. McCool and others [McCool et al. 2002], [McCool et al. 2004] have developed “meta-languages” to integrate shader functionality directly into the application code, with concepts like shader algebra and a separation of frontend and backends for different platforms. In addition, Buck et al. [Buck et al. 2004] have developed a streaming-oriented extension that regards operations as kernels applied to data, with according representations for the data, parameters, etc. on the graphics hardware.

Yet another approach is the representation of shader functionality as directed graph, going back to concepts like Cook’s shade trees [Cook 1984] that have been used in almost all systems for computer generated images – even in the pre-GPU era. Using this method, complex shaders can be constructed from rather simple and optionally shared components. Related approaches from Abram et al. [Abram and Whitted 1990] or, more recently, McGuire et al. [McGuire et al. 2006] regard shaders as building blocks, emphasizing the specific features of shader programs like different types or input parameters.

2.3 High dynamic range imaging

Mainly initiated by the seminal work of Debevec et al. [Debevec and Malik 1997], a lot of research on high dynamic range imaging, i.e., image data with luminance values of multiple orders of magnitude has been done since then. However, such HDR data is usually displayed on devices with a much lower dynamic range. Although first prototypes of HDR displays are available, so-called tone mapping (or tone reproduction) algorithms still need to be applied to visualize the data adequately. For reference, Reinhard et al. review

existing approaches and address the whole "pipeline" of high dynamic range imaging in their book [Reinhard et al. 2005]. Basically, tone mapping operators can be categorized into *global* or *local*, some of them with an additional time-dependency. Global operators define some function that maps equal input values to equal output values, thus being computationally inexpensive. These functions can be as simple as a linear function, whereas more advanced operators are more complex and usually incorporate a logarithmic term and other properties of human perception. The results of such tone mapping functions are illustrated in figure 3 and show the already high potential of global operators.



Figure 3: Different global operators applied to an HDR scene: linear (a), logarithmic (b), and exponential scaling (c), Reinhard's operator (d) (Images courtesy [Reinhard and Devlin 2005])

In contrast to the "constant" mapping, local operators are able to adapt to changes by considering the neighborhood of each value. While this approach is very powerful, applying such locally varying operators to medical data needs some discussion. Lately, Bartz et al. [Bartz et al. 2006] have proposed to use tone mapping operators on medical data for improved visual representation. Their algorithm is based on the local operator from Reinhard et al. [Reinhard et al. 2002] with an additional variant for regarding a three-dimensional neighborhood, as is advantageous for most volumetric data. Due to the local nature of the algorithm, the processing time of several seconds for moderately sized volumes is obviously not interactive. They also use only datasets of traditional modalities as CT and MRI, the former providing even a constant mapping (i.e., Hounsfield units) of the measured values. More involved data like PET or non-scalar MRI from functional or diffusion-tensor imaging, that cannot be tone mapped locally without introducing errors has not been considered or left as future work.

Finally, several work has been done to implement the techniques on graphics hardware, as image processing is a particularly suitable application for GPUs. In 2003 already, Goodnight et al. [Goodnight et al. 2003] have successfully realized a time-dependent (i.e., adapting over time, thus mimicking the human visual system) tone mapping system for color images. Due to the much lower performance and more restricted programming of graphics hardware back then, they have not been able to achieve real-time frame rates. Vollrath et al. [Vollrath et al. 2005] have proposed a generic, real-time capable framework to implement the volume rendering pipeline on the graphics hardware. Their system implements Reinhard's tone mapping operator successfully, but they do not discuss it in the context of medical data. Another representative work is that of Yuan et al. [Yuan et al. 2006] who have developed a sophisticated system for visualizing large high dynamic range data volumes at interactive rates. The 3D and 4D data mainly from numerical simulations, geosciences, etc. is processed with especially handling precision issues. However, they also have not investigated in the applicability of their methods on medical data sets.

3 Technical overview

In this section we will explicate the technical details of our contribution. After describing our programming framework and discussing some of its key features related to GPU-accelerated volume

processing and visualization, we will outline the tone mapping algorithms we have used for our experiments. In addition, we will discuss the different modalities of medical data with respect to tone mapping operators.

3.1 The GPU-based system "Cascada"

Starting with a project on segmenting medical data, we have developed a system for (GP)GPU programming called "Cascada". This cross-platform framework focuses on processing (medical) volume data by applying modular algorithms implemented as shader programs, i.e., on graphics hardware. Using the GPU for such computation is highly motivated by the rapid performance increases in contrast to CPUs, as shown in the introduction of this paper. However, the raw performance of the graphics hardware does not include all other steps like data transfer, shader handling etc. To this end, our framework is also used as platform for comparing different types of algorithms aiming at general categories where the GPU is preferable or where the overhead might outweigh the performance gain.

3.1.1 Hierarchical representation of functionality

In order to abstract from the graphics programming details, algorithms are represented hierarchically (see figure 4): so-called sequences encapsulate procedures that can range from simple thresholding to more complex operations like region growing. Sequences in turn consist of multiple passes, i.e., drawing geometry with assigned shader programs, usually to offscreen buffers to allow for advanced processing. These passes can resemble single operations or multiply run passes, controlled by a fixed number of iterations or until some condition is met (e.g., region growing has converged). The output of one pass is then used as input to the subsequent pass by setting the texture parameters accordingly. Finally, shader programs resemble objects containing a GLSL vertex and fragment program, together with an automatic infrastructure for handling uniform parameters on both the CPU and GPU efficiently, concatenation of shaders, etc.¹

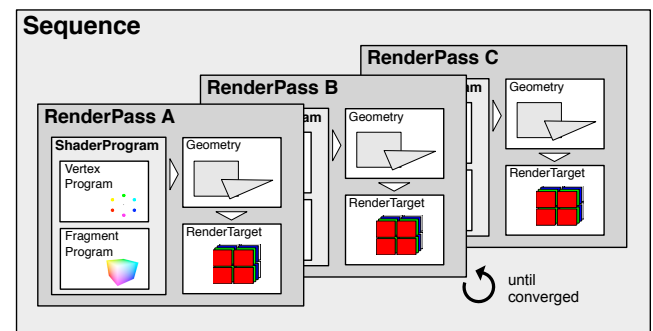


Figure 4: Hierarchy of rendering components in Cascada. Sequences of render passes consist of shader programs which are applied to geometry. The passes can be controlled by conditions and render into targets used as inputs for subsequent passes, read from for CPU operations, etc.

¹It should also be noted that by utilizing the composite pattern, sequences can contain sequences themselves, thus allowing for even more flexible designs.

3.1.2 Internal data handling

The data itself is represented as volumes packed into RGBA-tuples, thus allowing for direct rendering into the volume and exploiting the SIMD architecture of GPUs. Therefore, four successive slices of the scalar volume (or four DICOM slices) are combined into one RGBA slice. For both backward compatibility and better performance, the system uses aside from native 3D textures a two-dimensional representation of the volumetric data as introduced by Harris et al. [Harris et al. 2003]. As shown in [Langs and Biedermann 2007], the time for accessing the flat-3D texture by converting the 3D texture coordinates into the 2D address is even less than direct 3D access. This scheme is depicted in figure 5.

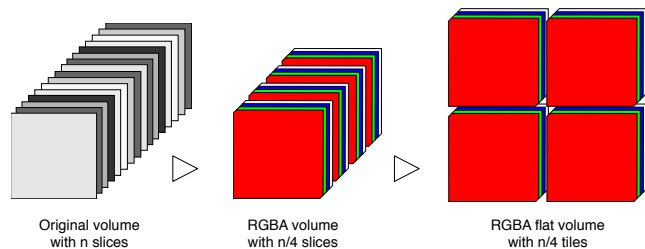


Figure 5: Representation of volumes in Cascada: The scalar volume is “compressed” into RGBA channels and finally spread into a 2D RGBA texture to be also used as render target.

Cascada also provides CPU equivalents of the aforementioned sequences and therefore allows for transferring data between graphics and main memory. To this end, Cascada uses a “lazy evaluation” policy to avoid unnecessary bus communication for the complete volume. As the CPU versions of the algorithms are not optimized to leverage parallelism with SIMD units and instructions or utilizing multiple cores, they serve as reference with respect to the results rather than performance.

3.1.3 Visualization and interaction

At the level of render passes in our hierarchical representation, the system does not distinguish between processing and visualization steps. The only difference is the target of the pass/sequence (i.e., offscreen/onscreen buffer) which can be changed during runtime, if needed. Therefore, “Cascada” implicitly allows for displaying the results of algorithms while they are computed, with a negligible performance overhead on modern graphics hardware. Currently, our system provides several volume visualization modes:

- Standard multi-planar reformation display with slices movable in the main directions
- GPU ray casting with different compositing modes (X-ray, maximum intensity projection, isosurface rendering)
- Direct volume rendering with basic transfer function support
- Integration of geometric primitives (glyphs, lines, color coding) in volume rendering (work in progress)

Together with the support for additional interaction devices (3D mouse, haptics devices) it is thereby possible to literally interact with the algorithms: “drawing” parameters for ray casting or controlling iterative algorithms are two examples for the advantage of computing directly on the graphics hardware.

3.1.4 Performance examples

In order to provide some measure for the efficiency of our framework, we have compared its processing performance with MeVis-

Lab [MeVis 2007], a widely used software system for efficient (medical) data processing and visualization. Most of MeVisLab’s core is based on a sophisticated image processing library with high performance even for very large data sets. Additionally, the modular system of the components in combination with the extremely powerful, graph-oriented frontend enable the user to develop applications rapidly. However, the system does not utilize the graphics hardware except for visualization purposes and graphical shader programs.

To give some idea of the performance relations between Cascada and CPU-only implementations, we have set up some scenarios of increasing computational complexity. We will provide both the times for the pure GPU pass, and the time needed for data transfer to and from the video memory for better comparison. Our system has been an Intel Core2Duo (2.4 GHz) with 2 GB RAM and an Nvidia Geforce 8800 GTS with 640 MB RAM, running on Windows XP. For MeVisLab we have used the latest SDK version (1.5.1) available from the website with default settings. We have applied the algorithms in both applications on the same datasets: a $512 \times 512 \times 223$ CT scan loaded from DICOM files (first value), and an MRI scan $256 \times 256 \times 256$ as raw file (second value).

Operation	MeVisLab	Cascada ¹	Cascada ²
Binary	0.73 / 0.21	1.9 / 0.55	0.038 / 0.011
Gradient 2D	10.1 / 2.9	2.6 / 0.7	0.06 / 0.017
Gradient 3D	14.9 / 3.9	2.6 / 0.72	0.059 / 0.018
Gauss 2D	1.36 / 0.37	2.48 / 0.73	0.061 / 0.017
Gauss 3D	3.65 / 1.01	2.59 / 0.78	0.09 / 0.026

Table 1: Average computation times (in seconds) for applying operations of different complexity on the two data sets. ¹ Timing including data transfer to and from the GPU; ² Time for image operation only, but including shader and handling.

The results show a clear advantage of the GPU version over the CPU implementation, with a speedup factor of up to 250. However, the impact of the data transfer on the performance outweighs the gain for simple operations (binary threshold) or filters with small neighborhood (Gauss 2D). These timing give only some coarse impression of the results, with Cascada being not fully optimized with respect to data transfer yet.

In another test run we have compared a sequence consisting of algorithms of different complexity (incl. iterative operations) to evaluate the relevance of data transfer, leaving the GPU (including all overhead) at least one order of magnitude faster than the software implementation. Therefore, our proposition of utilizing the graphics hardware for image processing operations is highly advantageous has been confirmed, even with the additional rendering infrastructure. However, not all operations have been considered yet, especially where the GPU suffers from its architectural limitations. We are still investigating in this to allow for a categorization of different algorithm types and be able to decide whether a GPU or CPU implementation is more suitable over the other – even at run time with our system already supporting the mixed use of both platforms.

3.2 Medical data

For medical diagnostics and, of course, depending on the current problem, there are lots of different imaging modalities available, with the majority being tomographic data nowadays. Dedicated hardware and software integrated in those systems allow for rapid reconstruction of the measured data resulting in slices of usually scalar values. These slices have certain geometric properties (e.g.,

thickness, distance, resolution) and can thus be regarded as a volume representing the examined anatomy. That is, we assume a regular grid of voxels with each voxel containing a value at a certain point in space. These values are usually scalar values, i.e., gray values within some range defined by the image acquisition system. Due to technical and computational reasons the current range is usually 16 bits, while not all bits have to be used; special DICOM tags specify the exact layout. The following table lists typically used bit depths and value ranges for the modalities discussed in this work. Aside from these properties the data differ in their interpreta-

Modality	Bit depth	Typical range
CT	12 (integer)	-1k...3k
MRI	10/12/16 (integer)	0...1-32k
fMRI	16 (integer/float)	-32k...32k
DTI	16/32 (float)	-32k...32k
PET	16 (integer/float)	0...32k

Table 2: Typical properties of data in common modalities

tion and thus need consideration for tone mapping algorithms. Computed tomography data, for example, has a fixed relation between the value at some position within the volume and the subject’s density at that position. That is, the higher the measured volume the lower the radiographic density, and vice versa – a relation represented by the Hounsfield scale. In our case, transforming a CT signal I with some nonlinear global tone mapping function Φ would imply to interpret the data as if the Hounsfield function H had also been transformed, thus leading to the result I_H , that can be used for classification etc.:

$$I_H = H(I) \iff \Phi(I_H) = \Phi(H(I)) \quad (1)$$

In principle, only positron emission tomography (PET) also allows for a proportional relation between the measured intensity and the corresponding data (usually metabolic activity). All other modalities discussed here do not provide a direct mapping between the values and some scale, as even the values of different acquisitions can have different meaning. Things become even more complicated for diffusion tensor imaging (DTI) data, where not only scalar values per sample are to be interpreted: at least six entries for the partial derivatives in all directions are computed. Usually an additional channel specifies a “confidence” for the data at the current position. Thus, any function for mapping the data needs to be defined over six dimensions which is closely related to the topic of transfer functions that will be discussed further in section 4.

Based on these considerations it is clear that only global tone mapping provides a reasonable chance to maintain an interpretation of the data. Local operators, i.e., a non-uniform transformation of input data depending on local neighborhood etc., would change it in a way that interpreting or processing the data is practically impossible. Therefore, we concentrate on global operators that consider properties of the whole volume and can be adjusted by parameters similar to the current windowing technique. Although implementing such algorithms on graphics hardware faces the same computational complexity as CPU implementations, global tone mapping algorithms are especially suitable for GPUs due to their inherent parallelism. Thus, we expect a performance in the order of magnitude as shown before in table 1.

3.3 Tone mapping

As already described in sections 1 and 2, tone mapping is an important procedure in high dynamic range imaging. As explicated in the preceding section, the data in diagnostic visualizations is manually compressed by specifying a window of controllable width and

center position. Although this linear interpolation between the minimum and maximum of the output device can also be regarded as tone mapping function, it cannot achieve proper, data-driven results automatically. In addition, the mapping will result in an information loss if the input range is larger than the output range – which is usually the case for medical data (see 3.2). In order to investigate the results of applying tone mapping algorithms from high dynamic range imaging to medical volume data while not sacrificing the real-time visualization of the system, we have decided to use global operators.

All of the following algorithms require some information about the data itself and some global user definable parameter. First, the so-called *background intensity* I_{avg} has to be estimated. Instead of simple averaging the intensities by computing the arithmetic mean, we have used the geometric average as suggested by Reinhard [Reinhard et al. 2005]:

$$I_{avg} = \exp\left(\frac{1}{N} \sum_{i=1}^N \log(I_i + \varepsilon)\right) \quad (2)$$

In addition, the unitless parameter α is known as the *key* and represents the overall light level of the data in an interval of $[0...1]$. Let L be the luminance if the input data, then α can be estimated according to [Reinhard et al. 2005] by: ²

$$f = \frac{2 \log L_{avg} - \log L_{min} - \log L_{max}}{\log L_{max} - \log L_{min}}, \quad \alpha = 0.18 \cdot 4^f \quad (3)$$

While α can also be controlled by the user, providing some reasonable default value is usually preferable. Some of the following algorithms that we have implemented introduce further parameters that will be discussed with the according description.

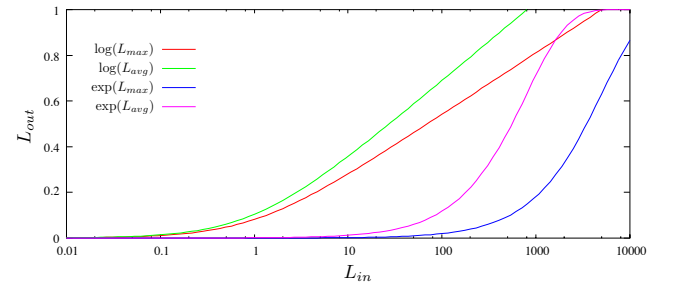


Figure 6: Graphs for the default logarithmic/exponential tone mapping curves with varying parameters.

3.3.1 Logarithmic and exponential scaling

Using the logarithm or an exponential mapping function is quite straightforward and has its background in Weber-Fechner’s law considering the relationship between measured and perceived stimuli. However, these approaches only achieve reasonable results for medium ranged images, i.e., in our case CT or low-valued MRI data. First the logarithmic function: ³

$$L_o(\vec{x}) = \frac{\log(1 + L_i(\vec{x}))}{\log(1 + L_{i_{max}})} \quad (4)$$

²Note that the logarithms used in the following equations are assumed to be to the base of 10, if not stated otherwise.

³For the equations in the following subsections we use the luminance function’s input parameter as n -dimensional vector \vec{x} , where n denotes the dimension of the data ($n = 3$ for volumetric scalar data)

The exponential function in (5) maps input luminances to output luminances, where input values close to zero are mapped to zero, infinitely bright values are mapped to 1.0. This implies a renormalization because the output will never cover the full range available. Reinhard [Reinhard et al. 2005] reports that exchanging $L_{i_{max}}$ with $L_{i_{avg}}$ and vice versa yields a different effect, also shown in the corresponding graph in figure 6.

$$L_o(\vec{x}) = 1 - \exp\left(-\frac{L_i(\vec{x})}{L_{i_{avg}}}\right) \quad (5)$$

3.3.2 Extended logarithmic scaling

Following the logarithmic behaviour of the human visual system, Drago et al. [Drago et al. 2003] have further investigated in improvements of logarithmic functions. They proposed to adjust the logarithm's base with the input value and thus achieve a wider range of values to be reasonably mapped. As can be seen in the second term of the following equation, this base is interpolated within the range of 2 and 10:

$$L_o(\vec{x}) = \frac{L_{o_{max}} \cdot 0.01}{\log(1 + L_{i_{max}})} \cdot \frac{\log(1 + L_i(\vec{x}))^{\frac{\log(p)}{\log(0.5)}}}{\log\left(2 + 8 \left(\frac{L_i(\vec{x})}{L_{i_{max}}}\right)^{\frac{\log(p)}{\log(0.5)}}\right)} \quad (6)$$

There are two parameters that can be specified by the user: the bias p controls the contrast, with larger values reducing the contrast. Also, the maximum output luminance $L_{o_{max}}$ (in cd/m^2) can be set, with a default value of 100.

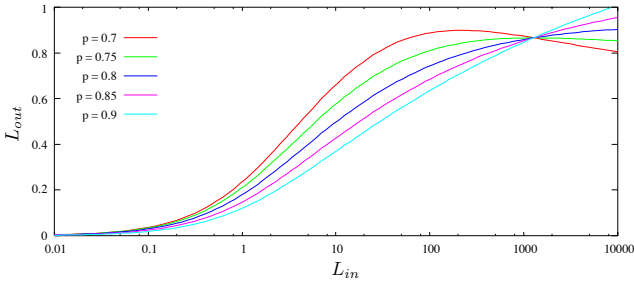


Figure 7: Graphs for the tone mapping after [Drago et al. 2003] with varying parameters.

3.3.3 Photoreceptor Model

While assuming that the human visual systems is of logarithmic nature is basically correct, Reinhard et al. [Reinhard and Devlin 2005] propose that this holds only for a certain range of values. As logarithms produce negative values and have no upper bound, they need to be modified for an adequate model. This leads to the following relation:

$$L_o(\vec{x}) = \frac{L_i(\vec{x})}{L_i(\vec{x}) + \sigma(L_{i_a}(\vec{x}))} \quad (7)$$

$$\sigma(L_{i_a}(\vec{x})) = (fL_{i_a}(\vec{x}))^m \quad (8)$$

$$k = \frac{L_{i_{max}} - L_{i_{avg}}}{L_{i_{max}} - L_{i_{min}}}, \quad m = 0.3 + 0.7k^{1.4} \quad (9)$$

In the equations (7) and (8), the term L_{i_a} denotes the adaptation level and can be set to $L_{i_{avg}}$ in our case, as no temporal or chromatic adaptation is needed. In addition, σ is often regarded as semisaturation constant. The following graphs illustrate the influence of the parameters k and f on the result:

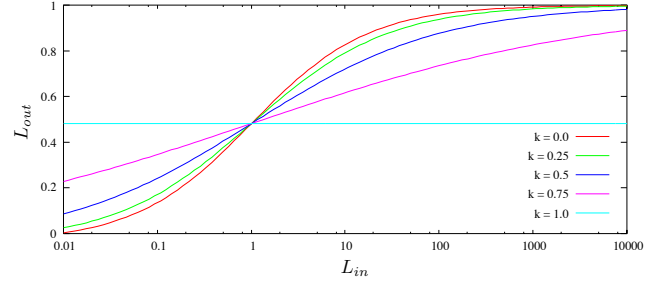
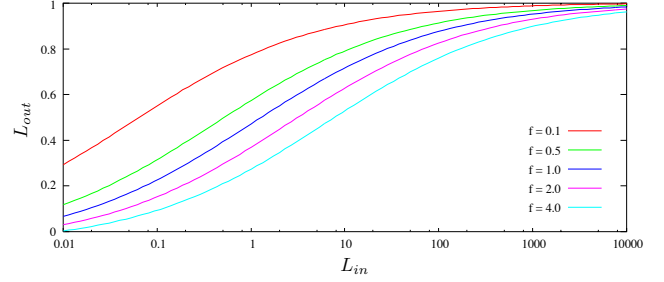


Figure 8: Graphs for the tone mapping curves according to [Reinhard and Devlin 2005] with varying luminance parameter (top) and key parameter (bottom).

4 Results and Discussion

We have used different kinds of datasets, focusing on modalities with a large dynamical range. Therefore, the data sets used in this paper are CT and PET data, with properties specified in the following table. As can be seen from the images, the rendering is in our case currently without using transfer functions. This is mainly due to the different ways how transfer functions can be used in the context of high dynamic range rendering and is subject to further investigation.

Data set	Resolution	Bit depth	min/max
CT backpack	$512 \times 512 \times 373$	12 (int)	0 / 4072
PET body	$168 \times 168 \times 715$	16 (int)	2 / 32767
PET head	$128 \times 128 \times 83$	16 (int)	0 / 32767

Table 3: Properties of the data sets used in this paper

4.1 Performance

As expected, the higher the dynamic range of values, the larger the improvement over linear mapping. MRI data sets, for example, usually provide values up to approximately 1k and thus do not benefit much from tone mapping. As can be seen in figure 9, the implemented algorithms result in quite different renderings, with Drago's algorithm being a good candidate also for other modalities. In addition, with only one parameter (see 3.3.2) this operator is very easy to use.

The algorithms have been implemented directly within the displaying shader programs. Without exhaustive optimization, the performance overhead is negligible for all renderings: even ray casting of the whole volume maintains real time performance on current commodity hardware. Adjusting the user parameters as well as the statistical data of the volume data is done via uniform variables that are transferred to the graphics hardware only when updated and thus do not impose a performance penalty.

4.2 Usability

In comparison to the standard windowing scheme, only little interaction is needed with our approach. While this automatic initialization of the visualization parameters is generally considered positive, medical staff emphasize to still be able to manually focus on specific ranges. The parameters for the tone mapping algorithms, however, are hard to relate with the traditional approach. Therefore, we propose to combine the two techniques by approximating the automatically determined tone mapping function linearly with adequate windowing parameters.

Also, for registration purposes the visualization of the whole range of values is considered to be useful. The more detailed structures in the tone mapped data (as in figure 1, for example) provide a better visual guidance when specifying landmarks in two modalities with different value ranges, e.g. PET and CT or MRI data.

5 Conclusion and future work

In this paper, we have shown that using the GPU is advantageous for improving the visualization of volume data of higher dynamic range than the output device. After reviewing several approaches to compress high dynamic range data, we have discussed their applicability to medical data. Global operators offer a good way of transforming the data while being computationally less expensive and inherently amenable to parallel architectures. On the other hand, local operators would introduce an uncertainty into the visual representation impeding diagnostic interpretation. Therefore, we have implemented some representative global operators in our GPU framework and discussed how to extend the method in different ways.

As the results are promising, we would like to further investigate in this topic. First, applying such operators to other, more specific image modalities like fMRI/DTI has not been addressed in detail yet. Especially the growing importance of functional imaging with data not simply corresponding to scalar properties such as luminance or density will lead to more complex algorithms. With the real time performance at hand, visual approaches such as rendering different settings as preview (so-called "design galleries") would improve both the handling of the algorithms' parameters and multiple dimensions.

Another direction of research is the role of the transfer function in the context of high dynamic range imaging. Similar to pre-/post-classification in volume rendering, one can use the (HDR) transfer function with the original data and apply tone mapping procedures afterwards. Alternatively, the transfer function can be accessed by (low dynamic) values of the volume that has been tone mapped before. The implications are subject to further research, as well as its applicability to different modalities, especially for non-scalar types.

Acknowledgements

We would like to thank Guido Lorenz for his great support in the development of our framework, as well as all the students helping to extend Cascada's possibilities. Also thanks to Cornela Massin and Stephan Palmer for inspiring parts of this work by their theses, and our co-operation partners at the hospitals for discussion, collaboration and providing data sets. Additional data sets are courtesy of OsiriX's sample image website [Rosset et al. 2007]. The author would also like to thank many people at the Centre for image analysis of Uppsala University for providing a great working environment during the first development stages of Cascada. Finally, we would like to thank the reviewers for their feedback to improve this paper.

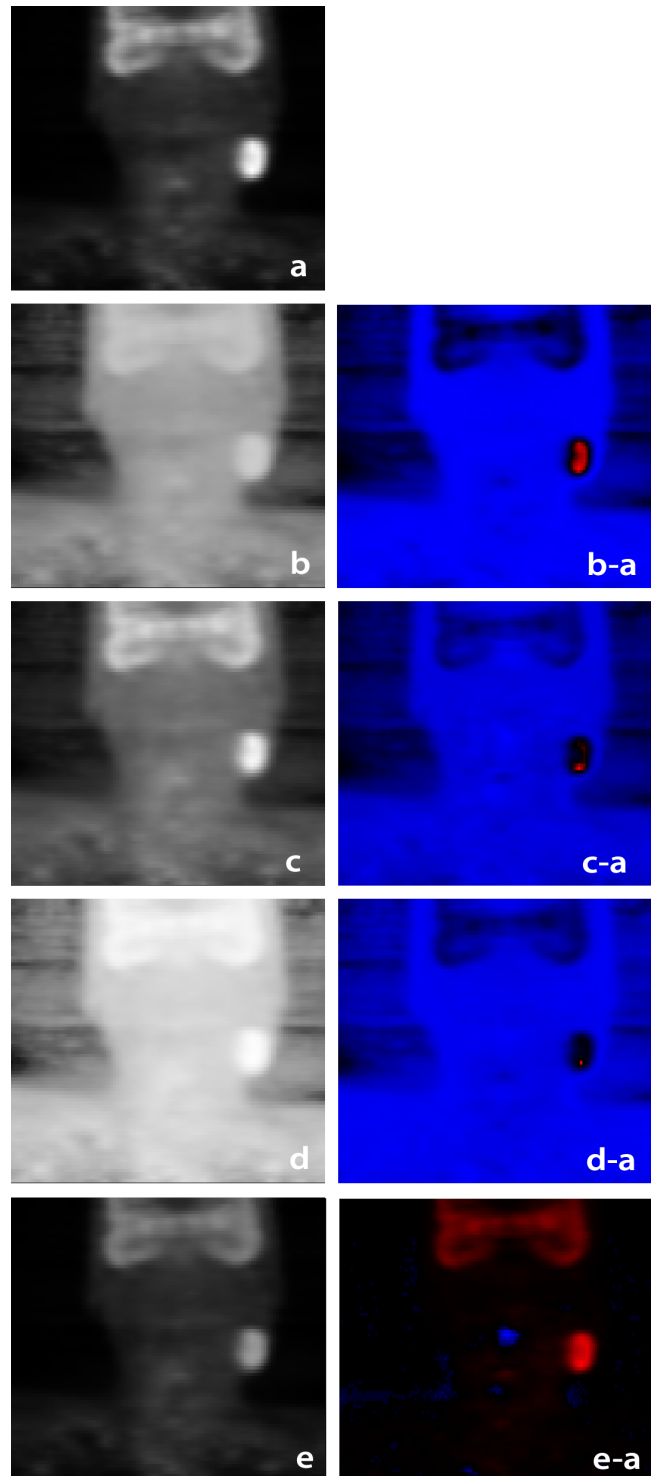


Figure 9: Overview of all the tone mapping algorithms implemented, applied to a human head/neck PET scan (coronal view). The linear mapping (a) is compared to the operators by Reinhard (b), Drago (c) and the logarithmic (d) and exponential (e) methods. The right column shows the subtraction of the linear mapping, with blue colors denoting positive, red colors denoting negative values.

References

- ABRAM, G. D., AND WHITTED, T. 1990. Building Block Shaders. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 283–288.
- BARTZ, D., SCHNAIDT, B., CERNIK, J., GAUCKLER, L., FISCHER, J., AND DEL RÍO, A. 2006. Volumetric High Dynamic Range Windowing for Better Data Representation. In *Afrigraph '06: Proceedings of the 4th international conference on Computer graphics, virtual reality, visualisation and interaction in Africa*, ACM, New York, NY, USA, 137–144.
- BUCK, I., FOLEY, T., HORN, D., SUGERMAN, J., FATAHALIAN, K., HOUSTON, M., AND HANRAHAN, P. 2004. Brook for GPUs: stream computing on graphics hardware. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, ACM, New York, NY, USA, 777–786.
- COOK, R. L. 1984. Shade Trees. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 223–231.
- DEBEVEC, P. E., AND MALIK, J. 1997. Recovering high dynamic range radiance maps from photographs. In *ACM SIGGRAPH Conference Proceedings*, 369–378.
- DRAGO, F., MYSZKOWSKI, K., ANNEN, T., AND CHIBA, N. 2003. Adaptive Logarithmic Mapping For Displaying High Contrast Scenes. In *Proc. of EUROGRAPHICS 2003*, Blackwell, Granada, Spain, P. Brunet and D. W. Fellner, Eds., vol. 22 of *Computer Graphics Forum*, 419–426.
- FERNANDO, R., AND KILGARD, M. J. 2003. *The Cg Tutorial – The Definite Guide to Programmable Real-Time Graphics*. Addison-Wesley Professional.
- GOODNIGHT, N., WANG, R., WOOLLEY, C., AND HUMPHREYS, G. 2003. Interactive Time-Dependent Tone Mapping Using Programmable Graphics Hardware. In *Proceedings of the 14th Eurographics workshop on Rendering*, Eurographics Association, 26–37.
- HARRIS, M. J., BAXTER, W. V., SCHEUERMANN, T., AND LASTRA, A. 2003. Simulation of Cloud Dynamics on Graphics Hardware. In *HWWS '03: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, 92–101.
- HENSLEY, J., 2007. Close to the metal. Course 24 - GPGPU: General-Purpose Computation on Graphics Hardware. ACM SIGGRAPH, San Diego, CA., August.
- LANGS, A., AND BIEDERMANN, M. 2007. Filtering Video Volumes Using the Graphics Hardware. In *SCIA*, Springer, B. K. Ersbøll and K. S. Pedersen, Eds., vol. 4522 of *Lecture Notes in Computer Science*, 878–887.
- MCCOOL, M. D., QIN, Z., AND POPA, T. S. 2002. Shader Metaprogramming. In *HWWS '02: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 57–68.
- MCCOOL, M., TOIT, S. D., POPA, T., CHAN, B., AND MOULE, K. 2004. Shader Algebra. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, ACM, New York, NY, USA, 787–795.
- MCGUIRE, M., STATHIS, G., PFISTER, H., AND KRISHNAMURTHI, S. 2006. Abstract Shade Trees. In *I3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games*, ACM, New York, NY, USA, 79–86.
- MEVIS, 2007. MeVisLab: A Development Environment for Medical Image Processing and Visualization. <http://www.mevislab.de>.
- NVIDIA CORPORATION. 2007. *NVIDIA CUDA compute unified device architecture programming guide*, Jan. <http://developer.nvidia.com/cuda>.
- OWENS, J. D., LUEBKE, D., GOVINDARAJU, N., HARRIS, M., KRÜGER, J., LEFOHN, A. E., AND PURCELL, T. J. 2007. A Survey of General-Purpose Computation on Graphics Hardware. *Computer Graphics Forum* 26, 1, 80–113.
- REINHARD, E., AND DEVLIN, K. 2005. Dynamic Range Reduction Inspired by Photoreceptor Physiology. *IEEE Transactions on Visualization and Computer Graphics* 11, 1, 13–24.
- REINHARD, E., STARK, M., SHIRLEY, P., AND FERWERDA, J. 2002. Photographic Tone Reproduction for Digital Images. *ACM Trans. Graph.* 21, 3, 267–276.
- REINHARD, E., WARD, G., PATTANAİK, S., AND DEBEVEC, P. E. 2005. *High Dynamic Range Imaging – Acquisition, Display, and Image-Based Lighting*. Morgan Kaufmann, November.
- ROSSET, A., ET AL., 2007. OsiriX Medical Imaging Software. <http://www.osirix-viewer.com>.
- ROST, R. J. 2005. *OpenGL(R) Shading Language (2nd Edition)*. Addison-Wesley Professional.
- VOLLRATH, J. E., WEISKOPF, D., AND ERTL, T. 2005. A Generic Software Framework for the GPU Volume Rendering Pipeline. In *Vision, Modeling, and Visualization VMV '05 Conference Proceedings*, 391–398.
- YUAN, X., NGUYEN, M. X., CHEN, B., AND PORTER, D. H. 2006. HDR VolVis: High Dynamic Range Volume Visualization. *IEEE Trans Vis Comput Graph* 12, 4, 433–45.