Eurographics Symposium on Rendering 2003
Per Christensen and Daniel Cohen-Or (Editors)

Delivering Interactivity to Complex Tone Mapping Operators

Alessandro Artusi,† Jiří Bittner, Michael Wimmer and Alexander Wilkie

Vienna University of Technology

Abstract

The accurate display of high dynamic range images requires the application of complex tone mapping operators. These operators are computationally costly, which prevents their usage in interactive applications. We propose a general framework that delivers interactive performance to an important subclass of tone mapping operators, namely global tone mapping operators. The proposed framework consists of four steps: sampling the input image, applying the tone mapping operator, fitting the point-sampled tone mapping curve, and reconstructing the tone mapping curve for all pixels of the input image. We show how to make use of recent graphics hardware while keeping the advantage of generality by performing tone mapping in software. We demonstrate the capabilities of our method by accelerating several common global tone mapping operators and integrating the operators in a real-time rendering application.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image GenerationDisplay algorithms; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and RealismColor, shading, shadowing, and texture

1. Introduction

The conversion from high dynamic range images to images suitable for display devices is known as tone mapping. The goal of tone mapping is to reproduce the overall impression of brightness, contrast, and colors of an image on a device with a restricted range of displayable colors[18].

Many tone mapping operators have been proposed in the last decade. An important subclass of these are global tone mapping operators. They consider each pixel in an image separately, and are therefore well suited for acceleration. Although these operators do not take into account the neighborhood of a pixel (like local tone mapping operators), they can still simulate many important effects related to the human visual system.

Some global operators evaluate just a simple function, but many of them perform complex computations for each image pixel[18, 22, 12, 4, 8]. While convincing results can be obtained with these complex global tone mapping operators, the high execution times of these algorithms prevent their usage in interactive applications.

3.7s 0.27s 0.04s

Figure 1: *(left) Image tone mapped using the operator of Pattanaik et al.; computational time 3.7s. (center) Result of a software implementation of the proposed technique using the same tone mapping operator (0.27s). (right) Result of the hardware implementation (0.04s).*

One possibility of delivering interactivity is implement-

† email:artusi@cg.tuwien.ac.at

ing a tone mapping operator directly in graphics hardware with support for floating point color representation and programmable pixel shaders. Due to the hardware constraints, this approach however imposes severe limitations on the operator: (1) it cannot perform arbitrary data manipulation, (2) its complexity is limited by the maximum number of instructions of a pixel shader.

We propose a novel acceleration framework that delivers interactivity to complex global tone mapping operators. This brings the user a qualitatively different understanding of the behavior of different operators as (he)she can study their response by interactive manipulation of the image or the parameters of the operator. The framework can be implemented either as a pure software acceleration technique, or it can perform some algorithmically simple but computationally costly operations in the graphics hardware. In the latter case, the proposed framework also provides the benefit of an efficient integration of the method into the flow of the graphics hardware rendering pipeline and permits to use complex global tone mapping operators in real-time rendering applications.

Unlike previous techniques that exploit graphics hardware for tone mapping, our method does not require modification of the rendering pipeline, as it works as a postprocess of the high dynamic range image. The framework allows an efficient subdivision of the workload between the CPU and the GPU. The GPU resolves the computationally simple but costly stages of the algorithm. The tone mapping operator is applied on the CPU, which allows to use existing implementations of tone mapping operators.

The work presented in this paper is novel in several aspects. First, according to our best knowledge it is the first general framework for acceleration of arbitrary global tone mapping operators. Second, parts of the framework can be implemented in the graphics hardware to improve efficiency of the algorithm. Third, the method seamlessly integrates into the flow of the hardware rendering pipeline. This allows applying realistic tone mapping in real-time applications using high dynamic range capabilities of recent graphics hardware.

The paper is organized as follows: Section 2 discusses the related work. Section 3 provides an overview of the proposed method. Section 4 discusses the different steps of the proposed framework. Section 5 discusses an implementation of the method on the graphics hardware. Section 6 presents results of the proposed technique. Finally, Section 7 concludes the paper.

2. Related Work

The concept of tone mapping has been introduced by Tumblin and Rushmeier[18], who proposed a tone reproduction operator that preserves the apparent brightness of scene features. Ward[20] described a tone reproduction operator which preserves the apparent contrast and visibility. A model that includes different aspects of adaptation was introduced by Ferwerda et al.[8]. Pattanaik et al.[11] developed a computational model of adaptation and spatial vision for realistic tone reproduction. A model that perceptually expands and enhances the perceived dynamic range was presented by Spencer et al.[16], and Nakamae et al.[9] presented a model in order to improve the realism of the modeled scene. Schlick[15] proposed a quantization technique for visualization of high dynamic range pictures. Chiu et al.[2] introduced a spatially nonuniform scaling function for high contrast images. Tumblin et al.[17] proposed two methods for displaying high contrast images. The LCIS algorithm proposed by Tumblin et al.[19] aims to preserve the contrast reduction. The model presented by Ward et al.[22] proposed a new histogram adjustment technique and considers glare, visual acuity, and color sensitivity. As a further step forward, a time-dependent visual adaptation model was introduced by Pattanaik et al.[12]. Fattal et al.[7] proposed a method capable of significant dynamic range compression while preserving fine image details. Reinhard et al.[13] adapt techniques used in photographic practice to create a robust tone mapping operator.

The methods discussed above concern accurate operators that attempt to reproduce individual visual effects at non-interactive or close-to-interactive speeds. We are not aware of a prior work which provides a general framework for tone mapping acceleration. Interactive tone mapping operators are typically tightly connected with the graphics hardware. Cohen et al.[3] proposed tone mapping operators suitable for hardware implementation. Due to hardware constraints, these operators are rather simple and do not consider effects like time dependency, chromatic adaptation, or the visual effects of the human visual system. Certain steps of our framework are related to acceleration techniques used by Scheel et al.[14], Durand and Dorsey[5, 6], and Ward et al.[22]. Scheel et al.[14] perform ray casting to obtain high dynamic range image samples and use the texture mapping hardware to apply tone mapping on vertices of a model with precomputed radiosity. In contrast to their method, our technique does not require ray casting and does not affect rendering by using specialized texturing mapping techniques. Durand and Dorsey[5] downsample the image to compute the adaptation luminance and use a lookup table to speed up their interactive time dependent tone mapping algorithm. In their recent work, Durand and Dorsey[6] also use downsampling and linear interpolation. Ward et al.[22] also downsample the image to extract histogram information which is used to set up parameters for their tone mapping algorithm. Our framework is more general since we suggest to pass the samples directly to the (arbitrary) tone mapping operator and interpolate its results. Additionally we show how to implement the technique on recent graphics hardware with a floating point color representation, without modifying the rendering pipeline.

3. Overview

The proposed framework consist of four steps: sampling, tone mapping, fitting, and reconstruction. As an input we take a high dynamic range image. The *sampling* algorithm produces a set of samples that form a compact representation of the luminance distribution in the image. The samples are passed to the *tone mapping* operator that assigns each sample a luminance in the color space of the display device. The result is processed by the *fitting* algorithm that finds interpolation coefficients for the point-sampled tone mapping curve. Finally the *reconstruction* algorithm applies the interpolated tone mapping curve on all pixels of the input image. The framework is depicted in Figure 2.

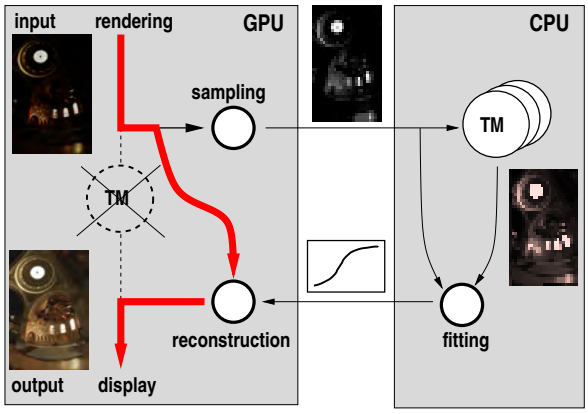

Figure 2: *Overview of the proposed framework. The sampling and the reconstruction is suitable for an implementation on the GPU, the tone mapping and the fitting are performed on the CPU (the main data flow is depicted in red).*

Figure 2 shows a possible subdivision of the computation between the graphics hardware and the general purpose processor. Sampling and reconstruction are relatively simple algorithms, but they are applied on all pixels of the input image. Tone mapping and fitting are usually more complicated algorithms, but they are applied on a relatively small number of samples. Thus using the suggested subdivision of the computation we exploit the raw computational power of graphics hardware while keeping the framework open for arbitrarily complex global tone mapping operators. The overhead of transferring data between the CPU and GPU is minimized by transferring only the sampled image and the coefficients established by the fitting algorithm. The main data flow takes place on the GPU.

4. The Framework

In this section we discuss the four main steps of our framework in more detail.

4.1. Sampling

The goal of the sampling algorithm is to compute a compact representation of the luminance distribution in the input image. The set of samples should provide an accurate representation of the histogram of the input image, while keeping the number of samples small. To avoid maintaining a high dynamic range histogram, we sample directly in the image domain. We have used two techniques that can be seen as representatives of two extremes—random sampling and downsampling—and one technique that combines the advantages of both—filtered random sampling.

Random sampling takes a specified number of samples from the input image. The samples are taken at image coordinates given by the Halton sequence. Alternatively if the gaze of the user can be predicted, the sampling density at the fovea can be increased[14].

Downsampling subdivides the image into n regions and computes an average luminance for each region, which corresponds to the application of a box filter on the samples. A similar technique was used by Scheel et al.[14] to compute a global adaptation luminance.

Filtered random sampling combines random sampling with downsampling in order to improve the ability of the algorithm to capture both high-frequency as well as low-frequency information. It applies a box filter on every pixel obtained by the random sampling technique.

See Figure 3 for an illustration of the sampling techniques.

4.2. Tone Mapping

The tone mapping operator is applied on the computed set of samples. We assume that the operator works only with the luminance component of CIE XYZ color space. To map a pixel of the input image to the device color space, the components of the input color are multiplied by the ratio of the device and world luminances. If a color transformation is required, it can be applied as a postprocess.

The tone mapping operator assigns each sample a low dynamic range luminance in the device color space. The set of samples and their mappings represent a 1D point-sampled *tone mapping curve*.

4.3. Fitting

The fitting algorithm aims to interpolate the point-sampled tone mapping curve in order to tone map all pixels of the input image. The fitting of the curve should accurately capture the point-sampled tone mapping curve, while keeping the computational costs low. We used piecewise linear interpolation because of its speed and simplicity. We also tried higher order interpolation, for example using natural cubic splines, but the slight increase in accuracy does not compensate the higher computational cost.

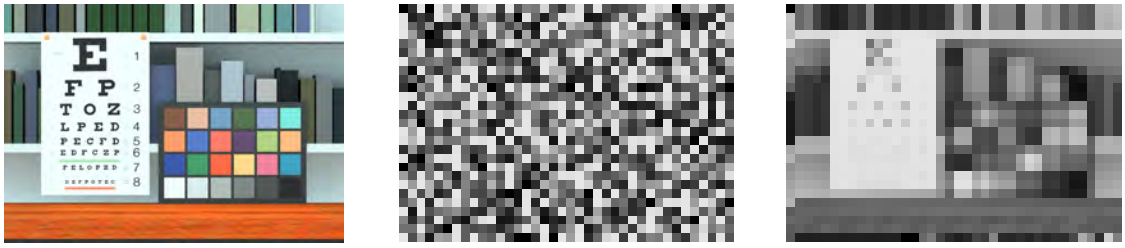

Figure 3: *The input image and the luminance samples. (a) The input image (1000x700 pixels). (b) Random sampling (1024 samples). (c) Downsampling (1024 samples).*

To compute the coefficients of the interpolation, we first sort the samples according to the world luminance. Then we compute the interpolation coefficients for each interval between two sample points.

4.4. Reconstruction

The reconstruction of the tone mapping curve proceeds as follows: Given the high dynamic range luminance L_w we locate an interval of the interpolated curve that covers L_w using a binary search. For the linear interpolation the device luminance L_d is extrapolated as:

$$L_d = w_1^i L_w + w_0^i, \qquad b_i \leq L_w < b_{i+1}, \tag{1}$$

where w_1^i, w_0^i are coefficients established for the i-th interval, and b_i, b_{i+1} are the boundaries of the interval.

4.4.1. Lookup Table

To accelerate the reconstruction algorithm, we can resample the interpolation coefficients and store them in a lookup table. Given a world luminance, the lookup table can be directly accessed to obtain the corresponding device luminance value. To improve the accuracy of the reconstruction we can use a logarithmic scale to capture the low dynamic part of the tone mapping curve more accurately.

A similar technique was used by Durand and Dorsey[5] and Scheel et al.[14]. These methods computed the samples using the tone mapping operator directly whereas our method resamples the interpolated tone mapping curve. The advantage of our method is that we can treat the tone mapping operator as a black box without altering the tone mapping curve. This is more general, since we do not have to know what parameters the tone mapping operator uses.

5. Hardware Implementation

We integrated the proposed framework to the rendering pipeline of a graphics hardware with the support for floating point per-pixel operations. In particular we used the NVIDIA GeForceFX card, which supports 32 bits per color component and pixel shaders (also called fragment programs) of up to 1024 instructions[10].

Following Figure 2, we ported two steps of the framework to the graphics hardware: the sampling and the reconstruction. The corresponding algorithms are implemented as fragment programs written in NVIDIA's Cg language[10]. In this section we discuss specific issues associated with the hardware implementation.

5.1. Sampling

Random sampling is implemented by dependent texture lookups. We generate a small texture containing values of the Halton sequence. The size of the texture is given by the number of the desired samples. This texture is rendered into the frame buffer as follows: for each pixel we use the corresponding texel as indices to the input image (treated as a secondary texture) and the pixel is set to the color of the addressed pixel. The fragment program for random sampling consists of 5 assembly instructions.

Downsampling is implemented by subsequent rendering of a textured quad of smaller resolution. Initially we use the input image as a texture and render it on a quad with $1/k$ of the resolution of the input image. The fragment program computes an average of k^2 texels for each pixel covered by the quad. If the input image consists of p pixels after n steps we obtain p/k^{2n} samples assuming the dimensions of the image are a power of k. The fragment program generated for $k = 2$ consists of 37 assembly instructions, for $k = 4$ of 133 instructions, and for $k = 10$ of 805 instructions.

Filtered random sampling is a simple extension to random sampling where a predefined image region around the input image pixel is averaged in the fragment program to obtain the sample value.

5.2. Tone Mapping and Fitting

After the sampling step the samples are transferred from the GPU to the CPU using a frame buffer read-back. The tone mapping operator is applied on the samples and the fitting is used to find the coefficients of the interpolated tone mapping curve as described in Section 4.3.

The coefficients obtained by the fitting algorithm are sent

to the graphics card as a texture. Each texel of the texture represents coefficients of the interpolation for one interval between the samples. The coefficients are encoded as color components of the texel.

5.3. Reconstruction

The lookup table for reconstruction is constructed by "rendering" the coefficients established by the fitting algorithm into the frame buffer. The coefficients of an interval are rendered as a narrow horizontal textured quad. The vertical limits of the quad are determined by the world luminance of the sample points that bound the associated interval. The texture of the quad consists of the interval coefficients. Each column of the frame buffer then represents a fixed size interval of world luminance. Given a luminance value, the reconstruction algorithm can then directly access the lookup table without the necessity of performing a binary search. The corresponding fragment program consists of only 10 assembly instructions including the reconstruction using linear interpolation.

6. Results

We implemented both the software and the hardware variant of the proposed framework. The software implementation was evaluated on a PC equipped with Athlon 800MHz running Linux. The hardware implementation was tested on a PC with Athlon 1.8GHz, and an NVIDIA GeForceFX graphics card. We used the following images in our tests: the Memorial church [Devebec and Malik 97] (512x768 pixels), the Hotel room [Crone, Fawler and Kerrigan 97] (3000x1950 pixels), and the Aeroport [Ehrlich, Ward 97] (1024x705 pixels)[21]. The images are depicted in Figures 1, Figure 5, and Figure 6.

6.1. Software Implementation

We conducted a series of tests to capture the behavior of different parts of the framework in dependence on the type of the sampling algorithm, the number of samples, and the type of interpolation. For the first series of tests, we used the tone mapping operator of Pattanaik et al.[12]. To evaluate the time performance, we measured the time of the sampling, the tone mapping, the fitting, the reconstruction, and the total time. The results are summarized in Table 6.1.

Our framework provides a speedup of Pattanaik's operator in the range between 9.23 and 17.6. For example using 1000 samples, our method accelerates the operator 16.18 times on an image of 3000x1950 pixels. We can see that downsampling provides more accurate results than random sampling at the cost of a slight increase in computational time. As the number of samples increases, the accuracy of random sampling becomes comparable with downsampling. In order to reduce the increased of computational costs of downsampling, we tried the filtered random sampling approach. We have obtained good results using a filter window size of 3x3 and 2000 samples. In Figure **??** we can see that the results for filtered random sampling are comparable to downsampling, but the computational cost is at the level of random sampling as shown in Table 6.1.

For all measurements, the time of the reconstruction is the major component of the total computational time. This justifies the subdivision of the computation between the GPU and CPU depicted in Figure 2, which suggest to implement the reconstruction in the GPU.

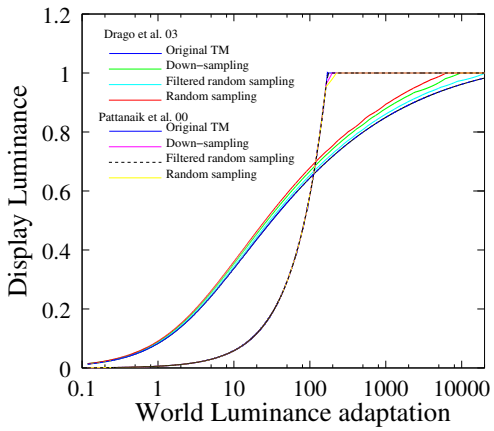

Figure 4: *Fitting accuracy of the framework using random sampling, downsampling with 1000 samples, and filtered random sampling with 2000 samples. The tone mapping operators used are Pattanaik et al. 00 and Drago et al. 03, both on the Memorial church image.*

We also evaluated our method using other tone mapping operators, namely the operator of Tumblin et al.[18] and the operator of Drago et al.[4]. For each input image we measured the results obtained by direct application of tone mapping and the results obtained by applying our technique using random sampling with 1000 samples and linear interpolation. Table 6.1 summarizes the results.

image	operator	time orig. [ms]	time accel. [ms]	speedup [-]
hotel	Pattanaik et al. 00	54852	3391	16.18
	Tumblin et al. 93	5657	3344	1.70
	Drago et al. 03	9359	3336	2.81
Aeroporto	Pattanaik et al. 00	6815	426	16.00
	Tumblin et al. 93	687	404	1.70
	Drago et al. 03	1154	400	2.89

Table 2: *Evaluation of the framework applying three tone mapping operators using random sampling (1000 samples) and linear interpolation on three different images.*

sampling	#samples	time					speedup
		samp. [ms]	TM [ms]	fit [ms]	reconst. [ms]	total [ms]	[-]
no acceleration				-	-	54852	1.0
halton	300	0.642	3.232	0.012	3118	3122	17.57
	1000	2.287	7.212	0.029	3381	3391	16.18
	2000	5.780	10.968	0.055	3423	3440	15.94
	3000	7.823	13.071	0.074	3469	3490	15.72
downs.	300	2024	3.604	0.014	3131	5159	10.63
	1000	2024	13.834	0.132	3522	5557	9.870
	2000	2051	20.075	0.162	3729	5800	9.457
	3000	2032	30.080	0.300	3879	5941	9.233
filt rand. 3x3	300	1.098	3.498	0.013	3222	3226	17.00
	1000	3.833	11.260	0.047	3610	3625	15.13
	2000	7.906	19.992	0.162	3813	3841	14.28
	3000	12.127	29.502	0.330	3963	4005	13.70

Table 1: *Evaluation of the framework in software using Pattanaik's operator on the hotel room.*

We can observe that all three complex global operators are sped up, and that the time performance of the framework is practically independent from the tone mapping operator used: The accelerated time in Table 6.1 is practically the same for all tested operators.

6.2. Hardware Implementation

For an evaluation of the hardware implementation of the method we used a simple OpenGL interactive application, which uses high dynamic range environment maps. The tests were conducted only for the tone mapping curve produced by the Ashikhmin operator. We measured the time performance of the method in dependence on the type of the sampling technique. We also measured the time of the actual tone mapping itself (executed on the CPU), and the total frame time. We also measured the time performances of a direct GPU implementation of the tone mapping operator used in these experiments. Note that we did not encounter any significant delays due to latency between CPU and GPU because only a very small image is read back from the graphics card into main memory. The results are summarized in Table 3.

The results show that the hardware implementation provides real-time performance to Ashikhmin's tone mapping operator. Our interactive test application was running at 65 to 80 frames per second. It is interesting to note that our framework can even compete in terms of speed against a direct hardware implementation of the quite simple Ashikhmin operator, albeit at reduced quality due to the sampling. The filtered random sampling technique does not cause a noticeable performance degradation as compared to random sampling, but provides better image quality.

Our original experiments with the hardware implementation revealed significant temporal aliasing in the form of flickering. This occurred especially when random sampling was used, because the input parameters of the used tone mapping algorithms, i.e., minimum, maximum or average, were not captured well by the small sample set used in the implementation. In order to reduce these artifacts, we reuse each sample for a certain number of frames as discussed by Scheel et al.[14]. Note that the measurements in Table 3 already reflect this improvement. Each sample was reused for 8 frames, which reduced the flickering by a significant amount. An interesting observation was that even the direct implementation of the Ashikhmin operator exhibited noticeable flickering in dynamic scenes.

samp.	samples	time		FPS [-]
		TM CPU [ms]	total [ms]	
random	1024	1.23	18.71	56
downs.	1024	1	21.42	48
filt. rand. 3x3	1024	1.01	18.87	56
direct GPU	-	-	19.76	51

Table 3: *Evaluation of the real-time application using Ashikhmin's tone mapping operator accelerated by our framework in hardware, and its direct implementation on the GPU. The image resolution is 512x512.*

7. Conclusion and Future Work

We have presented a framework that delivers interactive performance to complex global tone mapping operators. When integrated into the rendering pipeline, the proposed framework allows studying the response of complex global tone mapping operators interactively. It also enables to select in-

teractively the most appropriate tone mapping operator for a given application.

The structure of the framework is designed for easy integration into the rendering pipeline of recent graphics hardware. Unlike previous tone mapping techniques using graphics hardware, our method does not require modification of the rendering pipeline as it works as a postprocess on a high dynamic range image. We proposed an efficient subdivision of the workload between the CPU and the GPU. The tone mapping operator is applied on the CPU, which maintains the generality of the method. The GPU resolves the computationally simple but costly stages of the algorithm. An important feature of the framework is that the GPU implementation is simple and it does not need to be modified for an application with a different tone mapping operator.

The hardware implementation of our method shows a speedup of one order of magnitude compared to the pure software solution. This proves the potential of the proposed technique for interactive rendering applications.

The modularity of the proposed framework enables to concentrate on improving each of its parts independently. We currently work on improving the sampling algorithms in order to reduce temporal aliasing artifacts. Another interesting topic is an extension of the framework for the use with local tone mapping operators.

Acknowledgements

Many thanks to Katja Bühler for her helpfull comments. This research was supported by the Austrian Science Foundation (FWF) contract no. p-13876-INF, and by the European Comunity within the scope of the RealReflect project IST-2001-34744 "Realtime visualization of complex reflectance behavior in virtual prototyping".

References

1. Michael Ashikhmin. A tone mapping algorithm for high contrast images. In *Rendering Techniques '02 (Proceedings of the 13th Eurographics Workshop on Rendering)*, pages 145–156, June 26–28 2002.
2. K. Chiu, M. Herf, P. Shirley, S. Swamy, C. Wang, and K. Zimmerman. Spatially Nonuniform Scaling Functions for High Contrast Images. In *Proceedings of Graphics Interface '93*, pages 245–253, May 1993.
3. Jonathan Cohen, Chris Tchou, Tim Hawkins, and Paul Debevec. Real-Time high dynamic range texture mapping. In *Rendering Techniques '01 (Proceedings of the 12th Eurographics Workshop on Rendering)*, pages 313–320, 2001.
4. Frédéric Drago, Karol Myszkowski, Thomas Annen, and Norishige Chiba. Adaptive logarithmic mapping for displaying high contrast scenes. In *Computer Graphics Forum (Eurographics 2003 conference proceedings)*, volume 22, pages 000–000, 2003.
5. Fredo Durand and Julie Dorsey. Interactive tone mapping. In *Rendering Techniques '00 (Proceedings of the 11th Eurographics Workshop on Rendering)*, pages 219–230, 2000.
6. Frédo Durand and Julie Dorsey. Fast bilateral filtering for the display of high dynamic range image. In *Proceedings of SIGGRAPH 2002*, pages 257–265, 2002.
7. Raanan Fattal, Dani Lischinski, and Michael Werman. Gradient domain high dynamic range compression. In *Proceedings of SIGGRAPH 2002*, pages 249–256, 2002.
8. James A. Ferwerda, Sumanta N. Pattanaik, Peter Shirley, and Donald P. Greenberg. A model of visual adaptation for realistic image synthesis. In *Computer Graphics (SIGGRAPH '96 Proceedings)*, pages 249–258, August 1996.
9. Eihachiro Nakamae, Kazufumi Kaneda, Takashi Okamoto, and Tomoyuki Nishita. A lighting model aiming at drive simulators. In *Computer Graphics (SIGGRAPH '90 Proceedings)*, pages 395–404, August 1990.
10. NVIDIA. Graphics hardware specifications. `http://www.nvidia.com`, 2003.
11. Sumanta N. Pattanaik, James A. Ferwerda, Mark D. Fairchild, and Donald P. Greenberg. A multiscale model of adaptation and spatial vision for realistic image display. In *Computer Graphics (SIGGRAPH '98 Proceedings)*, pages 287–298, July 1998.
12. Sumanta N. Pattanaik, Jack Tumblin, Hector Yee, and Donald P. Greenberg. Time-dependent visual adaptation for fast realistic display. In *Computer Graphics (SIGGRAPH '00 Proceedings)*, pages 47–54. Addison Wesley, July 2000.
13. Erik Reinhard, Michael Stark, Peter Shirley, and Jim Ferwerda. Photographic tone reproduction for digital images. In *Proceedings of SIGGRAPH 2002*, pages 267–276, 2002.
14. A. Scheel, M. Stamminger, and H.-P. Seidel. Tone reproduction for interactive walkthroughs. In *Computer Graphics Forum (Proceedings of Eurographics 2000)*, volume 19(3), pages 301–312, 2000.
15. Christophe Schlick. Quantization techniques for the visualization of high dynamic range pictures. In *Photorealistic Rendering Techniques '94 (Proceedings of the 5th Eurographics Workshop on Rendering)*, Eurographics, pages 7–20, 1994.
16. Greg Spencer, Peter Shirley, Kurt Zimmerman, and Donald Greenberg. Physically-based glare effects for digital images. In *Computer Graphics (SIGGRAPH '95 Proceedings)*, pages 325–334, August 1995.
17. Jack Tumblin, Jessica K. Hodgins, and Brian K. Guenter. Two methods for display of high contrast images. In *ACM Transactions on Graphics*, volume 18 (1), pages 56–94, 1999.
18. Jack Tumblin and Holly E. Rushmeier. Tone reproduction for realistic images. *IEEE Computer Graphics and Applications*, 13(6):42–48, November 1993.
19. Jack Tumblin and Greg Turk. LCIS: A boundary hierarchy for detail-preserving contrast reduction. In *Computer Graphics (SIGGRAPH '99 Proceedings)*, pages 83–90, 1999.
20. Greg Ward. A contrast-based scalefactor for luminance display. In *Graphics Gems IV*, pages 415–421. Academic Press, 1994.

21. G. Ward Larson. High dynamic range images, 1997. http://positron.cs.berkeley.edu/ gwlarson/pixformat/tiffluvimg.html.

22. Gregory Ward Larson, Holly Rushmeier, and Christine Piatko. A Visibility Matching Tone Reproduction Operator for High Dynamic Range Scenes. *IEEE Transactions on Visualization and Computer Graphics*, 3(4):291–306, October 1997.

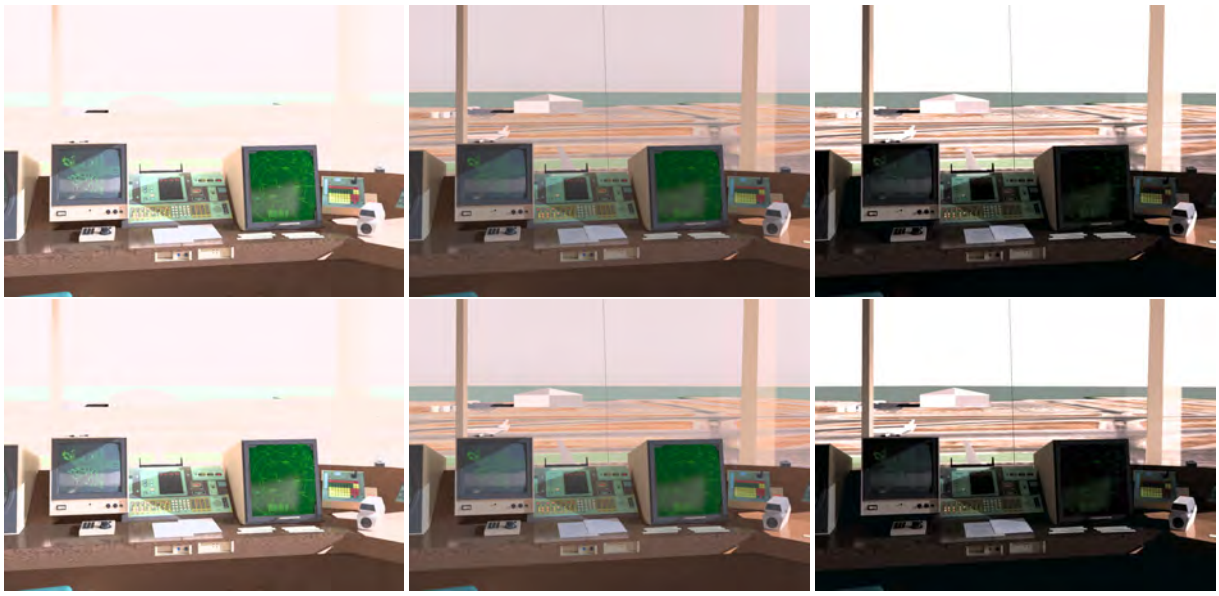

Figure 5: *The Aeroport [C. Ehrlich, G.Ward, 97] (1024x705); The top images are obtained using the original tone mapping operators, the bottom images are obtained by applying our framework. (Left) Pattanaik et al.*[12]*; (center) Drago et al.*[4]*; (right) Tumblin et al.*[18].

Figure 6: *Hotel room [Crone, Fawler and Kerrigan 97] (3000x1950); The top images are obtained using the original tone mapping operators, the bottom images are obtained by applying our framework. (Left) Pattanaik et al.*[12]*; (center) Drago et al.*[4]*; (right) Tumblin et al.*[18]