REAL-TIME HARDWARE BASED TONE REPRODUCTION

A Thesis

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Master of Science

by

John Crane Mollis

May 2004

ABSTRACT

The human visual system is exposed to a vast range of illumination conditions, far greater than any display device can reproduce, and its response to these conditions varies greatly. To create an immersive impression and accurate portrayal of a scene on a computer monitor requires complex modeling of visual response through tone reproduction algorithms and the simulation of adaptation effects in real-time. However, all current tone reproduction operators are off-line and address only a portion of the visual phenomena necessary for completeness, thereby limiting their applicability. A mostly unexplored problem is how perceptually accurate and full featured tone reproduction can be incorporated into interactive applications where visual effects will be dynamic and often very dramatic. Previous work in this area has been constrained with respect to the generality of tone reproduction models used, the scope of available input and the hardware output performance.

The aim of this thesis is two-fold. First, we create a real-time tone reproduction operator that includes as many phenomena as possible and is based upon psychophysical data. This requires a combination and extension of the best operators for predictive tone mapping and significant acceleration using current commodity graphics hardware. Care is taken to not restrict available input or compromise the predictive nature of the operator through artificial approximations. The result is a widely applicable, fast and comprehensive operator with applications in lighting engineering, architectural walkthroughs, flight and car simulators, entertainment and due to it's predictive nature, low vision simulation.

The second aim of this thesis is to use our perceptually based operator to construct a low vision simulation tool for evaluating real or simulated environments for suitability with older individuals.

BIOGRAPHICAL SKETCH

John Mollis was born in Flemington New Jersey on June 24th 1979 and grew up in the small town of Ringoes NJ. This was the most idyllic and wonderful time of his life. There was a train station with a real locomotive, corn fields, countryside and a summer paradise of fruit trees. No doubt such a stimulating environment shaped him profoundly. In fifth grade his family moved to Titusville, NJ where they still reside. After high school he applied to Cornell University and finished undergrad with a BA in Physics and then immediately joined the Program of Computer Graphics in pursuit of a Master of Science degree.

ACKNOWLEDGMENTS

I would like to thank Professor Donald P. Greenberg for giving me the chance to do research at the Program of Computer Graphics. Without your faith in me none of this work would be possible. Thanks must also go emphatically to Jim Ferwerda. I would not have been able to get through this without your guidance and I'm very grateful. You taught me a good deal about research and science. Many others at the PCG have also helped me along the way. Peggy Anderson and Linda Stephenson, it's great to have kind people to talk to when pacing the halls and to remind me of the forms I've completely forgotten about. Mary Armstrong, I appreciate the conversation and especially the cabbage and potatoes. To everyone else at the PCG, I'm grateful for your help and company, without which my time here would have been much tougher. Finally, I thank my family and friends for supporting me through the often rough time of graduate school. Bay, you were my life jacket and my sunlight in gloomy Ithaca.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

Tone reproduction in computer graphics, at the highest level of abstraction, is an attempt to match the visual response of a scene viewer with the visual response of a display viewer looking at a digitally sampled version of the actual scene presented on some display device (Figure 1.1). At this level display technology and image capture techniques define the accuracy with which a captured scene can be physically reproduced. If we had some way to exactly capture and reproduce the real radiance incident on an observer's eyes the problem would clearly be solved. However, the current state of display technology allows only limited fidelity with respect to maximum displayable luminance, dynamic range, color gamut and screen resolution. It is then the job of tone reproduction algorithms to fill the gap between display and scene observer, as best as possible, given current limitations of display hardware.

General tone reproduction, which artists have dealt with for centuries (with the advantage of being the human observer) was first noticed formally by photographers trying to faithfully reproduce impressions of real world scenes on photographic paper. Many practical methods were developed to address this problem [London98]. Some examples are the Zone System [Adams80, Adams81, Adams83] and dodging and burning. In computer graphics the development of radiosity and monte carlo path tracing [Glassner95] allowed physically based illumination simulations and thus high dynamic range output. More recently Debevec [Debevec97] pioneered high dynamic range digital photographs, adding to the need for good display mappings. Initial display mapping methods were ad hoc, for example taking the cube root of the

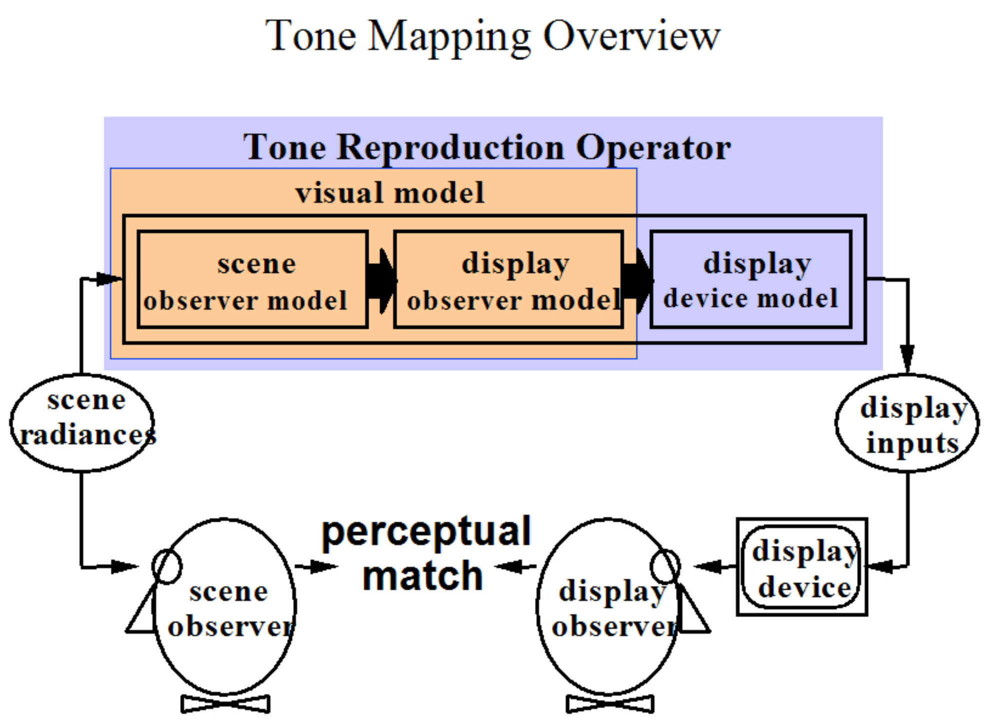


Figure 1.1: High level view of a tone mapping operator [Tumblin93].

luminance to map into the display range. This distorts impressions of brightness and contrast. Linear scaling was also used, a process which works for scenes with similar dynamic range to the display but fails for high dynamic range scenes [Ward97]. Informed and robust tone mapping was badly needed and a conceptual framework for tone reproduction in computer graphics was first introduced by Tumblin in 1993.

A tone reproduction operator in Tumblin's framework (Figure 1.1) has two parts. First, and most importantly is the visual model which allows the visual state of the scene observer and display observer to be related to create a mapping that will optimally reproduce the visual experience of the actual scene on the display. Second is the display model, which includes the physical parameters of the display device. For example the absolute and dynamic range, color gamut, screen resolution and gamma correction factor. These parameters are used in mapping from visual response to the display. The central premise of all such operators is that the visual system loses information in the encoding process, for example through insensitivity to absolute luminance levels, so that different visual stimuli (in this case the radiance in a real scene and the radiance from a CRT display) will produce the same visual response and in turn the same visual experience [Ferwerda98].

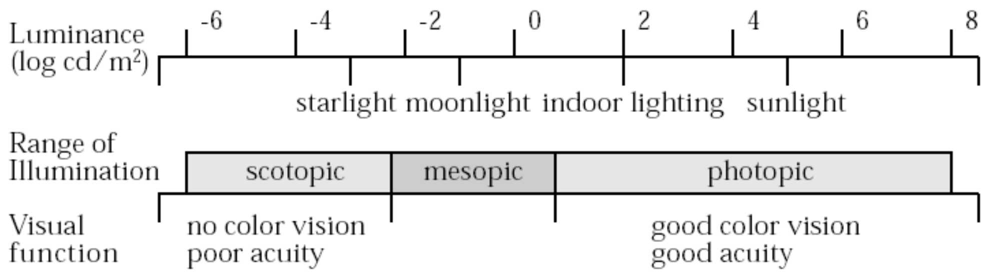


Figure 1.2: The range of luminances in the natural environment and related visual parameters [Ferwerda96].

To better appreciate the tone reproduction problem we briefly examine the range of luminances encountered in nature and associated visual response parameters (Figure 1.2). In the real world the overall range of light energy varies dramatically. For instance the light of the sun at noon can be up to one million times brighter than moonlight and the visual system responds differently over this range. Furthermore, dynamic range (the ratio of the greatest luminance to the least) can easily be 100,000:1 in a night scene with bright light sources or caustics [Tumblin99]. A human observer can adapt to a dynamic range of 10,000:1 in one view. However, a CRT display can only reproduce a luminance dynamic range of ~100:1 [Devlin02]. Simple operators are often ineffective. Figure 1.3 illustrates the difficulty of displaying the appearance of a high dynamic range scene if one only varies exposure. Further difficulties arise if a faithful mapping is desired for all illumination levels because human vision varies, for example, with respect to acuity and color sensitivity (Figure 1.2, visual function), two of a host of variable perceptual phenomena.

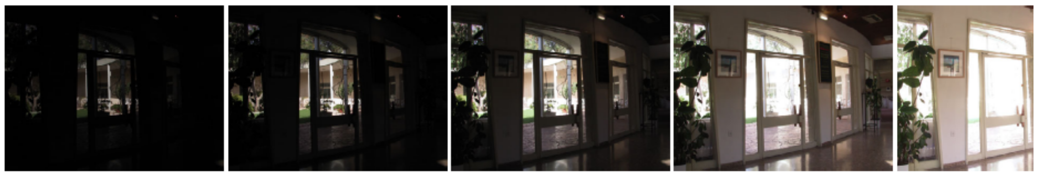


Figure 1.3: Five photographs with exposure increasing from left, illustrating the difficulty involved with reproducing the subjective appearance of an HDR scene on limited dynamic range display devices with simple uninformed mapping like using a global exposure [Fattal02].

There are two related but distinct approaches to tone mapping in the current literature, *preference-based* and *predictive*. *Preference based* operators concentrate all efforts toward mapping the intensity of images such that detail is preserved, and specifically, that subjectively the displayed image looks plausible and free of artifacts. They rely mostly on the assumptions that the human visual system is not very sensitive to absolute intensities that reach the retina, but rather, it responds to local intensity ratios and minimizes large global differences [DiCarlo00, Oppenheim68]. The second approach we call *predictive* operators. These operators refer specifically to the human visual system via psychophysical data and try to mimic visual response over one or more visual phenomena. Some example visual phenomena are dynamic range compression, color saturation and visual acuity as a function of light level, glare, and the time course of light and dark adaptation. The goal is to accurately reproduce the true appearance of a scene (possibly changing over time) on a display while maintaining validity through the use of psychophysical data.

A further useful distinction between operators that applies to how they compress dynamic range is *local* versus *global*. Local operators vary their scene-to-display mapping functions over individual portions of the image. Such operators tend to be computationally more intensive because of this but high quality results can be obtained (e.g [Fattal02]). Global operators use a single mapping function applied to the entire image. Thus, global operators are more efficient but often sacrifice some quality. The sacrifice is due to the one-to-one monitonic nature of the mapping curves. They have trouble reproducing local contrasts in images, particularly where the intensities of a region where detail should be visible populate the entire dynamic range in a nearly uniform fashion [Fattal 02]. For either variety of operator the mapping is applied to the luminance or to each channel (r, g and b) of each image pixel. Some researchers refer to local and global operators with different terminology. DiCarlo and Wandell refer to global operators as TRCs (tone reproduction curves) and local operators as TROs (tone reproduction operators).

In the next chapter we discuss the tone mapping literature and attempt to place the work into the framework just described. This sets the stage for us to discuss our specific tone reproduction goals and to evaluate what current work is valuable to assist in making them possible.

Chapter 2

Previous Work

2.1 Current Operators

To begin our discussion of past work in tone reproduction we chart the space of operators in the following order: preference based global operators, preference based local operators, predictive global operators, predictive local operators and real-time operators. Real time operators are for the most part attempts to accelerate previous operators for use in interactive applications.

2.1.1 Preference Based Global Operators

Few preference based global operators are in serious use in computer graphics today. However, they are used in digital photography standards, and may include logarithmic RGB signal representations to accommodate larger dynamic ranges [Holm00]. Preference based global operators were the forerunners to informed tone reproduction. Prime examples are a gamma corrected linear mapping or taking the cube root of the intensities as discussed in the introduction. However, there are a few notable exceptions.

Early work in computer graphics by Schlick 1994, used a globally applied rational mapping function as an alternative to linear, exponential or logarithmic mappings for compressing contrast. His aim was to improve efficiency and reduce complexity of use.

Geigel and Musgrave [1997] suggested a tone reproduction operator based on photographic principles to simulate the process of black and white film development for digital images. They use models based on empirical data for film development along the dimensions of *density* response, spectral sensitivity, resolution and *granularity*. *Density* measures the opacity induced in an emulsion from light. It is called transmission density in the case of film and reflective density in the case of photographic paper. *Granularity* is the graininess of photographic paper. First an exposure is determined by setting an exposure time interval, filtering the result by a spectral sensitivity curve, and simulating film resolution through internal scattering based on the modulation transfer function (MTF) of an emulsion. Then the *density* response is calculated from the formerly computed exposure value using the characteristic curve for a given emulsion. Film grain is simulated by stochastically modulating the density values with gaussian noise. Finally, the density values are converted to transmission values (film) or reflection values (paper).

Tumblin, Hodgkins and Guenter [Tumblin99], use Schlick's global rational mapping function and take the illumination versus reflection component observation (see [Oppenheim68] for a review) to it's conclusion for synthetic scenes. Illumination and reflectance information is available in synthetic renderings because the material properties are known and the light sources are completely specified. A separate layer for each can be calculated. Then the illumination layer is compressed using an s-shaped or sigmoid curve and recombined with the reflectance layer. The result is dynamic range compression and preservation of scene details. Tumblin also included a foveal adaptation approach in which the mouse position sets a viewer's supposed focus and adaptation level.

2.1.2 Preference Based Local Operators

The earliest work in electronic imaging seems to be Oppenheim, Schafer and Stockam [1968]. They recognize the issue of limited dynamic range in displays and propose an algorithm based on *homomorphic filtering*. Their primary observation (also used by [Stockham72], [Tumblin93] and others) is that a scene can be divided into an illumination component and a reflection component. The problem with dynamic range is due mostly to the magnitude low frequency illumination component. The reflection component contains mostly high frequency information with small intensity variation. Thus by attenuating the magnitude of the low frequency component scene features across the intensity range can be displayed. They accomplish this by attenuating low frequencies in the Fourier domain. A major assumption in this approach is that detail preservation is tantamount to appearance reproduction, which is the primary goal of tone mapping. This early method is slow and suffers from halo artifacts. Halo artifacts are reversals in large intensity gradients that lead to dark regions where there should be light ones. Mostly, this occurs around light sources. As can be seen from Figure 1.1, they are not a desirable artifact.

Chiu et al. also developed a spatially non-uniform scaling function for dynamic range compression. They deliberately avoid other elements of adaptation and operate under the assumptions outlined by Land and Marr [Marr92, Land77] that the scene can be divided into a reflection and illumination component and that the illumination component is responsible for the high dynamic range. Their algorithm hinges on using a mapping that is the reciprocal of the inverse (blurred) intensity image to preserve detail. Unfortunately, halo artifacts (reverse intensity gradients) result around large image intensity gradients [Ward97]. Furthermore, the method is rather computationally intensive.

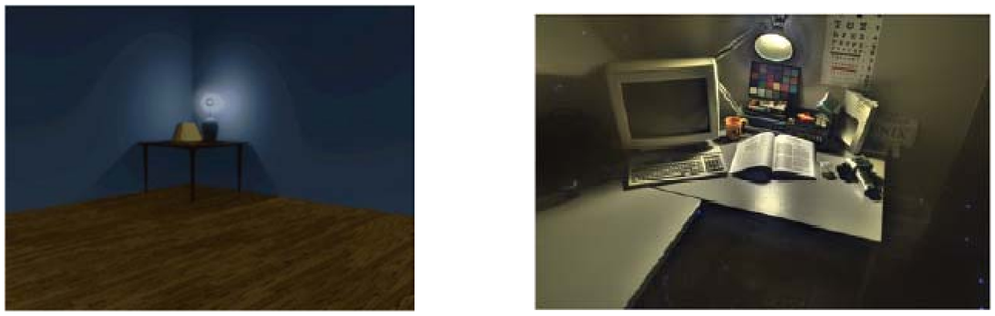


Figure 2.1: Two images illustrating halo artifacts. The image on the right is output from Pattanaik's multi-scale model of adaptation [Pattanaik98]. On the left is output from Chiu [Chiu93]. Both suffer from halo artifacts around the light sources.

Schlick [1994] built upon the work of Chiu. His primary contributions were new methods for creating spatially varying mappings. The focus was on simplification and computational efficiency. The basis for improvement was the use of a rational mapping function for both global and local mappings. He offers three methods to compute the rational mapping function parameters per pixel. The first is low pass filtering just as Chiu. This method creates halo artifacts as before. The second is micro-zones, which is a global mapping derived by reducing the low pass filter of the previous method to the size of a single pixel. The third is segmentation, which attempts to divide the picture up into zones of similar intensity and map these individually. The segmentation step is nontrivial and not worth any improvements over his global mapping.

Tumblin and Turk [1999] built further upon the well entrenched idea that decomposition into low frequency luminance variation (illumination component) and the high frequency details (reflection component) is roughly how the visual system compresses high dynamic range and how one can algorithmically mimic the process.

In this case they took as inspiration the process artists use when painting. Their innovation is the *Low Curvature Image Simplifier* (LCIS) method which uses anisotropic diffusion to separate the scene into a smoothed boundary and shading preserving component and a detail component. The first component is compressed and the details are added back in afterward. The method preserves fine details and avoids halo artifacts (except in some cases of extreme compression). Still, it does not agree with the subjective impression of a scene observer, since it exaggerates details excessively giving a grainy unnatural look. Furthermore, there are many free parameters and the algorithm is computationally intensive.

Ashikmin [2002] provides a tone reproduction algorithm that tries to preserve local contrast in an image by gauging the local adaptation level and applying a simple mapping to compress HDR values into the displayable range. Local adaptation is given by the average luminance in some pixel neighborhood. A pixel neighborhood is gauged through a measure of uniformity and starts small, then increments larger until this criterion is violated or a maximum neighborhood size is reached. For a mathematical treatment of uniformity he uses the band-limited local contrast. Then for each pixel the local adaptation level is passed to a tone mapping function that is based on relating the *perceptual capacity* of a scene and display observer. *Perceptual capacity* is the notion that sensitivity to luminance changes as given by the threshold-versus-intensity (TVI) functions, provides a natural scaling factor for a given small range of luminances. The TVI functions describe the smallest luminance increment detectable at a given background luminance. Finally, contrast is preserved through a locally linear mapping by simply dividing the current pixel luminance times the tone mapping function by the local adaptation value. Although Ashikmin uses TVI data and other data on the HVS (Human Visual System) the accuracy obtained is probably not sufficient because he does not attempt to exactly follow the operation of the HVS.

Still, since neighborhoods are chosen carefully no halo artifacts occur and the algorithm yields good results.

Reinhard et al. [2002] applies techniques from photography, in particular the long standing Zone system [Adams80, Adams81, Adams83] to tone reproduction. They adapt the system by using a *key* approach to setting an initial exposure then effectively *dodge* and *burn* areas as needed. In photography the *key* of a scene indicates whether it is subjectively light or dark. *Dodging* and *burning* refers to adding more light or restricting light during print development. This will lighten or darken regions of the print relative to the normal development process. Reinhard sets an effective *key* for each pixel by using a center surround measure of local contrast based on brightness perception [Blommaert90] at multiple scales to find localities (via thresholds) around pixels bounded by large contrasts. Although there are a few free parameters left to discretion such as *center surround ratio* (ratio of center size to surround size) and some parameters that affect center surround scale, the method produces very attractive results and is reasonably fast and automatic. Because the method effectively finds regions bounded by large contrasts and scales radiances in each region with similar exposure the method is directly related to the base/detail layer approach. However, in the base layer there is not blurring across large intensity gradients and so there are no halo artifacts. Also, detail is not extracted and added in later so it doesn't look grainy.

Durand and Dorsey [2002], present another variation of the base/detail layer approach seen previously [Oppenheim68 etc]. They share the quality of results obtained by Reinhard et al. In this case the base layer is obtained by using a non-linear edge preserving filter known as a bilateral filter which they relate to anisotropic diffusion. The result is a filter that blurs small details but preserves large discontinuities. The base layer is compressed and then recombined with the detail layer producing very satisfactory results. This is done in the framework of Chiu et al. (using a non-uniform scaling function and taking the reciprocal of the inverse for the mapping) but uses a robust filter rather than a low pass filter thus avoiding halos. It lacks the grainy look of LCIS because it doesn't explicitly extract detail. Furthermore, they focus on efficiency (piecewise linear approximation to bilateral filtering as a fast alternative to anisotropic diffusion) so the method is fast compared to previous work (almost real time).

Fattal, Lischinski and Wermann 2002 approach tone reproduction with the same assumptions seen elsewhere in this section, but from the perspective of gradients. They compress large gradients and leave small gradients unchanged thereby preserving small details and compressing the dynamic range. To accomplish this they use a variation of multi-resolution techniques (see reviews of [Pattanaik98] and [Jobson97] for details). They find gradients that need attenuation at multiple scales using an edge detection algorithm and then propagate gradient attenuation factors up to the full resolution image. In this way they avoid the halos which plague multi-resolution techniques. To get an LDR image a Poisson equation must be solved and they use standard finite difference techniques. The method is effective and relatively efficient.

2.1.3 Predictive Global Operators

The earliest work in predictive tone reproduction was by Miller, Ngai and Miller [Miller84]. They used experimental data to attempt to match perceived brightness in a real scene and displayed scene. However, they did not provide a solid theoretical basis for tone reproduction as Tumblin and Rushmeier [Tumblin93] later did.

Another important early work is Upstill's 1985 thesis which is the first multi-featured predictive operator. He first sets forth a framework not unlike Tumblin [1993] in which an attempt is made to match the visual response of a scene viewer and a display observer looking at a digital representation of the scene. After considering two approaches, algorithmic and heuristic, Upstill chose a heuristic approach. The justification is that there is no accurate comprehensive characterization of visual processing and thus no basis for a complete algorithm. Instead he uses psychophysical data collected for various visual phenomena and treats each phenomena separately to create a mapping for HDR images. Although data is collected under ideal lab conditions with simple stimuli, using such data was deemed the most appropriate way to proceed. Most other predictive work relies on the same kind of data. Upstill tackled contrast sensitivity, visual acuity, color appearance, and dynamic range compression.

Basic contrast and dynamic range control proceeds in two steps. First, brightness response is estimated by relating it to the intensity response function, effectively an s-shaped curve. Then the luminance required on the display to evoke the same response is determined. This is done by relating the brightness response function computed for the scene data and the response function from viewing on the CRT, and then solving for the CRT intensity needed for a match. The user controls the saturation levels for the input image and the parameters of the display viewer response function.

Visual acuity is addressed through linear filters. Fourier-space filtering is used to account for a relative loss or enhancement of contrast response from luminance values in the scene to those in the displayed image. Assuming that perceived contrast is proportional to contrast sensitivity, frequencies will be enhanced or attenuated by the ratio of the contrast sensitivity function for the scene and display observer.

Color appearance is treated for color sensitivity as a function of light level, the *Bezold-Brucke effect* (changes in light intensity tend to also change the hue), desaturation of color at high light levels and *chromatic adaptation* (For a review of these phenomena see [Upstill85]). These are treated within the *opponent-colors* response model [Jameson56, Hurvich56, Hurvich81]. This model expresses chromatic response as one achromatic (Black/White) channel and two chromatic channels (Yellow/Blue and Red/Green). Rod response is treated separately. Color sensitivity is controlled with a linear scale factor over the range of photopic, mesopic and scotopic light levels between the rod and cone response. The remaining effects can be modeled with appropriately chosen scale factors on the two color channels.

Pioneering work for both tone reproduction and predictive operators in computer graphics was done by Tumblin and Rushmeier [Tumblin93]. They focused on preserving the perception of brightness for a scene observer (person in the real scene) and display observer (person looking at a CRT display). See Figure 1.1. Brightness is estimated by a functional fit to Stevens [1960] brightness versus luminance curves for different adaptation levels. For the scene viewer adaptation is estimated by the mean logarithm of the luminance. For the display viewer the adaptation state is estimated by the typical peak luminance of a CRT display. The tone reproduction operator is formed by setting both the scene brightness and display brightness to be the same, and then solving for display values given scene luminances. The operator can handle scenes with extremes of brightness with the drawback that brightness may be preserved at the expense of visibility.

In a new approach to tone reproduction Ward [1994] concentrates on preserving apparent contrast or visibility rather than brightness. A single scale factor is chosen for the image based on a method that hinges on the fact that adaptation can be viewed as a scaling of the absolute difference in luminance required for an observer to notice a difference. This result is known as Weber's Law which says that the size of a just noticeable difference is a constant proportion of the original stimulus value. To use this fact Ward takes psychophysical data for contrast sensitivity over a range of world adaptation levels, sets the adaptation level using a supposed fixation point, and chooses a scale factor that will preserve the Just Noticeable Differences (JNDs) in contrast in the real scene and on the CRT display (at least at the fixation point). The operator is very fast, but because it uses a linear scaling factor and one fixation point, the highest and lowest values will be clipped and visibility won't be maintained everywhere in the image. The method is also dependent on how applicable contrast sensitivity measurements (which are done in ideal laboratory conditions) are in complex scenes. However, all Predictive operators have such dependencies.

Ferwerda [Ferwerda96] built upon and extended the threshold visibility approach seen in Ward by adding additional elements from human visual adaptation that have a significant effect on how a scene will appear under various conditions. In addition to dynamic range compression via thresholds and a linear operator Ferwerda modeled rod response, color appearance, visual acuity, changes in achromatic contrast sensitivity and changes in visual sensitivity over time.

Color appearance changes due to the differences in relative spectral sensitivity of the rod and cone system over the photopic, mesopic and scotopic ranges of intensity levels (Figure 1.2). In the scotopic range Ferwerda uses an operator based on the rod TVI (threshold-versus-intensity) data and monochromatic response. In the photopic range he uses the cone TVI function. Finally, in the mesopic range he uses a linear combination of the rod and cone response. The TVI functions are experimentally measured and indicate the luminance difference between target and background necessary to detect the target over a full range of background luminances.

To model visual acuity and thus preserve resolvable detail in the display and scene observers Ferwerda uses Shaler's [Shaler37] data for grating acuity as a function of background luminance. This data gives the spatial frequencies that will be visible at a given adaptation level. By removing all spatial frequencies (via a convolution) from the image that are below the maximum resolvable frequency, one can mimic the visual response. This is done globally, whereas the eye adapts locally. Also, this may not account for the actual appearance of a scene but it is the first such work and the basis for all work afterward [Ferwerda96].

Visual sensitivity over time can be encapsulated in two related phenomenon: light and dark adaptation. We experience light adaptation going from a dark adapted state to a light adapted state. Sensitivity takes a few seconds to come back and contrasts are reduced. The same thing happens in dark adaptation but in reverse and over tens of minutes. Physiologically, light and dark adaptation are due to bleaching and regeneration of photopigments and neural processes (see [Ferwerda96] for a brief review). Ferwerda models both of these using a simple multiplicative gain control that changes over time.

It has been noted by Ferwerda [Ferwerda96] that there is an important distinction among predictive operators. There are those that address *threshold* appearance ([Ward94], [Ferwerda96], [Ward97]) and those that address *suprathreshold* appearance ([Tumblin93]). The first approach scales *suprathreshold* values according to the threshold measure. Therefore it may not capture *suprathreshold* appearance. The latter may preserve the appearance of surfaces in the scene but fail to capture visibility near threshold. An ideal predictive operator would account for both domains. To my knowledge only one such attempt at both exists: Pattanaik's [1998] multiscale model of adaptation, spatial vision and color appearance.

Ward, Rushmeier and Piatko extend the predictive visibility approach to tone mapping by introducing a novel histogram adjustment technique and models of glare, acuity and color sensitivity [Ward97]. Histogram adjustment is intended to overcome the problems encountered if a single adaptation level and linear mapping are used, namely clipping and poor subjective appearance, while avoiding the computational cost of local operators.

The histogram adjustment algorithm depends on the assumption that the eye quickly and locally adapts to a one degree view around a fixation point. Ideally, one would want to take into account all possible fixations when forming a final global operator. In [Ward94] and [Ferwerda96] only a single adaptation level is chosen. The algorithm is directly related to histogram equalization but uses contrast sensitivity data [Ferwerda96] to guide histogram changes. First, a histogram (log approximation of brightness) of each one degree fixation point (forming a one degree foveal image) is made. Then the global operator is formed by calculating the cumulative distribution function of the histogram. However, the slope of the operator is constrained so that for each adaptation level JND's are preserved. Ward converts the constraint on the cumulative distribution function to a ceiling on bin counts in the histogram, which is enforced using an iterative procedure. The algorithm is very efficient and produces good results for a range of scenes.

Ward borrows his acuity and color sensitivity algorithms from Ferwerda [Ferwerda96]. Ferwerda's color mapping is used without change. However, acuity is modeled to take into account local adaptation, while Ferwerda applied a single global blurring function. Local blurring is done according to the one degree foveal adaptation image using a mipmap image pyramid [Williams83] with actual luminance values rather than integer pixels. At each pixel in the input image the effective adaptation level at the pixel is an interpolation of the four closest foveal adaptation samples. Local acuity is derived from Shaler's data [Shaler37].

Glare is treated in terms of veiling luminance, the added effective luminance around bright lights and reflections due to scattering in the optics of the eye. The operator is a direct application of Moon and Spencer [Moon and Spencer45], which computes an effective adaptation luminance using the one degree foveal average luminance, glare source position and illuminance. The approach is novel because glare is included in a larger tone mapping context and effects how dynamic range compression, acuity and color sensitvity are calculated. However, the glare model is missing a treatment of two phenomena: lenticular halos (radial bright lines) and cilliary coronae (circular rainbow patterns) often seen when looking at bright lights at night [Spencer95].

In a departure from previous lines in predictive global operators, Pattanaik, Tumblin, Yee and Greenberg [Pattanaik00], thoroughly address the time course of adaptation. This includes light and dark adaptation. However, all other desirable elements (predictive ability with respect to acuity, threshold visibility and/or suprathreshold brightness and color saturation) are left behind for the sake of simplicity and fast computation. Also, the eye is idealized as having a uniform adaptation state. The operator is designed for automatic use with image sequences.

Calculation proceeds in four steps. Initially, rod and cone responses are estimated using the luminance based portion of Hunt's visual response model [Hunt95]. Then time driven exponential filters are placed on parameters of the response model to simulate the response compression and bleaching associated with light and dark adaptation (see section on the psychophysical basis for adaptation). Next, the appearance model is applied. This model assumes that the viewer estimates scene intensities by comparing display intensities against mental estimates of reference white and reference black and is a linear mapping. Finally, an inverse appearance model and inverse visual response model is applied giving displayable values. Much further work is possible on the time course of adaptation. First, including more of Hunt's model of static color vision to include dynamic color adaptation is a clear path. Second would be to add other effects like loss of acuity with light level, color sensitivity etc. Finally, local adaptation could be considered. Unfortunately, no further adaptation time course work has been done.

2.1.4 Predictive Local Operators

Most local operators do not strictly take the processing of the HVS into account, but rather depend on a few observations about how feature appearance is preserved for an observer despite high dynamic range in real scenes. However, in the first of two exceptions to this, Pattanaik, Ferwerda, Fairchild and Greenberg [Pattanaik98] develop a method that attempts to address the need for threshold and suprathreshold appearance in tone reproduction by using an explicit model based on a multi-scale decomposition of pattern, luminance and color processing in the HVS. This is important on two fronts. First, as Ferwerda [Ferwerda96] noted there is a need for a visual match at threshold and at suprathreshold levels of illumination (This is the only work that tackles both). Second, more local operators should incorporate data from the HVS, thereby adding a measure of validity and allowing broader use. Pattanaik et al. break ground in this area, but further work is most certainly needed. Their operator allows sufficient compression of high dynamic scenes but suffers from annoying halo artifacts due to the multi-resolution decomposition and subsequent separate processing of each level (a fundamental problem with multi-resolution approaches) [DiCarlo00]. Also, the method is computationally intensive.

The multiscale model of Pattanaik et al. can model threshold visibility, color discriminablity, and suprathreshold brightness, colorfulness (including chromatic adaptation), and apparent contrast over the full range of illumination levels. This versatility was the product of a framework they developed that relates research on adaptation with research on spatial vision. The result is a unified view of variations in threshold performance and suprathreshold appearance. Processing is divided into two parts. First, the visual model is applied to encoded perceived contrasts for chromatic and achromatic channels with their respective bandpass mechanisms. Second, a display model is employed which uses the response information to reconstruct an image appropriate for display on a CRT.

Before Pattanaik et al. Jobson, Rahman and Woodell [Jobson97], adapted the retinex theory of color vision to make a spatially varying, multiscale tone reproduction operator designed to achieve dynamic range compression, color consistency and lightness rendition. In terms of the more comprehensive framework of Pattanaik [Pattaniak98], Jobson et al. addressed suprathreshold appearance issues, while ignoring threshold measures. Since Jobson’s model depends on a multiscale decomposition it to suffers from halo artifacts. Otherwise the method performed well over a range of test images with difficulty only with extremes of contrast and if the gray world assumption of the retinex theory were violated [Jobson97].

2.1.5 Real-Time Operators

Real-time operators are, for the most part, attempts to accelerate tone reproduction operators for use in interactive applications. This is invariably done by using graphics hardware. Very little work has been done in this area. The work that has been done has significant limitations with respect to the range of phenomena treated and the type of input assumed. All existing approaches are restricted to radiosity solutions with luminance values at the vertices (texture and other forms of lighting can not be correctly included). This is a significant restriction that rules out use for processing complex global illumination solutions, HDR images and video streams ([Debevec97],[IMS]) in real time

The first work in real-time tone reproduction was done by Scheel, Stamminger and Seidel [Scheel00]. Scheel developed a tone reproduction operator for use in displaying radiosity solutions. Radiosity values were stored at mesh vertices and tone mapped before display using one of Ward's operators. For each frame the adaptation level is chosen using a center weighted average to determine a plausible point of focus and ray tracing to get samples. Either a luminance histogram is formed with these samples in the case of Ward [1997] or a single adaptation level is chosen in the case of Ward [1994]. The global operator is set using this information (see [Ward97] and [Ward94]) and applied to the vertex radiosities. Scheel's operator only treats dynamic range compression and makes no attempt to include a full range of perceptual effects such as glare, acuity changes with light level, color saturation, or time course adaptation. Although good performance was achieved, the sacrifice in usefulness is significant. Furthermore, because not all samples are updated each frame, there is an ad hoc time course model such that it takes some time for the operator to catch up to a new view.

Durand and Dorsey [Durand00], approached real-time tone mapping similarly to Scheel. A radiosity mesh with possibly high dynamic range values at the vertices is used for input and vertex values are tone mapped before display. It suffers from the same limitations in usefulness as Scheel, but includes many more perceptual phenomena. Durand's orperator is based directly on Ferwerda's [Ferwerda96] and borrows Ferwerda's threshold based contrast mapping, model for acuity, color appearance and time course adaptation. The contrast mapping is accelerated on hardware by caching the final mapping function in a lookup table (although in their case this is not necessary, and other mappings beyond a global scale factor can be substituted). Adaptation level is chosen using photographic techniques (centered weighted average [Nikon00]) and acuity calculations are done the same as Ferwerda but using hardware convolution to remove unresolvable frequencies from the final image.

In addition to these phenomena, simple models of *blue shift*, and *chromatic adaptation* are added. With regards to *blue shift*, researchers have hypothesized that rod signals are interpreted as slightly bluish [Millerson91, Hunt52] . *Chromatic adaptation* is that we tend to perceive objects as having a constant color even when observed under illuminants with various hues [Fairchild95]. Finally, an interactive glare model adapted from Spencer [Spencer95] is applied to light sources. The glare algorithm restricts calculation to light sources and displays glare with textured polygons. Overall, Durand skips major advances made in Ward's [1997] work by focusing on Ferwerda's [1996] work. Improvements include nonuniform acuity calculations and Ward's histogram adjustment algorithm.

2.2 Related Work

In addition to the work discussed in previous sections there is some related work worth pointing out. First is a comprehensive glare operator by Spencer et al. [1995]. Spencer does a rigorous treatment of glare including bloom and flare. Next, is work in real time tone mapping by Cohen et al [2001]. Cohen tackles how to use

HDR textures on commodity hardware that doesn’t support floating point textures and shows how to apply an exposure before displaying the texture in the low bit depth frame buffer

2.3 Summary

The current tone mapping literature offers a wide variety of approaches. Researchers have attempted to accurately derive tone mapping operators to visual perception by using psychophysical data. This approach can be described as predictive. Other researchers have used subjective measures of correctness to gauge the success of their operators. This approach can be called preference based. Predictive operators have mostly addressed threshold measures of perception. However, a few have tackled suprathreshold measures or both suprathreshold and threshold measures. Clearly, more work is needed to create an accurate predictive operator that includes both domains and is reasonably efficient. This is a difficult problem because the HVS is not completely understood and the validity of psychophysical data outside of laboratory conditions is debatable. Despite the need for work in predictive operators, the majority of recent work has been directed toward preference based operators and a number of high quality techniques have emerged based on high level descriptions of how the HVS functions overall. For example, the HVS preserves detail over large dynamic and absolute ranges. In the next chapter we discuss the aim of this current work in the context of the state of the art in tone mapping today. We describe what we think is a next plausible step in the evolution of tone reproduction operators.

Chapter 3

Requirements and Basis for a New Operator

The goals of this thesis are two-fold. First, to create a widely applicable tone reproduction operator within real-time constraints that includes as many visual phenomena as possible and is predictive (i.e. has a basis in psychophysical data on vision). Second, to apply this operator to low vision simulation by substituting data for normal subjects with data for visually impaired subjects. The requirements for our new tone reproduction operator are determined by these goals. This chapter explains why certain operators were chosen as a starting point for our real-time operator. Other sections detail how real-time performance is achieved and what approximations or additions are made to the starting operators. The previous chapters on past work (Chapter 2) will be useful for understanding the choices made herein.

Figure 3.1: A review of some target applications. At left, architectural walkthroughs and illumination engineering. At center, flight/car simulators. At right, high dynamic range video or photos of real environments. A darkened stairway is shown to emphasize use for low vision simulation.

A widely applicable real-time tone reproduction operator should not impose limiting assumptions concerning the form of high dynamic range data input. Input from simulation, high dynamic range photography [Debevec97] and streaming video [IMS03] should all be accommodated. Basically, any source of high dynamic range pixels should be easily processed. If the operator meets this criterion then applications in illumination engineering, architectural design, flight/automobile simulators, entertainment and real-time HDR video capture and processing are possible (Figure 3.1).

Real-time performance means processing within frame time. As data arrives the pictorial results are computed fast enough for smooth video, animations or hardware based computer graphics. The minimum frame rate that would make the operator useful in all three domains is widely accepted to be approximately 10-15 fps [Bruce96]. However, some have put this as high as 17 fps to fully convey dynamic human facial information [Bruce96].

Most importantly, any basis for our operator must satisfy two requirements: it must be predictive and amenable to acceleration. The ability for an operator to be predictive is essential if it is to be useful in a vision simulation context. In this case, threshold measures take precedence over suprathreshold measures because the former predicts visibility which is of primary concern when, for example, evaluating a building for suitability with low vision individuals. For a discussion of threshold and suprathreshold visual phenomena and tone mapping operators that include one or both categories see the previous work section (Chapter 2). The only published operators which satisfy both these requirements are threshold based predictive global operators, developed by Ferwerda [Ferwerda96] and Ward [Ward94, Ward97]. These operators have the required basis in psychophysical experiments and are readily accelerated (see Scheel [Scheel00] and Durand [Durand00] for examples of attempts to accelerate

[Ward97] and [Ferwerda96]) using graphics hardware. This is due mainly to the global mappings used. However, no single current real-time or off-line operator includes all relevant perceptual phenomena. A comprehensive predictive threshold based tone reproduction operator should address all the phenomena illustrated in Figure 3.2. We deem other phenomena like the perception of brightness, colorfulness and apparent contrast less crucial because they don't directly affect visibility and more subtly affect appearance.

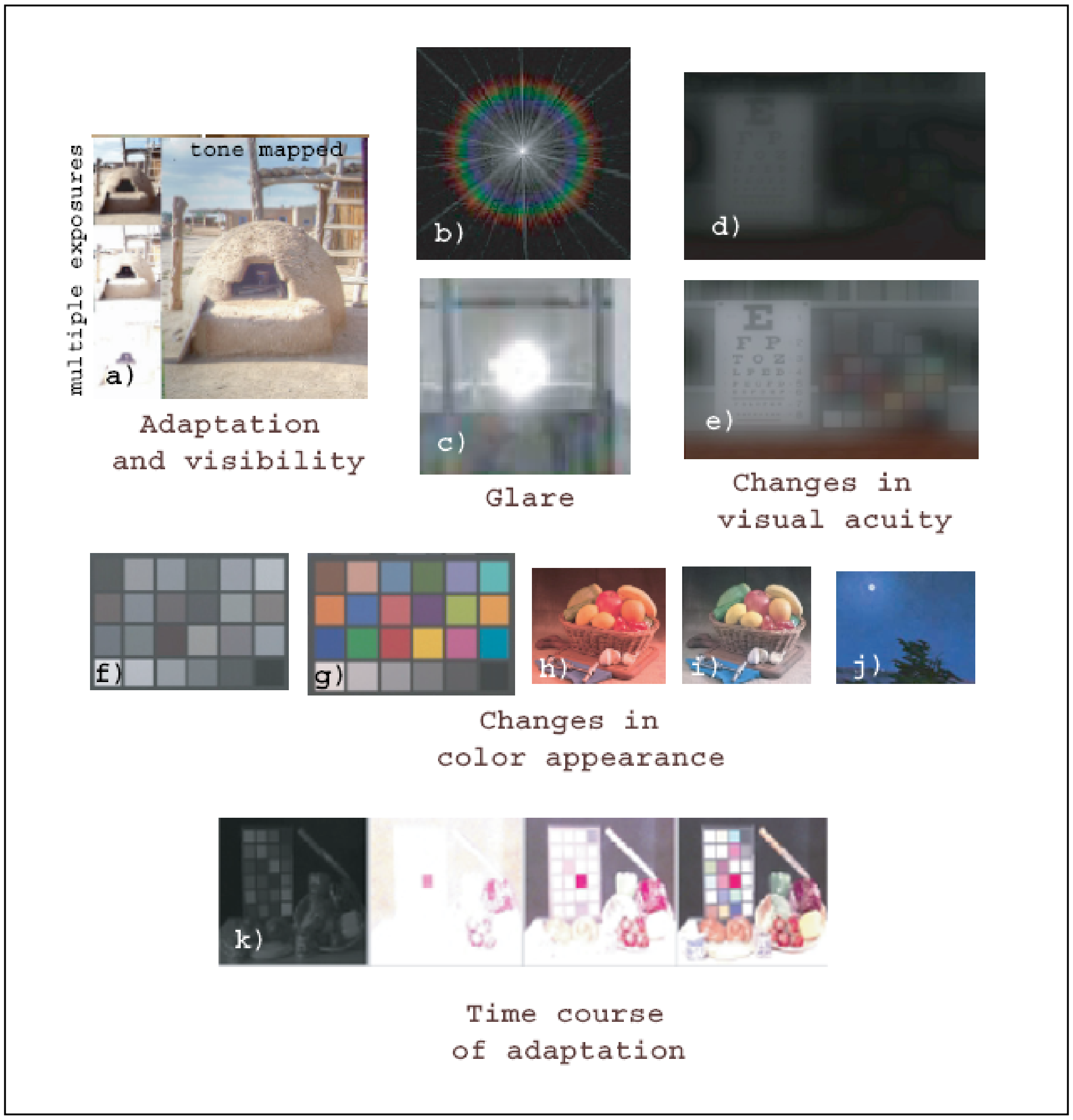
The following discussion refers to Figure 3.2. Each phenomenon is discussed in order from left to right and then top to bottom. For a more detailed review of the psychology behind these and other phenomena refer to [Ferwerda98].

First, the human visual system (HVS) has limited sensitivity and so adapts over the range of world illumination levels in an attempt to preserve scene features. Slow varying large intensity changes are mostly ignored and small contrasts are preserved. All work in tone reproduction hinges on this fact and thus most methods compress high dynamic range data in various ways in an effort to mimic this phenomenon. Some use a single global mapping, others a spatially varying mapping, some are predictive and others are preference based. For example, Figure 3.2a is from Fattal [2002], a preference based local operator. On the left hand side of the image are multiple exposures which can be contrasted with the tone mapped result on the right.

We require a predictive global operator with a threshold based criterion for dynamic range compression. Visibility is our foremost concern. Of the three candidates mentioned previously Ward [1997] stands out as the best choice. Ferwerda's work [Ferwerda96] is based on Ward's [1994] and Ward's newest work updates Ferwerda's. It represents the current state of the art in predictive global operators. Ward's main improvement is a histogram adjustment algorithm for dynamic range compression [Ward97]. It is a widely applicable visibility preserving algorithm: it preserves just noticeable differnences (JNDs) in luminance in the real scene on a CRT display over many localized adaptation levels. Instead of a global scale factor and it's associated disadvantages (clipping and poor subjective appearance) a histogram is formed from local adaptation estimates and is used to make the mapping. The results are far superior to previous global operators and the algorithm is still very efficient. Thus, our basis for dynamic range compression is Ward's histogram adjustment algorithm.

The effects of glare are very important for predicting visibility and correct subjective appearance in scenes with light sources that are bright relative to the ambient light level. It can be described by two main phenomena flare and bloom. Flare (Figure 3.2b) consists of a *cilliary corona* and a *lenticular halo* and is due mostly to the lens. The *cilliary corona* exhibits as radial lines emanating from the center of bright light sources and is caused by semi-random density fluctuations in the nucleus of the lens which scatters light. The *lenticular halo* is a set of concentric colored rings beyond the *cilliary corona* caused by diffraction in the fibers at the radial edge of the crystalline lens. Bloom (Figure 3.2c) is due to scattering of light in the cornea, lens and retina. It produces what is known as *veiling luminance* because the extents of the light source are effectively spread out and contrast is reduced in the region of spread. Of existing operators only Ward [Ward97] and Spencer [Spencer95] treat glare rigorously. Ward concentrates on veiling luminance while Spencer simulates both phenomena. The images in figure two show Spencer's glare kernel (See [Spencer95] for details) and its application to a bright indoor light. For our purposes, veiling luminance is all that is required, since this is the dominant factor in determining visibility. Secondary appearance effects caused by the cilliary corona and lenticular halo can be considered as computational resources permit. For this reason, Ward's work is a good starting point. Also, Ward's glare operator has the

Figure 3.2: Visual phenomena as modeled in various work. Image a) shows adaptation/ contrast visibility. The image shows the results of using three different exposures and compares them to a tone mapped version [Fattal02]. Glare consists of two effects: flare and bloom. Flare b) is due to the structure of lens and has two components, the cilliary corona (radial lines) and the lenticular halo (rainbow surround). Bloom (c) is caused by scattering in the cornea, lens and retina [Spencer95]. The acuity of the visual system is worse at scotopic light levels than photopic levels. It is gauged by the highest spatial frequency square wave grating visible at a given background luminance level. The top image (d) is at 0.1 cd/m^2 and the bottom (e) is at 1 cd/m^2 [Pattanaik98]. Color sensitivity is lost starting at mesopic down to scotopic levels. Images g) and f) show color appearance with daylight and moon light [Ferwerda96]. Images h) and i) show chromatic adaptation for red and green light [Pattanaik98]. Image j) illustrates a theorized blue shift for night scenes [Jensen00]. The time course of adaptation can be described by two related phenomena, light adaptation (k) and dark adaptation. It takes a few seconds up to many minutes for our visual system to adjust to changes in light level (starting from a dark or light adapted state) [Pattanaik00].


multiple exposures
tone mapped
a)
Adaptation and visibility
b)
c)
Glare
d)
e)
Changes in visual acuity
f)
g)
h)
i)
j)
Changes in color appearance
k)
Time course of adaptation

advantage that it treats veiling luminance in the context of tone reproduction. Veiling luminance is taken into account when determining scene viewer adaptation state so glare effects the computation of the global operator.

The acuity of the HVS changes with adaptation level (Figure 3.2d,e). Chapter 5 has a brief review of the science behind this phenomenon. Of the threshold based operators only Ferwerda [Ferwerda96] and Ward [Ward97] attempt to deal with changing acuity (see previous work). Ferwerda does a global acuity calculation and so misses local variation in acuity. Ward improves upon Ferwerda by blurring locally based on a foveal adaptation image [Ward97]. Thus, Ward offers an important improvement over Ferwerda as starting point for acuity modeling.

In addition to acuity change over light level, the perception of color changes as well. Color appearance changes due to a change in spectral sensitivity of the HVS over the photopic, mesopic and scotopic ranges of adaptation levels (Figure 3.2f,g) caused by differences in relative spectral sensitivity of the rod and cone system. Both Ferwerda and Ward use the same mapping for color, a combination of rod and cone response depending on the range of adaptation level. We chose to use this mapping as well. The two renderings of the MacBeth color checker chart [Ferwerda96] illustrate color perception at moonlight (Figure 3.2f) and at daylight (Figure 3.2g) adaptation levels. For the moonlight level the intensity is changed to accentuate the color differences. The actual picture would be much darker.

Visual sensitivity over time can be described by two related phenomena: light and dark adaptation. We experience light adaptation going from a dark adapted state to a light adapted state. Sensitivity takes a few seconds to come back and the world appears uncomfortably bright and desaturated until it does (Figure 3.2k). We experience dark adaptation going from a light adapted state to a dark adapted state. Initially, sensitivity is very low and we are temporarily blind then it slowly increases over about 35 minutes as the rod and cone system adapt. At the start sensitivity is dominated by the cone system until the sensitivity of the rod system surpasses it after about 7 minutes. The quick change in sensitivity seen in light adaptation and at the start of dark adaptation can be explained by a multiplicative gain control process in which the receptor input is scaled by a constant related to the background luminance. The slow recovery of sensitivity seen in dark adaptation can be explained by a subtractive process in which the base response caused by a constant background is reduced [Ferwerda96].

We refer to these changes as the "time course" of adaptation and it is absolutely vital for any real-time operator because time course adaptation changes are often dramatic, as we know from experience. Ward does not tackle time course adaptation. Ferwerda models both light and dark adaptation using a simple multiplicative gain control that changes over time. Pattanaik [Pattanaik00] however, more thoroughly addresses the phenomenon with a more sophisticated rod and cone response model based on Hunt's model of color vision [Hunt95] in addition to a multiplicative gain control for both rods and cones. Although more sophisticated, Pattanaik's work is designed for real-time use in animations or walkthroughs because it uses a global rather than local response model. Thus Pattanaik's operator is ideal for addressing the time course of adaptation. However, it is not useful in the context of a tone reproduction operator based on histogram adjustment and contrast sensitivity because it just displays the result of the rod and cone response curves without taking into account the mapping of JND's in the scene to JND's on the display. Thus, it may be accurate with respect to estimated visual response over time but won't insure correct visibility, a key goal of our operator. So instead of building directly upon previous work we construct our own time course model that attempts to be automatic, real-time and include heretofore unseen local adaptation effects.

Now, given the outlined requirements and starting point based on the current tone mapping literature, do any current real-time implementations fit our requirements ? No. There is no such operator. The only attempts are Scheel [Scheel00] and Durand [Durand00]. Scheel's operator only treats dynamic range compression and makes no attempt to include a full range of perceptual effects such as glare, acuity changes with light level, color appearance changes, or the time course adaptation. Furthermore, applicability is limited due to a restriction on input to radiosity solutions with radiosities stored at vertices. Durand's operator is also limited to radiosity solutions though he attempts to be more comprehensive in modeling perceptual effects. His work is based largely on Ferwerda's 1996 paper and its performance depends on the number of primitives in a radiosity solution. Another problem with Durand's work is that he didn't use commodity graphics hardware but rather a Silicon Graphics Onyx2 Infinite Reality using one R10K processor. Finally, since Durand doesn't attempt to use the updated acuity or dynamic range compression found in Ward's 1997 paper it is not useful for our purposes. Neither work can accommodate our need for a widely applicable real-time tone reproduction operator that includes as many visual phenomena as possible and is predictive. However, the use of hardware is essential to their performance gains. One difficulty that Sheel and Durand both faced was that a high dynamic range pixels could not be processed on the hardware and very limited calculations were allowed per fragment [OpenGL]. In the current generation of commodity hardware, textures with RGB bit depths of 16 bits are allowed. This advance will allow a purely HDR image based operator. In addition, advances in fragment shader technology allows much more sophisticated per fragment operations. We intend to use commodity hardware because of its wide availability and new found flexibility.

In the next two sections we discuss the current state of the art commodity hardware and then discuss our new real-time operator in this context. The operator is largely based on Ward's 1997 histogram adjustment algorithm. Care is taken not to sacrifice the validity of our operator with poor approximations and assumptions; a problem which has affected past real-time work.

Chapter 4

Capabilities of Current Commodity Graphics Hardware

In this section we review the current state of the art in commodity graphics hardware. We begin with a brief account of the evolution of these devices. Then we discuss in more depth those components which have proved useful for accelerating our tone reproduction algorithm. For detailed information on current hardware see the ATI and NVidia web sites ([ATI],[NVidia]) and the OpenGL 1.4 ARB specification [OpenGLSpec].

The history of commodity graphics hardware is not very long. Just ten years ago specialized devices on Silicon Graphics workstations dominated the graphics world and companies like NVidia were just starting out. Beginning in 1996 higher end commodity graphics hardware (the NVidia Voodoo 3Dfx in particular) implemented the Phong lighting model and Gouraud shading algorithm completely in hardware. Then in 1999 NVidia's GeForce256 also accelerated the transformation and lighting of vertices in hardware. The central motivation for moving functionality to the graphics hardware was to overcome limited bandwidth between the CPU (Central Processing Unit) and graphics board. The next evolution happened in 2001 with the advent of the NVidia GeForce3 graphics card and its GPU (Graphics Processing Unit), which allowed graphics programmers to control the transformation and lighting of vertices and fragment operations through assembly language programs. Before this only very limited control could be exercised over both the vertex and fragment pipelines. These highly parallelized hardware implementations became extremely fast and the challenge then became tailoring algorithms to fit graphics board architectures.

In the current generation of graphics hardware, vertex and fragment programs have become increasingly sophisticated with regards to available operations and how programs are written. High level API's like NVidia's Cg (C for graphics) [Mark03] and Microsoft's HLSL (High Level Shading Language)[HLSL] have been developed to simplify programming GPU's. Other core improvements include extended precision throughout the graphics pipeline, floating point P-Buffers and floating point textures. Given this new found flexibility and the fact that performance of graphics hardware doubles about once every twelve months, these devices have also found uses other than traditional computer graphics, providing high-performance platforms for image processing and numerical algorithms (see [Bolz03] and [Kruger03]).

We use the new found capabilities of modern graphics hardware to allow the implementation of our predictive real-time tone reproduction algorithm. The specific capabilities critical to our implementation are floating point textures, P-Buffers, automatic hardware mipmapping and fragment shaders. Each is described throughout the remainder of this chapter. Detailed examples of code are in Appendix A.

4.1 Floating Point Textures

Floating point textures are hardware textures with floating point values for the R,G,B and A channels. These textures are available through the same mechanism as normal 8-bit textures, the only difference is that the `components` argument is set to `GL_RGB16` or `GL_RGBA16` to get 16-bits of precision per channel. Sample code that gives the OpenGL function call for creating a 2D texture and the appropriate arguments for creating a floating point texture are shown in Appendix A1.

4.2 P-Buffers

P-Buffers are off screen pixel buffers available through the OpenGL extension WGL_ARB_pbuffer. Extensions are explained in the OpenGL extensions specification [OpenGL]. An extension name is divided into three parts. The first part can be GL (standard OpenGL), WGL (Windows OpenGL), or GLX (Xwindows OpenGL) and it describes the context in which the extension is available: cross platform, windows, or Xwindows. The second part describes who has approved or created a given extension. For instance ARB extensions are those approved by the OpenGL Architecture Review Board. Other examples are EXT (not ARB approved), SGIS (Silicon Graphics), WIN (Windows), IBM, HP, INTEL, ATI and NV (Nvidia) to name a few.

When WGL_ARB_pbuffer is used in conjunction with the WGL_ARB_pixel_ format extension, they enable hardware accelerated off screen rendering. The contents of the P-Buffer can then be read back or bound as a texture with WGL_ARB_render_ texture for use in cube maps [OpenGL] or dynamically changing textures. The newest P-Buffers (Radeon 9700) allow floating point values. These features were used for the 'Rendering With Natural Light Demo' at SIGGRAPH 2002 [Natural] which showcased the use of HDR textures for increased realism in hardware rendering. Appendix A2 shows a block of code that creates a 16-bit float P-Buffer in an OpenGL rendering context. For an explanation of the functions involved see the MSDN online library and the OpenGL 1.4 Specification. For a primer see the NVidia web site [NVidia]. Comments in the code describe each step in the process of creating a simple floating point P-Buffer.

To create a P-Buffer in Windows, the current device context (a reference to the graphics hardware being used) must be retrieved along with the current OpenGL rendering context. Next, parameters for the P-Buffer such as the number of channels, bit depth of each channel, and modes of operation are selected. This is done by specifying the desired parameters and then querying the current device context to see if what you want is available. If the P-Buffer format desired exists a new P-Buffer will be created. Then a device context is retrieved for the new P-Buffer and a new rendering context is created based on the device context. Once this has been done all OpenGL calls will apply to either the standard or the P-Buffer rendering context depending on which is selected at the time. Switching between rendering contexts is done using the wglMakeCurrent function. Textures can be shared between contexts using the wglShareLists function.

4.3 Automatic Hardware Mipmapping

This OpenGL extension allows a programmer to specify a texture and let the hardware calculate the mipmap levels on the fly. Each time the texture is updated the mipmaps are recalculated. Specifying hardware mipmapping is straight-forward and also works for 16-bit depth textures (on Radeon 9700 and new graphics boards). Appendix A3 shows a block of OpenGL code that sets up a 16-bit texture for hardware mipmapping. Further details can be found under 'Automatic Mipmap Generation' in the OpenGL 1.4 Specification. It has been promoted from the GL_SGIS_generate_mipmap extension which is used in the code shown in Appendix A3 so that all one needs to do now is set the texture parameter `GENERATE_MIPMAP` to `TRUE`. The example in Appendix A3 traces the standard formation of an OpenGL texture enabled for hardware mipmapping. First a texture object is created and bound as the current texture. Next standard magnification and minification filters, and texture wrap parameters are set for the texture. Then, assuming that we are using the

GL_SGIS_generate_mipmap extension, we first request `GL_NICEST` as a hint (suggestion) to the card to make the highest quality mipmaps possible. Then we set the base level, an integer identifier that refers to the bottom level of the mipmap and the maximum level which gives the final level to be calculated during mipmap formation. Finally, the texture parameter `GL_GENERATE_MIPMAP_SGIS` is set to `TRUE` and the texture is formed.

4.4 Fragment Shaders

The capabilities of fragment shaders have moved from simple texture combination operations and lookups to a powerful complement of assembly language operations. Furthermore with the development of high level APIs, creating custom programs is relatively quick. The newest specification, approved by the OpenGL Architecture Review Board, is the GL_ARB_fragment_program in the OpenGL 1.4 specification. Table 4.1 below has a list of the commands available. Information on fragment operations, syntax, and conventions is also summarized in Table 1. We will only discuss in detail those commands which are relatively new to fragment programs and enable a much wider range of capabilities.

Table 4.1: Summary of fragment program instructions. "v" Indicates a floating-point vector input or output, "s" indicates a floating-point scalar input, "ssss" indicates a scalar output replicated across a 4-component result vector, "ss--" indicates two scalar outputs in the first two components, "u" indicates a texture image unit identifier, and "t" indicates a texture target. There are 33 fragment program instructions.

Instruction	Inputs	Output	Description
ABS	v	v	absolute value
ADD	v,v	v	add
CMP	v,v,v	v	compare
COS	s	ssss	cosine [-PI,PI]

Table 4.1 (Continued)

DP3	v,v	ssss	3-component dot product
DP4	v,v	ssss	4-component dot product
DPH	v,v	ssss	homogeneous dot product
DST	v,v	v	distance vector
EX2	s	ssss	exponential base 2
FLR	v	v	floor
FRC	v	v	fraction
KIL	v	v	kill fragment
LG2	s	ssss	logarithm base 2
LIT	v	v	compute light coefficients
LRP	v,v,v	v	linear interpolation
MAD	v,v,v	v	multiply and add
MAX	v,v	v	maximum
MIN	v,v	v	minimum
MOV	v	v	move
MUL	v,v	v	multiply
POW	s,s	ssss	exponentiate
RCP	s	ssss	reciprocal
RSQ	s	ssss	reciprocal square root
SCS	s	ss--	sine/cosine without reduction
SGE	v,v	v	set on greater than or equal
SIN	s	ssss	sine with reduction to [-PI,PI]
SLT	v,v	v	set on less than
SUB	v,v	v	subtract
SWZ	v	v	extended swizzle
TEX	v,u,t	v	texture sample
TXB	v,u,t	v	texture sample with bias
TXP	v,u,t	v	texture sample with projection
XPD	v,v	v	cross product

The core operations that make acceleration of Ward [1997] possible are LG2, EX2, TEX, TXB, RCP and POW [ARB02]. In later chapters we will discuss the details of how these operations are used in the implementation of our tone reproduction operator.

LG2 approximates the base 2 logarithm of the scalar operand and replicates it to all four components. Thankfully, the log base 2 of variable x can be related to the natural logarithm and log base anything with a proportionality constant. For example we can get ln(x) from log2(x) by the relation ln(x) = log2(x)/log2(e).

EX2 approximates two raised to the power of the scalar operand and replicates the approximation to all four components of the result vector. We can get an approximation to the exponential by computing EX2(x/ln(2)).

TEX takes the first three components of its source vector and maps them to texture coordinates s, t and r. These coordinates are used to sample from the specified texture target on the specified texture image unit in a manner consistent with its parameters.

TXB does the same thing as TEX except the fourth component is used to further set the level of detail mipmap bias of a sampled texture. This mechanism provides per fragment level of detail control.

POW approximates the value of the first scalar operand raised to the power of the second scalar operand and replicates it to all four components of the result vector.

RCP approximates the reciprocal of the scalar operand and replicates it to all four components of the result vector.

The use of fragment programs is straight forward and is much like using textures. A fragment program ID can be requested and then the program can be bound to become the current program. Fragment shader commands will then apply to the currently bound fragment program. A sample program is shown in Appendix A4 to illustrate what functions are used. Comments within the sample code provide explanations for what is happening.

4.5 Summary

The current set of tools available on graphics boards allows for a wide range of computations not restricted to 3D graphics. Image processing and other numerical methods can be implemented and accelerated using graphics hardware architectures. In the next chapter we detail how the capabilities of current commodity graphics hardware showcased here are used in the context of accelerating a tone reproduction operator.

Chapter 5

Hardware Acceleration

The tone reproduction operator that we build uses Ward's [1997] histogram adjustment dynamic range compression algorithm, as well as his glare, color and acuity models as a basis. Each component is accelerated either using current commodity hardware functionality or approximations to speed up calculations. The primary means of acceleration is the use of a central fragment program which receives and processes acuity, glare, color sensitivity and mapping data for display. Hardware functionality like mipmapping, multi-texturing are also used. In addition to acceleration we also build a novel dark and light adaptation model inspired by psychophysical data and the work of Pattanaik [1998].

In this section we discuss the details of each component of the algorithm, starting with a high level view. For a sketch of Ward's operator see the section on previous work. The algorithm is summarized in Figure 5.1.

Our explanation will follow the order of Figure 5.1. However, the order is not fixed. The veiling luminance can be added before or after the blurring function since it varies slowly over the course of the image. Also, the color sensitivity function can be applied anywhere after the veil is computed and before histogram adjustment. After discussion on the core operator we explore how the algorithm can be modified to include the time course of adaptation. Both dark and light adaptation are treated and offer significant improvements over previous work ([Ferwerda96],[Pattanaik00]).

Ward Overview

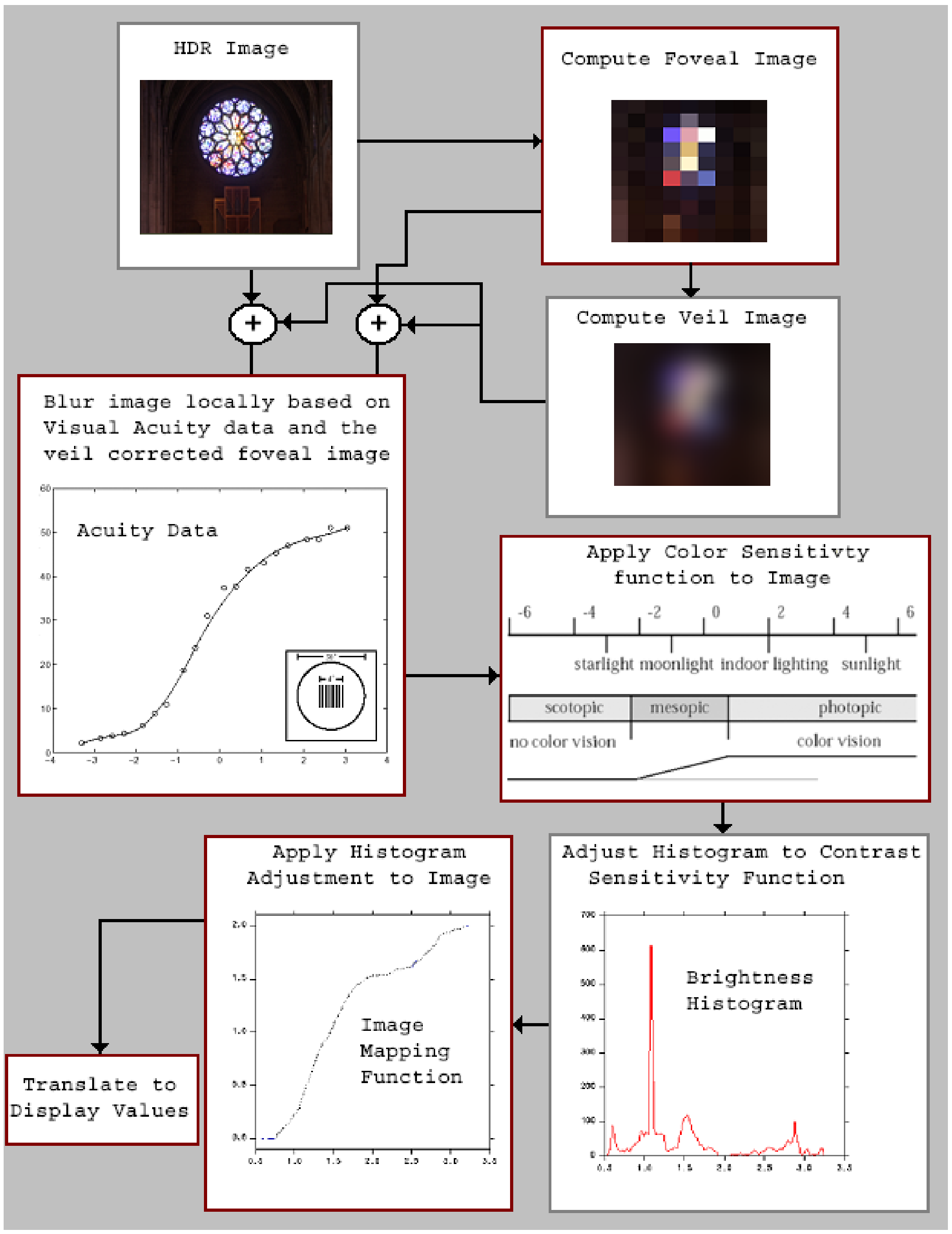

HDR Image
Compute Foveal Image
Compute Veil Image
Blur image locally based on Visual Acuity data and the veil corrected foveal image
Acuity Data
Apply Color Sensitivty function to Image
starlight
moonlight
indoor lighting
sunlight
scotopic
mesopic
photopic
no color vision
color vision
Adjust Histogram to Contrast Sensitivity Function
Brightness Histogram
Apply Histogram Adjustment to Image
Image Mapping Function
Translate to Display Values

Figure 5.1: An overview of Ward's histogram adjustment algorithm with glare, acuity and color sensitivity. Steps highlighted in red are those that benefit from the use of graphics hardware.

5.1 Input Images

Input is in primarily in the form of 2D floating point arrays. We have used Ward's rgbe format [RGBE] for storing HDR images, which are then read into arrays at startup before processing. Since we are using hardware, one additional piece of information is needed for each frame. A global scale factor must be chosen that normalizes the input into the range 0.0-1.0. The reason for this is that values in OpenGL hardware textures must be within this range. If no normalization factor is provided it could be quickly estimated by randomly sampling the image and picking the highest value encountered.

Input can also come from hardware based renderings. In this case the results will already be in the range 0.0-1.0 and the result must be made available for processing in our algorithm by rendering to a floating point P-Buffer and then binding this as a texture using the Render to Texture facility in OpenGL. However, in order for processing of color and acuity to be done correctly a scale factor that relates the rendered values to physical values must be provided.

Once in texture memory, input is displayed by texture mapping a screen-aligned grid of polygons. The resolution of the grid corresponds to the resolution of the foveal adaptation image because glare and acuity information are determined based on the foveal image and must be linearly interpolated to get data for each pixel of the original image. Storing glare and acuity data in the texture coordinates of the grid gives automatic interpolation in the graphics hardware. We will discuss the polygon grid in detail later in this chapter.

5.2 Foveal Image

All subsequent steps in the algorithm depend on the initial calculation of a *foveal image*, which is an averaged version of the original image with the property that each pixel subtends roughly one degree of solid angle. Each pixel corresponds to a potential foveal fixation point. It is assumed that adaptation for the best view occurs in the fovea and that this adaptation should be simulated over the extent of the image as if the observer is looking directly at each point in the image.

For an input image created using a linear perspective projection the foveal image resolution can be determined by the following formula:

$$S = 2\tan(\theta/2)/0.01745 \qquad \text{(Eq. 5.1)}$$

where S = width or height in pixels of the foveal image, θ = horizontal or vertical full view angle, and 0.01745 = the number of radians in one degree. Ward computes the foveal image by using a simple box filter.

In our operator there are three different ways to compute the foveal image. The first follows Ward and performs the box filtering in software. To do this quickly we use the Intel Image Processing Library function `ippiResize_32f_C3R`. The second method uses the graphics hardware exclusively and computes the foveal image as a side effect of creating a custom mipmap pyramid for use in acuity calculations. This process is shown in Figure 5.5. When a mipmap level with resolution above that of the foveal image is created a separate pass is executed to compute the foveal image at the correct resolution. The image is then read back from the P-Buffer into a software array. Then the mipmap creation continues as normal. The third way uses both hardware and software. Box filtered mipmaps can be automatically created by the graphics hardware with automatic mipmap creation routines. Then the mipmap level nearest the foveal image can be read back and resized in software. All three methods are very fast and thus execution time is not an issue. However, computing the foveal image completely in software is more effective because full accuracy is maintained. Reading back from a 16-bit depth texture can lead to luminance values of zero in the foveal image, which must be clamped to the minimum visible luminance. However, this artificially increases the dynamic range.

5.3 Veiling Luminance

Ward's veiling luminance calculations are directly from the work of Holladay [1926] and Moon and Spencer [1945]. Moon and Spencer's approach is appropriate for an operator that depends on a foveal adaptation image because they relate the effective adaptation luminance to the foveal average via the glare source position and illuminance. Veiling luminance is calculated for the low resolution foveal image and then added back to the foveal image to get an effective adaptation image for use in contrast, color and acuity models. To add veiling luminance to the original image the foveal image values are bilinearly interpolated to find individual pixel values.

An overview of how Ward calculates veiling luminance is given in Figure 5.2. To compute the veiling luminance Ward uses a discrete formula adapted from Moon and Spencer [1945]. For large foveal images (images with large view angles) this calculation is the most expensive operation due to the $O(n^2)$ running time. Although the veiling luminance calculation would be much faster than if the operator was applied to the entire image as in Spencer [1995], it is still inadequate for real-time

Veil formulation of Moon and Spencer

$$L_a = 0.913\,L_f + \frac{K}{\pi}\iint_{\theta>\theta_f} \frac{L(\theta,\phi)}{\theta^2}\cos(\theta)\sin(\theta)\,d\theta\,d\phi$$

where:

L_a	= corrected adaptation luminance (in cd/m^2)
L_f	= the average foveal luminance (in cd/m^2)
$L(\theta,\phi)$	= the luminance in the direction (θ,ϕ)
θ_f	= foveal half angle, approx. 0.00873 radians (0.5°)
K	= constant measured by Holladay, 0.0096

Ward

$$L_{vi} = 0.087 \cdot \frac{\sum_{j\neq i} \frac{L_j \cos(\theta_{i,j})}{\theta_{i,j}^2}}{\sum_{j\neq i} \frac{\cos(\theta_{i,j})}{\theta_{i,j}^2}}$$

To compute veiling luminance for a given foveal sample convert integral to a sum over peripheral samples

where:

L_{vi}	= veiling luminance for fixation point i
L_j	= foveal luminance for fixation point j
$\theta_{i,j}$	= angle between sample i and j (in radians)

Add veil to original image

$$L_{pvk} = 0.913\,L_{pk} + L_v(k)$$

where:

L_{pvk}	= veiled pixel at image position k
L_{pk}	= original pixel at image position k
$L_v(k)$	= interpolated veiling luminance at k

Figure 5.2: Overview of Ward’s veiling luminance operator.

performance in many cases. Thus for our purposes, changes must be made to speed up the calculation. These can be in the form of optimizations and/or approximations.

The first property to notice about this operator is that it is effectively a convolution operation with a kernel that is not constant over the extent of the image (Figure 5.3). This fact keeps us from being able to store just one fixed kernel which can be reused for every foveal sample. Instead, the cosine terms must be recalculated for each sample over the entire image. It would be a big gain to avoid this. To

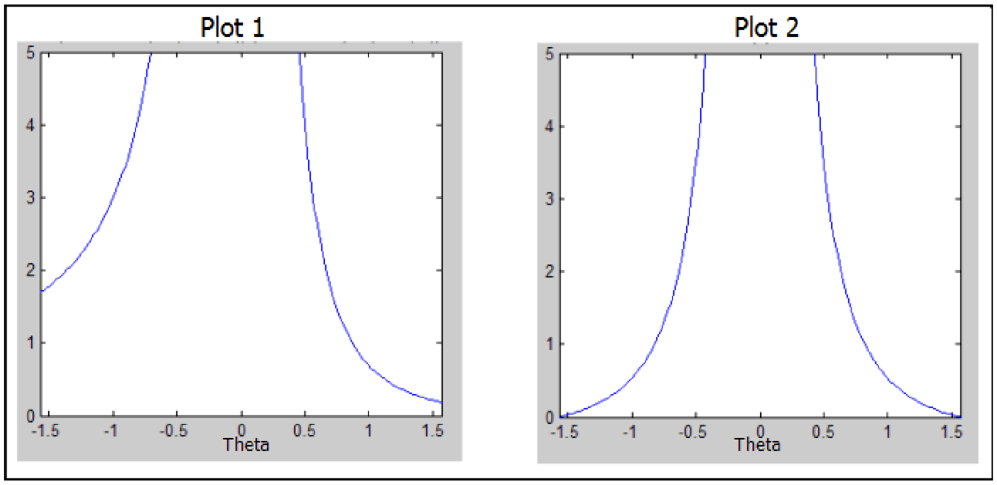


Figure 5.3: Two slices of the veiling luminance effective convolution kernel. Plot 1 shows the filter for a foveal sample 0.4369 radians from the center of the image. Plot 2 shows the filter for a pixel at the center of the image.

accomplish this we combine the approach of Ward [1997] and Spencer [1995] by computing veiling luminance with the foveal image but using Spencer's fixed convolution kernel.

Despite the improvements in speed one gets from implementing the veiling luminance calculation as a convolution with a fixed filter, performance is often still not real time for large foveal image resolutions. Much better results can be had by intelligently choosing the kernel size and foveal image resolution used for convolution. We could have a constraint on the computation time which determines what size kernel can be used or how much the foveal image can be downsized before the convolution is done. Either choice will introduce some error. In our experience the error due to an insufficiently large kernel is more significant because the extents of bloom are limited if the kernel is too small. Using a downsized foveal image gives much better results.

To balance both of these dimensions we need to determine on the fly how large of a kernel is needed for a given foveal image. This determination should be able to predict the perceptual quality of the result. For example, will the scene viewer notice the difference between a larger or smaller kernel. Given this information we can then check if the amount of computation is tolerable. If not, then we downsize the foveal image and if the foveal image is still bigger than the kernel size we again compute the necessary kernel size. Otherwise, we use the chosen kernel with the downsized foveal image. Finally, if a downsized image is used, we get the approximate veiling luminance image by bilinearly interpolating the lower resolution image.

To determine the kernel size needed for a given foveal image we use a perceptually-based metric. The idea is that we want a gauge of how far from a given image pixel the kernel must extend such that the luminance veil due to blooming is noticeable by a human observer. If the foveal image doesn't have too many outliers then a worst case scenario is to assume the pixel in question is at the minimum luminance for the image. Then we measure the contribution from a solitary foveal sample at the maximum luminance some number of pixels away. If the veil is below threshold for just noticeable contrast differences then the kernel is too large. If the correction is above the threshold, the kernel is too small. We want the kernel that gives us the closest result to the threshold of noticeable difference. In pseudocode, the procedure is as follows:

```
SelectKernelSize()
   For Discrete values of the normalized kernel radially
      outward 1…N
   Compute the weighted contribution: Lwmax*kernelVal
   Compute the veil correction: (contrib + 0.913*Lmin) – Lmin
   Find visibility threshold for a background Luminance Lmin
      using contrast sensitivity data
   Find kernel value that gives the closest match to threshold
   Record distance from center of kernel in pixels which is the
      kernel resolution needed
End
```

Given the use of a fixed kernel and the perceptually based metric just described, we can manage the computation time so as to reach a target frame rate and still maintain quality results. The metric proved useful only when foveal image resolution was high, otherwise a full convolution was not too expensive for real time performance. Finally, after determining kernel size the covolution is done using the highly optimized Intel Image Processing Library and the veil is added to the foveal adaptation image. Compositing the veil with the original HDR image is done by storing veil image samples in texture coordinates and then sampling them in the fragment shader. Multitexturing must be used because acuity data is also stored in texture coordinates. It should be noted that using texture coordinates to store the veil data is more favorable than using vertex color because it affords more precision.

5.4 Local Acuity Blurring

Ward bases his acuity determination on Shaler's [1937] data which measures the relationship between adaptation level and foveal acuity through subject studies. He uses the following functional fit to Shaler's data:

$$R(L_a) \approx 17.25 \cdot \arctan(1.4 \cdot \log10(L_a) + 0.35) + 25.72 \qquad \text{(Eq. 5.2)}$$

where $R(L_a)$ = visual acuity in cycles/degree, L_a = local adaptation luminance. Local acuity is determined for each pixel of the foveal adapation image. To allow for local acuity variation Ward implements blurring using a mipmap structure [Williams83] and interpolation. However, in this case the levels of the mipmap are the physical luminance values.

Mipmap level determination for each foveal sample is done using the camera parameters for the scene and the resolution of the image being processed. The lowest level of the mipmap is the original image. This image is a sampled version of the scene, thus as long as the viewer's acuity is above this sampling rate, we can use the base mipmap level. This baseline sampling rate is given by:

$$R = 0.5 \cdot resolution / \theta \qquad \text{(Eq. 5.3)}$$

where R = the sampling rate in cycles/degree, *resolution* = the x or y pixel resolution of the image, and θ = the x or y axis viewing angle in degrees. If the viewer's local acuity falls below the sampling rate of the base level we determine the mipmap level by equating the viewer's acuity with the corresponding, possibly interpolated, mipmap level. For hardware mipmapping we note that mipmap level is given as a value m_l

from 0.0 to the highest level (for example, for a base resolution of 512x512 m_l ranges continuously from 0.0 to 9.0). m_l can be determined with the relation:

$$x = \frac{R}{2^{m_l}} \rightarrow m_l = \log 2(R/x) \qquad \text{(Eq. 5.4)}$$

Where,

$x =$ the acuity at a given foveal sample.

To accelerate the acuity calculations (Figure 5.4) we can take advantage of hardware mipmapping or our own custom mipmap generation (Figure 5.5). Automatic hardware mipmapping has the advantage that it is faster and easy to use. However, only box filtering is available. Our custom mipmapping requires multiple passes but allows the use of custom filters for downsizing subsequently higher mipmap levels. Once the mipmaps are formed the mipmap bias (level) can be specified for each foveal sample by setting an unused component of the texture coordinates to the bias values. Then, when further processing is done for compositing the veil, calculating color sensitivity, and histogram adjustment in the fragment shaders, we will have access to interpolated bias values over the entire image (per pixel level of detail control). This is due to the fact that texture coordinates are automatically interpolated in hardware. In the fragment shaders, we use the TXB operation as discussed in the hardware capabilities section to sample the mipmap and pass our texture coordinate as a level of detail bias.

Custom mipmap generation including a step for computing the foveal adaptation image is illustrated in Figure 5.5. It is a multi-pass and efficient algorithm for computing mipmaps with custom filtering for successive mipmap levels done in the fragment/pixel shaders. Input is either a HDR texture, HDR video stream or

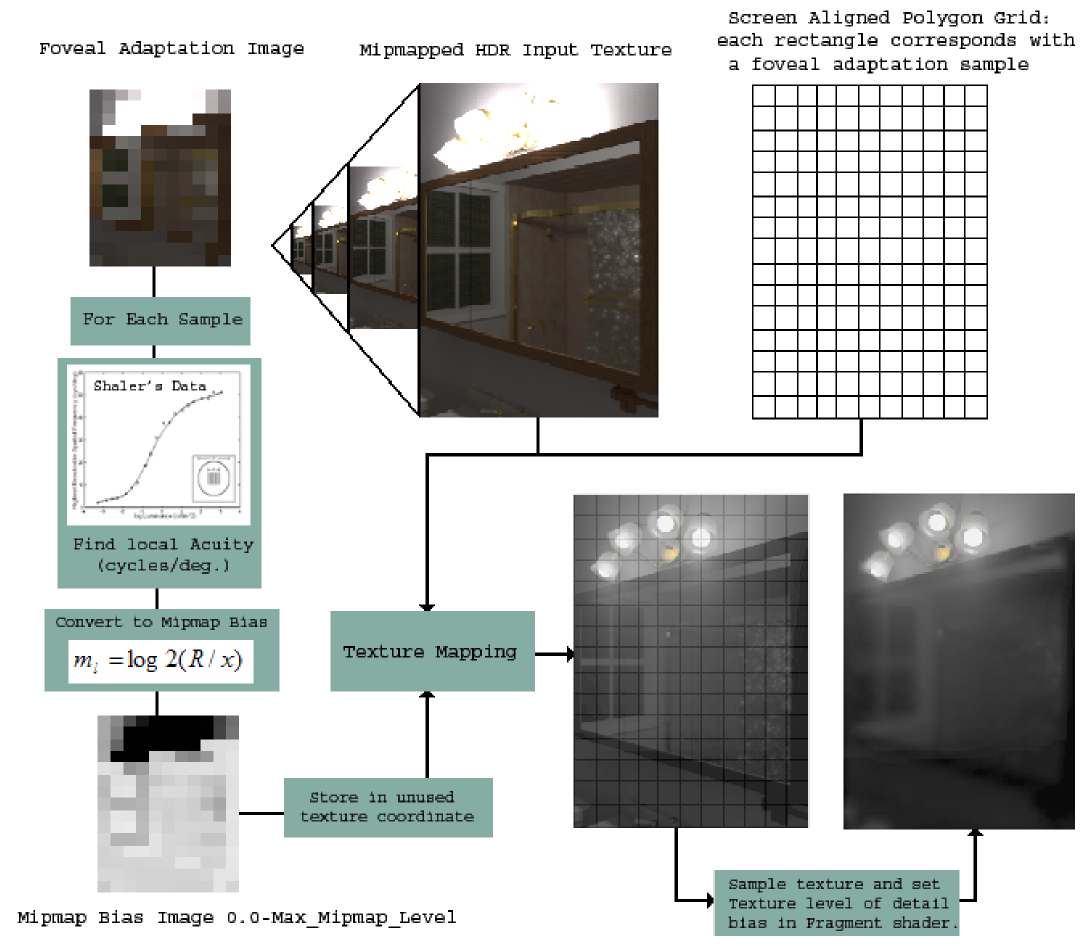

Foveal Adaptation Image
Mipmapped HDR Input Texture
Screen Aligned Polygon Grid: each rectangle corresponds with a foveal adaptation sample
For Each Sample
Shaler's Data
Find local Acuity (cycles/deg.)
Convert to Mipmap Bias
$m_i = \log 2(R/x)$
Texture Mapping
Store in unused texture coordinate
Mipmap Bias Image 0.0-Max_Mipmap_Level
Sample texture and set Texture level of detail bias in Fragment shader.

Figure 5.4: An Illustration of hardware accelerated local acuity blurring based on Ward's [1997] acuity calculations. The foveal adaptation image from Ward's histogram adjustment algorithm is converted to a mipmap bias image using Shaler's HVS acuity data. This bias image is then stored at the vertices of a screen aligned polygon grid in texture coordinates. A mipmaped HDR texture input is then texture mapped onto the grid and the mipmap level is chosen for each pixel by utilizing fragment shaders.

rendered in hardware. If input is a texture or video stream it is used to set the lowest mipmap level directly from software or texture memory. If the input is hardware rendered then it is rendered to a floating point P-Buffer and then read back using glCopyTexSubImage2D( ), which is an operation that keeps all data transfer on the card and thus has little overhead.

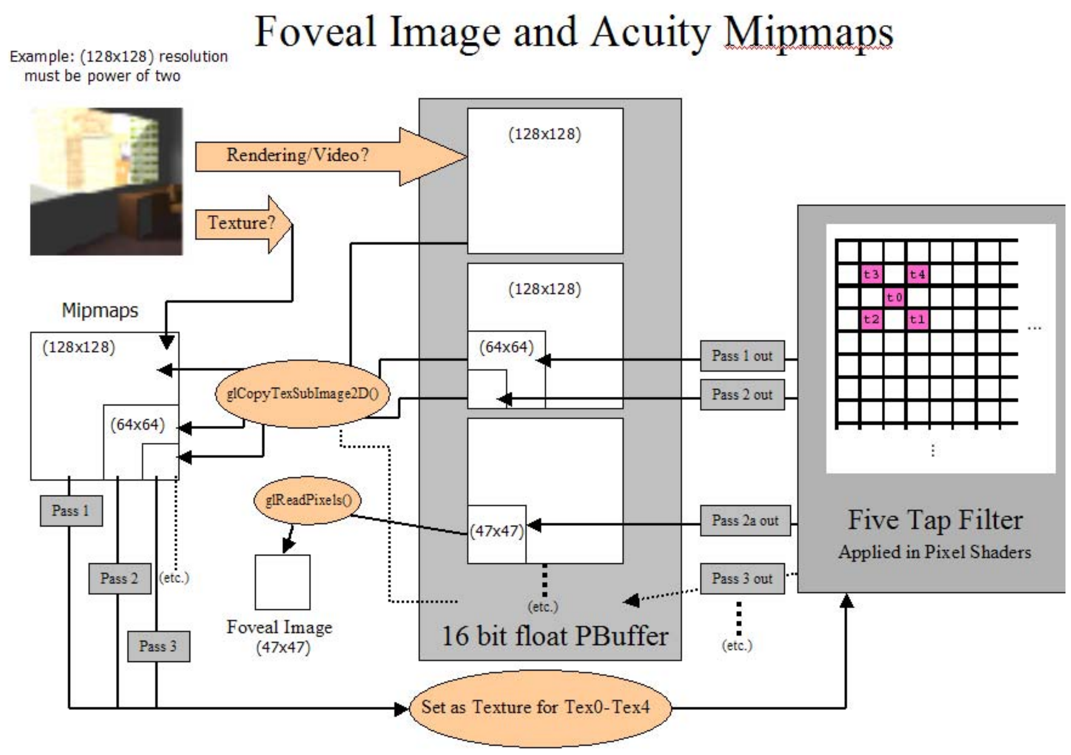


Figure 5.5: Custom acuity mipmap generation algorithm shown for a sample texture of 128x128. Foveal image formation is also illustrated in Pass 2a.

Once the lowest mipmap level is set, successive levels are derived by rendering to a P-Buffer in a region that is half the resolution of the current level and then reading back into the mipmap pyramid via glCopyTexSubImage2D( ). Fragment shaders are used to sample the texture and filter with whatever weighting is desired. To get samples at each pixel that include all surrounding pixels we offset the texture coordinates of our starting level to allow overlap. If one of the overlap textures does not cover a given pixel the edge values are extended by setting the texture parameters `GL_TEXTURE_WRAP_T and GL_TEXTURE_WRAP_S` to `GL_CLAMP`. An illustration of a five sample version is show in Figure 5.6. Nine samples are also possible and this is what we use in our final implementation coupled with gaussian filter weights. The empty white P-Buffer pixels as seen in the diagram will have repeated values from the edge of the texture. To make sure that sampling is adequate `GL_TEXTURE_MIN_FILTER` should be set to `GL_LINEAR`. Otherwise, the hardware will simply point sample the four incoming texels giving poor results.

When the mipmap algorithm gets to a level that bounds the resolution of the foveal adaptation image between the current and next mipmap level a separate pass is performed to create the foveal image. This is shown in Figure 5.5. Only a part of the P-Buffer at the resolution of the foveal image is rendered. Then the result is read back into processor memory using glReadPixels( ). This is typically a slow operation but the foveal image is very low resolution so the overhead is minimal.

5.5 Color Sensitivity

To simulate the loss of color vision at mesopic and scotopic adaptation levels Ward uses the same technique as Ferwerda [1996], interpolating between a scotopic monochromatic response function and a photopic chromatic response function. In the mesopic range of adaptation levels, both rods and cones contribute to the appearance of the scene and a linear ramp between the scotopic and photopic responses is calculated. The low end of the mesopic range where cones are just getting enough

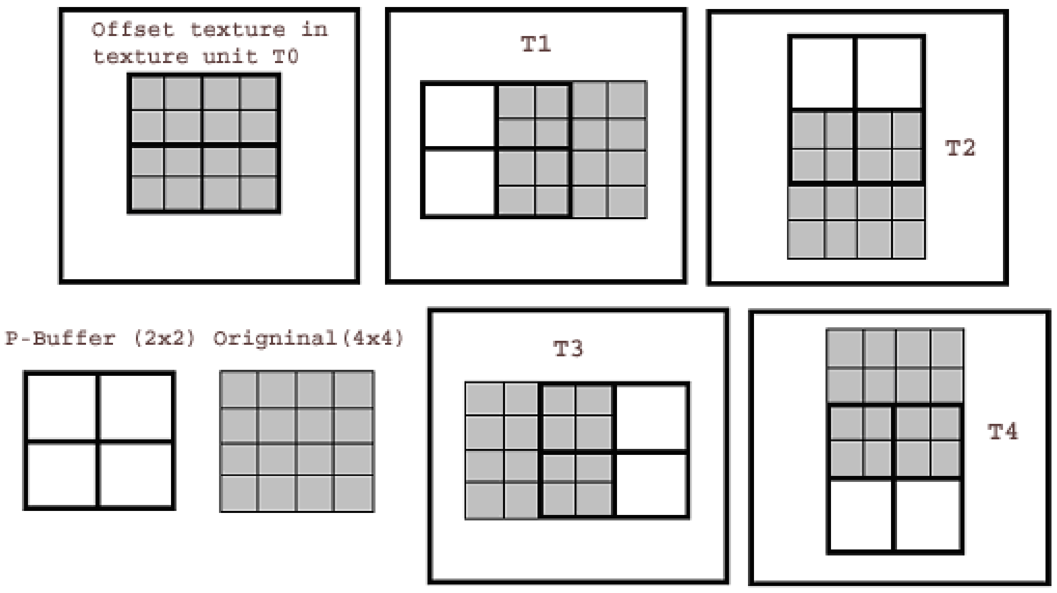


Figure 5.6: Example illustrating the formation of mipmaps using fragment shaders and the P-Buffer going from a 4x4 texture to the next mipmap level (2x2). Each frame shows how the texture is offset in relation to the P-Buffer before sampling in the fragment shaders. Each sample is then given a weight and combined with the others to get the final P-Buffer pixel which can then be read back into texture memory to fill the next mipmap level.

light is 0.0056 cd/m^2 and below this only the monochromatic scotopic luminance is used. The high end of the mesopic range is approximately 5.7 cd/m^2, after which the rods are no longer effective and only the chromatic photopic response is used [Ward1997].

The scotopic luminace at each pixel is estimated by using an approximation to the least squares fit to the colors on the MacBeth ColorChecker Chart:

$$Y_{s\cot} = Y \cdot \left[1.33 \cdot \left(1 + \frac{Y+Z}{X}\right) - 1.68\right] \qquad \text{(Eq. 5.5)}$$

where $Y_{s\cot}$ = the scotopic luminance and X,Y,Z = photopic color, CIE 2^o observer. If some other set of R,G,B primaries is used, we adopt Ward's approximation based on R,G,B values. However, because there is no standard for R,G,B primaries the approximation is less reliable.

Color sensitivity calculations are included in the main fragment shader, which also includes the addition of glare to the original source image, local acuity blurring, and the final application of the tone reproduction curve. Color sensitivity is done using a series of vector MUL (multiply) operations to get scotopic and photopic luminances and a LRP (linear interpolation) to get the final corrected color. (See the Fragment Shader section, which gives the full fragment shader code. The comments included in the code explain the necessary operations.)

5.6 Histogram Adjustment

Histogram adjustment based on human contrast sensitivity creates a mapping for an HDR image that both compresses the image data into a displayable range and matches display contrast visibility with world observer visibility at threshold. The method is based on histogram equalization. However, normal histogram equalization expands contrast in all regions of an image while Ward's work attempts to match world and display contrasts and thereby accurately reproduce visibility. Histogram adjustment uses the framework of histogram equalization but adds a constraint based on human contrast sensitivity data that effectively compresses under-represented regions of the histogram without expanding over-represented ones. This process simultaneously respects the contrast sensitivity limits of a human observer. The result of the histogram adjustment is a global tone mapping function that is applied to the HDR image being processed. The histogram adjustment step is sufficiently fast in software so that no acceleration is needed. Furthermore, the algorithm is too complicated to be accelerated using the operations available on graphics hardware. We therefore use Ward's histogram adjustment algorithm unchanged and concentrate on speeding up the application of the final mapping function to the HDR image being processed.

Histogram adjustment begins with the creation of an approximate brightness histogram from the corrected foveal image. The corrected foveal image has the veiling luminance correction added since this will affect the viewer's adaptation state. Once formed a constraint on the histogram bin counts is enforced. We first explain the constraint and then the application of the constraint to the histogram.

Let, $\Delta L_t(L_a) =$ "just noticeable difference" for adaptation level L_a, which is the detection threshold for luminance contrasts at a given adaptation level [Ferwerda96], $L_w =$ world luminance (in cd/m^2) and $L_d =$ display luminance (in cd/m^2). To ensure that contrasts on the displayed image are not more noticeable than in the real world, Ward constrains the slope of the global mapping to the ratio of the adaptation thresholds for the display and world:

$$\frac{dL_d}{dL_w} \le \frac{\Delta L_t(L_d)}{\Delta L_t(L_w)} \qquad \text{(Eq. 5.6)}$$

Then, substituting in the derivative of the histogram equalization function and rearranging Ward's equation yields a constraint on the bin counts:

$$f(B_w) \le \frac{\Delta L_t(L_d)}{\Delta L_t(L_w)} \cdot \frac{T\Delta b L_w}{[\log(L_{d\max}) - \log(L_{d\min})]L_d} \qquad \text{(Eq. 5.7)}$$

Where,

B_w = world brightness (in cd/m^2)

T = the total number of adaptation samples

Δb = the bin step size in $\log(cd/m^2)$

$f(b_i)$ = frequency count for the histogram bin at b_i

Enforcing the constraint (Equation 5.7) takes the form of truncating bin counts that exceed the given ceiling. However, this truncation changes T (the total number of adaptation samples) which in turn changes the constraint ceiling. Thus, Ward iterates the truncation procedure and imposes a tolerance criterion, which says: stop when fewer than 2.5% of the original samples exceed the ceiling.

Once histogram adjustment is complete the final mapping of the input image is completed. This is accomplished using the fragment shaders available in the OpenGL 1.4 specification discussed in Chapter 4 and, most importantly, the dependent texture read operation. The form of the mapping is the same as for histogram equalization:

$$B_{de} = \log(L_{d\min}) + [\log(L_{d\max}) - \log(L_{d\min})] \cdot P(B_w) \qquad \text{(Eq. 5.8)}$$

Where,

B_{de} = computed display brightness, $\log(L_d)$.

$P(B_w)$ = the cumulative distribution function of world pixel brightness.

The cumulative distribution function is computed in software and then stored as a one dimensional texture in hardware. It is then sampled using a dependent texture read in the fragment shaders to accomplish the mapping. However, before the output can be displayed the result must be converted to luminance and biased into 0.0-1.0 for display. Conversion from brightness B_{de} to luminance requires a simple exponentiation. Then the result is biased into 0.0-1.0 using:

$$L_{out} = \frac{L - L_{d\min}}{L_{d\max} - L_{d\min}} \qquad \text{(Eq. 5.9)}$$

Next an appropriate exposure value must be chosen to set the actual range of the mapping, since it is not always appropriate that the maximum luminance map to the highest display value of 1.0. For example, for low adaptation levels the highest value is significantly reduced. To chose an appropriate value we use the same method as Ward [1994] and Ferwerda [1996]. A global scale factor is chosen assuming a single well defined adaptation level L_{wa} (in this case the average log scene luminances). A scale factor is chosen such that a just noticeable different for a world observer at adaptation level L_{wa} is matched to a just noticeable difference at the midrange display value. To fix the displayed range the maximum adaptation level of the foveal image is mapped to it's corresponding display value using the scale factor. If it is less than one, then the maximum displayed value is reduced, thereby setting an appropriate exposure.

Details of how the mapping is done in hardware are provided in the next section. The discussion relates this section, the acuity section and color sensitivity section to the actual fragment shader code used.

5.7 Fragment Shader

Details on fragment shader syntax and parameters can be found in the OpenGL 1.4 specification [OpenGLSpec]. The fragment shader begins by naming attributes and parameters for later use in the program. These contain all relevant input data. We first identify the fragment program beginning with:

```
!!ARBfp1.0
```

This tells us that it is an ARB(OpenGL Architecture Review Board) fragment program version 1.0. Next we name our attributes.

```
ATTRIB tex = fragment.texcoord[0];

ATTRIB veil = fragment.texcoord[1];
```

‘tex’ is the texture coordinates of texture unit 0 and it references our input HDR texture which has been normalized into the range 0.0-1.0 by dividing the input by MAX_LUM (the maximum luminance in the scene). The *w* component contains the texture mipmap level bias calculated in Equation 5.4 and shown in Figure 5.4. ‘veil’ is the veil correction image computed by our glare calculations and stored in the texture coordinates of texture unit one. After attributes we name parameters.

```
PARAM misc = {0.4545454, 0.0, 0.0, 0.0};

PARAM photopic = program.local[0];

PARAM scotopic = program.local[1];
```

‘misc’ stores micellaneous constants. The only one we use is misc.x which stores a gamma correction factor for use after the final mapping. ‘photopic’ stores the conversion vector to convert RGB’s to photopic luminance and ‘scotopic’ stores the conversion vector to convert RGB’s to scotopic luminance.

```
PARAM bmap = program.local[2];

PARAM mapconst = program.local[3];

PARAM whtefficacy = program.local[4];

PARAM photopicw = program.local[5];
```

‘bmap’ contains constants that map the output luminance into the final 0.0-1.0 display range as seen in Equation 5.9. However, we divide the numerator into two terms so that the operation can be performed as a MAD (multiply add) in the fragment shader. ‘mapconst’ contains constants to map texture input values into 0.0-1.0 before the final mapping (Equation 5.8) is applied via a dependent texture read. It also contains two constants (the top and bottom of the mesopic range of vision) used in color appearance mapping for each pixel. We discuss this later on in the code. ‘whtefficacy’ holds the luminous efficacy for white light used in calculating luminance. ‘photopicw’ has a premultiplied version of ‘photopic’ and ‘whtefficacy’. The premultiplication is done to save fragment program operations.

```
OUTPUT outColor = result.color;

TEMP colin, bLookUp, Lscot, Lphot, cLookUp, Lphotw, newColin ;
```

Finally, we name the output variable ‘outColor’ and initialize some constants for use in the program. The use of each constant will become clear as we discuss the remainder of the fragment shader code.

```
TXB colin, tex, texture[0], 2D;
```

The first operation to perform is to sample the HDR texture and set the correct per pixel mipmap bias. This can be done using the TXB operation which samples a texture using the *t,u,v* texture coordinates and sets the mipmap bias with the *w* component. The result is stored in ‘colin’.

```
MUL colin, colin, 0.913;

MAD colin, veil, 0.087, colin;
```

Next we composite the veiling luminance calculation as illustrated in Figure 5.2.

```
DP3 Lphotw, colin, photopicw;

MAD_SAT cLookUp, Lphotw, mapconst.zzzz, mapconst.wwww;

DP3 Lscot, colin, scotopic;

LRP newColin, cLookUp, colin, Lscot;
```

Once interpolated veiling luminance has been added to the input we must choose the color desaturation parameter. For each pixel we map between the top and bottom of the mesopic range 0.0056 and 5.6 cd/m^2 based on the photopic luminance (Equation 5.5). Values above 5.6 cd/m^2 map to 1.0 and those below 0.0056 cd/m^2 map to 0.0. In between we have a linear ramp from 0.0-1.0. This is implemented first by finding the photopic luminance then mapping it over the prescribed range while clamping values above 1.0 and below 0.0. The mapping constansts are divided by MAX_LUM to cancel out the normalization of the HDR input (Equation 5.10).

$$cLookUp = \frac{L_{phot} - MesBottom}{MesTop - MesBottom} \quad \text{where} \qquad \text{(Eq. 5.10)}$$

$$MesBottom = 0.0056 cd/m^2 / MAX\_LUM$$
$$MesTop = 5.6 cd/m^2 / MAX\_LUM$$

Once the desaturation parameter is set we calculate the scotopic luminance and linearly interpolate between it and the input color to get our new input color which is stored in 'newColin'.

```
DP3 Lphotw, newColin, photopicw;
```

Then we calculate the photopic luminance with our new color in preparation for application of the global operator derived through the histogram adjustment.

```
LG2 bLookUp, Lphotw.x;                   // log2(L/MAX_LUM)
ADD bLookUp, bLookUp, mapconst.yyyy;     // add (BlMax - Bwmin)
MUL_SAT bLookUp, bLookUp, mapconst.xxxx; // multiply by 1/(Bwmax-Bwmin)
```

To properly map the input luminances we apply Equation 5.4. However, this requires having brightness values to index the cumulative distribution function. In the fragment shaders we only have access to log base two instead of the natural logarithm used by Ward as a brightness approximation. The two are related by a global constant which cancels out when brightness is normalized into 0.0-1.0 before sampling the cumulative distribution function. The normalization proceeds in three steps. First we take log2(L), which is actually log2(Lactual/MAX_LUM). Then we normalize with a multiply and add. The MAX_LUM will drop out because our first ADD operation subtracts off the contribution from MAX_LUM to the logarithm of L.

```
// Pass two
TEMP weight,out, Pb, Bnew, Lnew, rcpLphot, Bnewmul;
TEX_SAT Pb, bLookUp, texture[2], 1D;
```

```
MAD Bnew, Pb, bmap.xxxx, bmap.yyyy;";        // Equation 5.8
```

With the normalized brightness calculated in pass one we now perform a dependent texture read which samples the cumulative distribution function for each pixel signaling the start of pass two. This is stored in 'Pb'. Then we apply Equation 5.8 to get our display brightness values.

```
EX2 Lnew, Bnew.x;                            // convert to Lnew
MAD_SAT Lnew, Lnew, bmap.zzzz, bmap.wwww;    // Bias into 0.0-1.0
RCP rcpLphot, rcpLphot.x;                    // compute 1/Lold
MUL Lnewmul, Lnew, rcpLphot.xxxx;            // compute Lnew*(1/Lold)
MUL Lnewmul, Lnewmul, scotopic.wwww;         // apply exposure
MUL out, newColin, Lnewmul;
```

Before we can output the result we must first convert brightness to luminance by exponentiating and biasing the result into the displayable range of 0.0-1.0 using Equation 5.9. Then, to apply the final mapping we divide out the original luminance value of each incoming pixel and replace it with the newly mapped luminance value. However, we first multiply by an appropriate exposure value (discussed in the Histogram Adjustment section).

```
POW outColor.x, out.x, misc.x;
POW outColor.y, out.y, misc.x;
POW outColor.z, out.z, misc.x;
END
```

Finally, the output is gamma corrected and sent to be displayed.

Chapter 6

Adaptation Time Course

The advantages of a real time operator become clear when we start to process image streams. As described in Chapter 3, adaptation does not happen instantaneously. Thus, to provide the correct visual appearance when processing image streams we must calculate the effects of light and dark adaptation.

6.1 Dark Adaptation

We first consider dark adaptation within the context of the histogram adjustment algorithm. Figure 6.1 shows the time course of dark adaptation measured by Hect [1934] and borrowed from Ferwerda [1996]. Our time course model closely follows this published data to ensure perceptual validity. In the graph shown an observer was first adapted to a high background level of illumination and then put into total darkness. The plot gives us the detection threshold as a function of time in the dark. It is the combination of two curves, one for the cones and one for the rods. The envelope of the two curves determines the dominant detection threshold. Although only one set of curves is shown in Figure 6.1 there are a family of curves, one for each potential starting adaptation level. It is commonly assumed that each curve can be approximated by a shift of a single curve in time. That is, if the experiment starts the observer at a lower adaptation level, the effective starting threshold is simply at a point lower on a curve where the observer started at a higher adaptation level.

We also adopt this assumption which is vital to our new time course algorithm which depends on the existence of a single invertible time course function.

Since our dark adaptation model is in the context of a visibility matching tone reproduction operator it should influence how JND's in the scene are mapped to JND's on the display. Let us first consider how this might be accomplished given a single adaptation level. Assuming an observer starts at adaptation level $L_{wa1,}$ this corresponds to a specific dark adaptation curve. Since all curves can be found from a curve that maps a very high adaptation level to a very low level, we can find a point along this single curve that is our starting threshold $\Delta L_t(L_{wa1})$. This is the current just noticeable difference at world adaptation level $L_{wa1.}$ It can be straight forwardly found by using the contrast sensitivity curves in Figure 6.2 which gives us detection threshold as a function of background illumination level. For background information

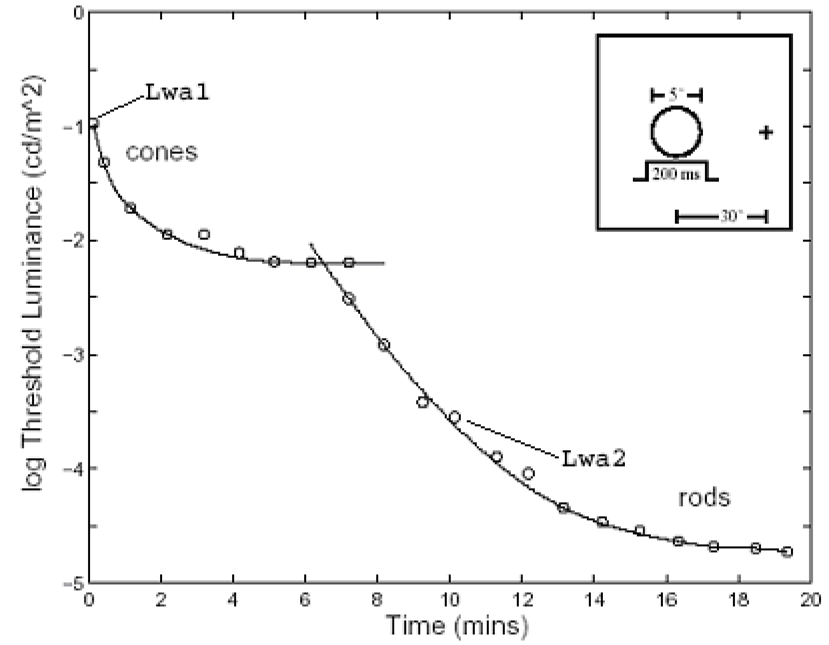


Figure 6.1: The time course of dark adaptation from Ferwerda [1996].

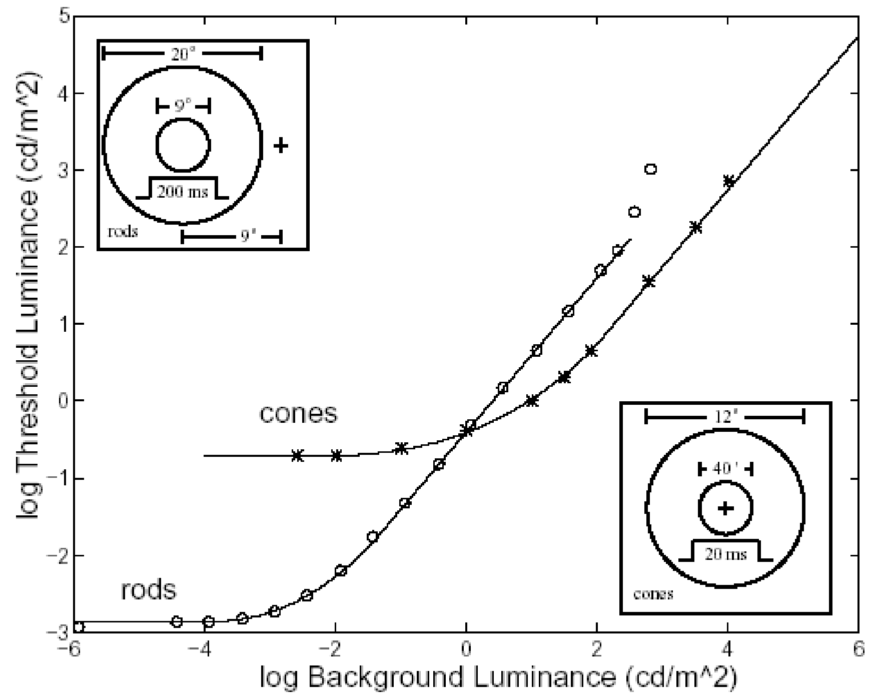


Figure 6.2: A model of detection thresholds of the range of adaptation levels.

and review of the experiments used to gather this data see Ferwerda [1996]. Now, suppose that we jump to adaptation level L_{wa2} and it is lower than L_{wa1}. This adaptation level will similarly place us at some detection threshold $\Delta L_t(L_{wa2})$ on our dark adaptation curve. If the dark adaptation curve is invertible then each of these JND's will correspond to a given time in the dark adaptation process. To track the correct world JND over time we simply start at time t_o (the time corresponding to $\Delta L_t(L_{wa1})$) and move forward in time over each frame of our animation or HDR video and look up the new JND at the new time. This will continue until the time corresponding to L_{wa2} is reached or L_{wa2} moves above the adaptation level associated with the current JND. During this process L_{wa2} can change and its current state is always tracked.

Relating the dark adaptation framework discussed for a single adaptation level to the histogram adjustment algorithm is fairly straight forward. Recall that to ensure that contrasts on the displayed image are not more noticeable than in the real world, Ward constrains the slope of the global mapping function to the ratio of the adaptation thresholds for the display and world (Equation 6.1):

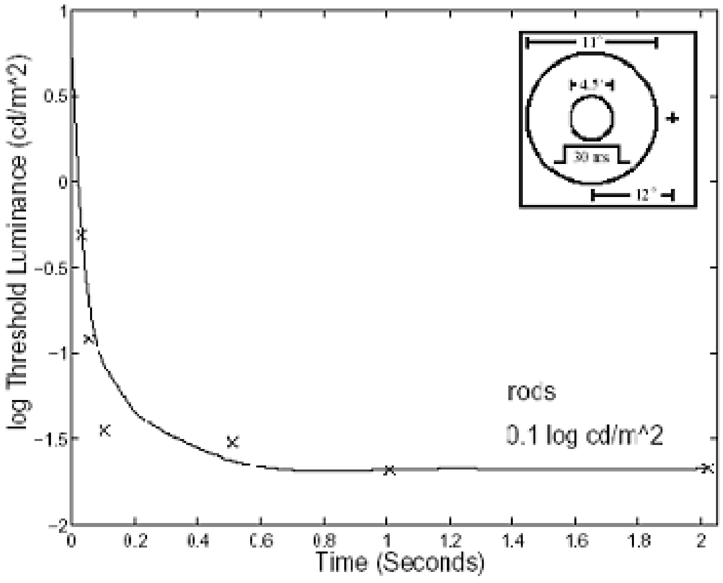
$$\frac{dL_d}{dL_w} \leq \frac{\Delta L_t(L_d)}{\Delta L_t(L_w)} \qquad \text{(Eq. 6.1)}$$

This is computed for each adaptation level in the scene, the number of which is the number of bins in the brightness histogram. thus, to create the correct mapping function at any moment in time we apply our method for a single adaptation level to each level considered in the brightness histogram. The time course model controls the world JND in the denominator of Equation 6.1. In this way we guarantee direct correspondence with the measured data of Figure 6.1 and correct use of JND's for mapping from world luminances to display luminances. In addition, because each luminance population considered is mapped differently we have an effective local adaptation model where different parts of a scene will behave based on the local luminance levels.

6.2 Light Adaptation

When we go from a dark adapted stated to a much higher adaptation level, such as going from a dark movie theater out into daylight, the world appears bright and almost painfully glaring. However, over time sensitivity is restored and the light no longer seems as intense. This phenomenon is known as light adaptation and it can be described by the curves shown in Figure 6.3. The plots show threshold luminance as a function of time for observers who have been adapted to the dark and then exposed to a large background field of illumination. Light adaptation for rods is shown in Figure 6.3a. The plot indicates that the rod system adapts very quickly to the new illumination level. More than 80% of sensitivity is recovered in the first 2 seconds [Ferwerda96]. Figure 6.3b shows light adaptation for the cone system in a similar experiment. The data indicates that cones adapt much more slowly than do the rods. A minimum threshold is reached after about three minutes of adaptation and a fully adapted state is reached after about 10 minutes. The threshold rises from the minimum due to interactions between neural and photochemical processes in adaptation. The cones dominate our visual experience for changes in illumination from scotoptic or mesopic to photopic jumps in adaptation level and from photopic to photopic jumps. Light adaptation in the rods only becomes important for jumps from one scotopic to another scotopic level. To proceed with our light adaptation operator we must have a single curve that gives us threshold over time because the histogram adjustment algorithm does not consider rods and cones separately. We choose the cone light adaptation curve as our basis because it covers the more common and noticeable transitions one might encounter. Light adaptation in the rods is modeled using the cone curves, which is not accurate with regards to the timing of adaptation but otherwise captures the characteristics of adaptation.

Light adaptation is modeled in the same framework as dark adaptation. The world JND's in the histogram adjustment constraint (Equation 6.1) are chosen using published data for light adaptation from Baker [1949]. As before, each luminance population of the histogram is treated separately, giving us local adaptation effects. The difference lies in how world JND's are derived from the raw data in Baker [1949].



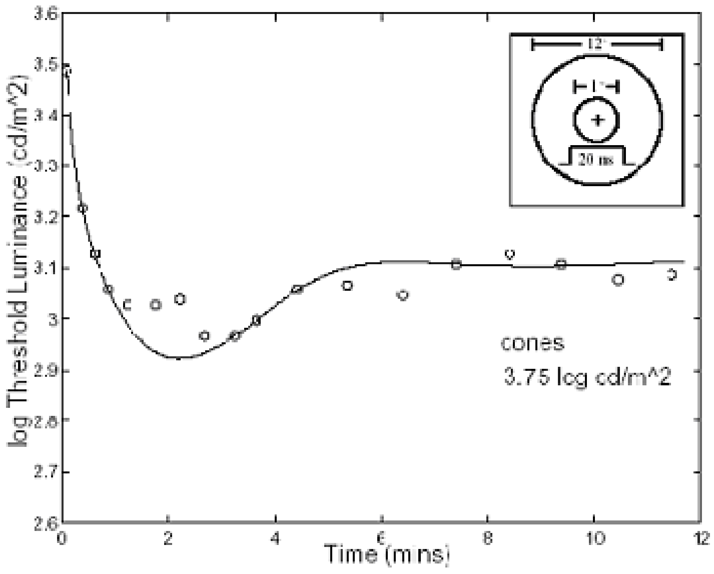


Figure 6.3a,b: The top plot shows the time course of light adaptation for rods and the bottom shows the time course for cones [Ferwerda96]. The plots are time versus threshold luminance.

Unlike the dark adaptation case, we cannot use a single curve to approximate the behavior of adaptation over the range of vision. The reason for this lies in the nature of the light adaptation data. A given jump from a dark adapted state to a photopic state gives a significantly different light adaptation curve depending on the size of the jump. Larger jumps, as shown in Figure 6.3b start at a higher threshold, drop farther in log units from the starting threshold and have a significant non-linear dip in the curve where the threshold drops below it's final adapted state value. Smaller jumps start at lower thresholds, drop progressively fewer log units from the initial threshold and have a more monotonic exponential character. Despite these differences the time over which adaptation takes place is relatively constant for both small and large jumps. A further issue, is that only data for jumps from a totally dark adapted state to a photopic state are available. There is no data for intermediate jumps starting at a partially dark adapted state. We attempt to accommodate these issues in our model.

To address the problem of curve variability we model each adaptation curve as an exponential. This will model the effects of light adaptation without the use of large amounts of stored data. Using an exponential model also has the advantage of simplifying adaptation curve construction for a given jump in adaptation level. The only information needed for each jump in adaptation level is the starting threshold and the ending adapted threshold. Then, since the time constant for light adaptation is approximately invariant with respect to level changes we can specify an exponential curve. Finding the instantaneous threshold consists of looking at the appropriate curve for a given change in adaptation level and extracting the threshold immediately after the change. Two issues arise in attempting to chose the starting threshold. First, we do not have data for the threshold at time zero. Second, Baker [1949] only measured curves for four jumps in adaptation level. With the initial threshold set the final threshold can be found straight-forwardly by using the contrast sensitivity curves in

Figure 6.2 which gives us detection threshold as a function of background illumination level.

To account for the lack of data for the first few seconds after the change in adaptation level in Baker's [1949] data, we can chose a starting threshold by fitting an exponential curve to the measured data and then extrapolate back toward time zero to within a few frame times to estimate the initial threshold. Moving a few frame times away from time zero is a reasonable point to chose a starting threshold because the model begins operating a few frames after a jump in adaptation level. Finally, to overcome the disadvantage of having only a few measured light adaptation curves from Baker [1949], we construct a curve from Baker's data that gives an initial light adaptation threshold for a given jump in adaptation level by fitting to the four values available. Once this curve is set, we can use it to find an estimate of how much above the adapted threshold we should be for a given jump in adaptation level. In the absence of data for intermediate changes in adaptation level (like from one photopic level to another) we simply use this curve, which is based on changes from a fully dark adapted state to a photopic state, to approximate the intermediate changes. The result is a light adaptation model based on JND's and tied closely to published data on light adaptation.

Chapter 7

Results

This chapter presents the results of applying our new real-time tone reproduction operator and is organized into five main sections. First we discuss results which test the performance of our operator over the absolute range of vision. To provide a comparison with previous work that addresses this issue, namely [Ferwerda96], we use the same scene which contains a Snellen acuity chart and Macbeth Colorchecker chart and we simulate the same illumination levels (Figure 7.1). These include daylight (1000 cd/m^2),dim interior lighting (10 cd/m^2), moonlight (0.04 cd/m^2) and starlight(0.001 cd/m^2) illumination levels. A second test, which uses the Ward [1997] bathroom scene at interior and moonlight levels is compared to the original Ward [1997] histogram adjustment operator (Figure 7.2). The results illustrate that although our operator uses graphics hardware and therefore approximates some of the functions used in Ward's operator, it is still predictive and matches the original Ward operator for static scenes. Color coded difference images between our operator and Ward's are provided and discussed (Figure 7.2). The third section presents a number of test cases designed specifically to show the compression of high dynamic range input and to verify that the operator does yield quality results and real-time performance over a range of image resolutions and fields of view. We process the office scene from Ward [1997], the desk and parking garage scenes from Pattanaik [1998] and the Stanford Memorial Church HDR image [Debevec97]. These results are shown in Figure 7.3. The fourth section shows image sequences that simulate the course of light and dark adaptation to demonstrate the time dependent properties of our operator. For dark adaptation we use the Ward [1997] bathroom scene and jump from a daylight adaptation level down to a starlight level (Figure 7.4). To illustrate light adaptation we again use the bathroom scene and jump from a moonlight adaptation level to an interior level (Figure 7.5). In these two cases the adaptation level changes via a global multiplier. The scene itself is not dynamic. A further case illustrates light adaptation for dynamic scene illumination. It shows a sequence in which car headlights pointed at the viewer are turned on (sequence courtesy of Markus Strobel at IMS Chips [IMS03]). In the fifth section we demonstrate that by modifying the psychophysical data on which our operator is based, we can simulate the differences in appearance of high dynamic range scenes to normal and low vision observers. Figure 7.6 shows frames from the bathroom scene for a dark adaptation sequence for a normal viewer of age 25 years and an older viewer of age 75 years. In Figure 7.7 an HDR sequence of a driver entering and exiting a tunnel is also shown. This is the same tunnel used in Pattanaik's [2000] work on time-dependent tone reproduction. The dark adaptation time course, contrast sensitivity and glare models are altered to account for related changes in vision.

7.1 Simulating Wide Absolute Changes in Illumination

The images in Figure 7.1 illustrate the performance of our operator over the absolute range of human vision. The Snellen and Macbeth image has a resolution of 1000 by 700 pixels and a field of view of 40 degrees horizontally and 28 degrees vertically. Once loaded into our operator this image can be processed at a rate of 68.8 fps (frames per second) and we can switch from one adaptation level to another instantly without change in update performance. Panel (a) of Figure 7.1 shows the output of our operator with the adaptation level set to daylight levels of illumination

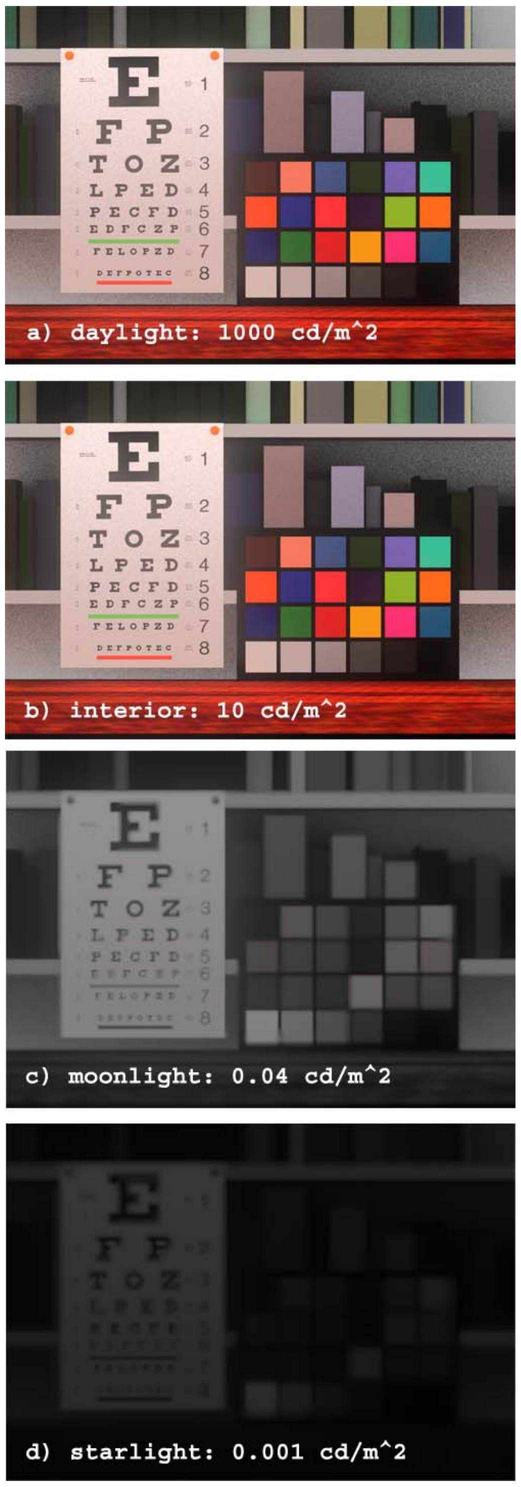


Figure 7.1: Visual function simulated over a large absolute range of vision.

(1000 cd/m^2). In terms of our operator this means that the average luminance in the scene is set to this value and the original scene is scaled appropriately to match it. The image simulates the appearance of photopic light levels and, as expected, the colors of the Macbeth chart are fully saturated and all letters in the Snellen chart can be recognized. There is no reduction in scene contrasts due to the limits of human contrast sensitivity. Another important feature to notice is the reduction in contrast around the Snellen chart and in the letters of the Snellen chart due to the effects of glare. Panel (b) shows the scene at dim interior conditions (10 cd/m^2). In this case we are approaching the top of the mesopic range. The scene is overall a bit darker, some contrast sensitivity has been lost, and colors are less saturated. In comparison with Ferwerda's [1996] results our operator predicts a scene that is not as dark overall. This could be due to how the adaptation level is set or due to the added benefit of the histogram adjustment algorithm that takes local luminance populations into account. Panel (c) shows the scene at a moonlight like adaptation level (0.04 cd/m^2). Overall the scene has become darker and visual acuity has been reduced. An important quality to notice is that the acuity in darker regions of the scene is worse than in brighter regions. Thus, an observer can still discern most if not all the letters of the Snellen acuity chart because it's luminance is much higher than the average for the scene. This highlights the local acuity blurring that our operator computes in contrast to the global blurring function based on a single adaptation level used by Ferwerda's [1996] operator. Another important quality of the scene is that the saturation of colors in the Macbeth chart have been greatly reduced. Shorter wavelength colors like blue and green have become completely achromatic while longer wavelength colors are still slightly saturated. Panel (d) shows the scene at a starlight adaptation level (0.001 cd/m^2). At this level of illumination vision is achromatic, only the largest contrasts are visible, and acuity is very poor except in the brightest areas of the scene. Panels (a)-

(d) highlight both the ability of our operator to map scenes quickly and to simulate visual appearance over the range of vision.

7.2 Comparison With Ward's Histogram Adjustment Operator

Figure 7.2 shows a side by side comparison of our real-time operator and Ward's histogram adjustment operator for the Ward bathroom scene at a dim interior adaptation level (10 cd/m^2) (Figure 7.2a) and a moonlight adaptation level (0.04 cd/m^2) (Figure 7.2b). The dynamic range of this scene spans six log units, the field of view is 50 degrees horizontally by 74 degrees vertically and the image resolution is 346 by 512. Our operator can process this image at 15 fps. A typical running time for Ward's software based histogram adjustment algorithm would be about 1 second. The frame rate for the bathroom scene is much less than in the case of the Snellen and Macbeth image even though the image resolution here is much smaller. An explanation for this performance difference is given in Section 3 of this chapter. Color coded difference images of our results and Ward's illustrate the magnitude and location of differences [on an 8-bit 0-255 scale (Figure 7.2)] which are introduced by the approximations in our hardware-based algorithm. The most significant errors are at the limits of maximum and minimum luminance. This is due to the limited precision of computations in the fragment shaders and glare kernel approximations. At far right in Figure 7.2 are the actual difference images, which show that the errors are hardly visible and not noticeable in our results. We found that our operator performs with similar error characteristics for other images tested.

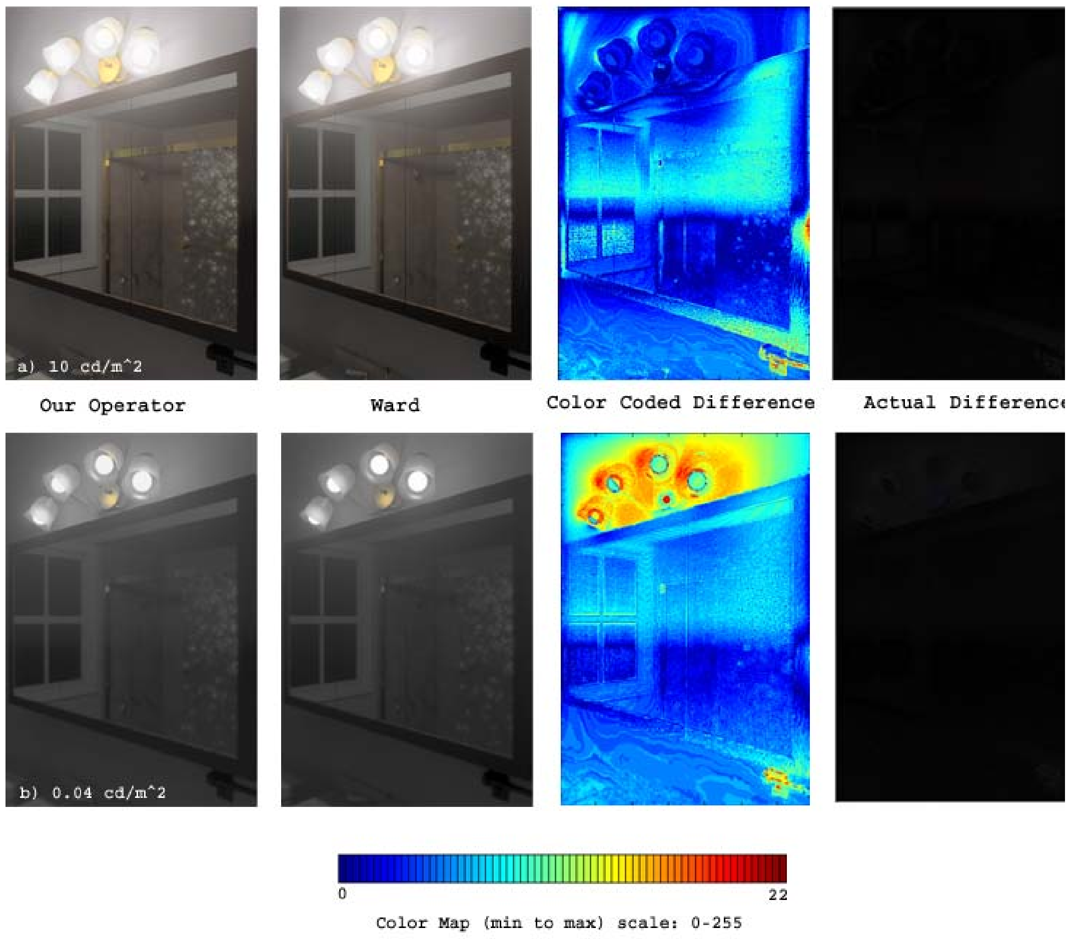


Figure 7.2: Ward Bathroom scene at (a) dim interior and (b) moonlight conditions compared with Ward's [1997] tone reproduction operator which is the basis for our real-time operator. Color coded difference images between our results and Ward's illustrate the magnitude and location of error. Most of the significant error is at the limits of high and low luminance, due to the limited precision of computations in the fragment shaders and glare kernel approximations. At far right are the actual difference images, which are hardly visible and not noticeable in the results.

7.3 High Dynamic Range and Operator Performance

Figure 7.3 illustrates the application of our operator to scenes with varying image resolution, field of view and dynamic range. Frame (a) and (c) are the desk and parking garage scenes from Pattanaik [1998]. Both images are 1536 by 1024, have a field of view of 37 by 25 degrees and an original dynamic range of roughly 10,000:1. They can be computed at 42.5 fps. The desk scene highlights the importance of glare to the proper appearance of brightness in a scene. Frame (b) is the office scene from Ward [1997], which is 1000 by 676, has a field of view of 63 by 45 degrees and originally spans a dynamic range of 1000:1. Our operator can process this image at 22 fps. Frame (d) is the Stanford Memorial Church, which is 512 by 768, has a field of view of 100 by 150 degrees and has an unmapped dynamic range of 250,000:1. It runs at less than one fps.

7.3.1 Performance

As previously mentioned, when we compared the software based Ward histogram adjustment operator to our real-time hardware based operator in Section 2, images with larger resolution are not slower to process than smaller images. The limiting factor is the size of the field of view. The reason for this is that large fields of view, as with the Stanford Memorial Church and Ward Bathroom scene lead to larger foveal images. This slows the histogram adjustment step, which must create a histogram from more samples. More significantly however, it slows the glare computation which is done in software and involves an expensive convolution operation. This fact explains why the Memorial Church runs very slowly. In

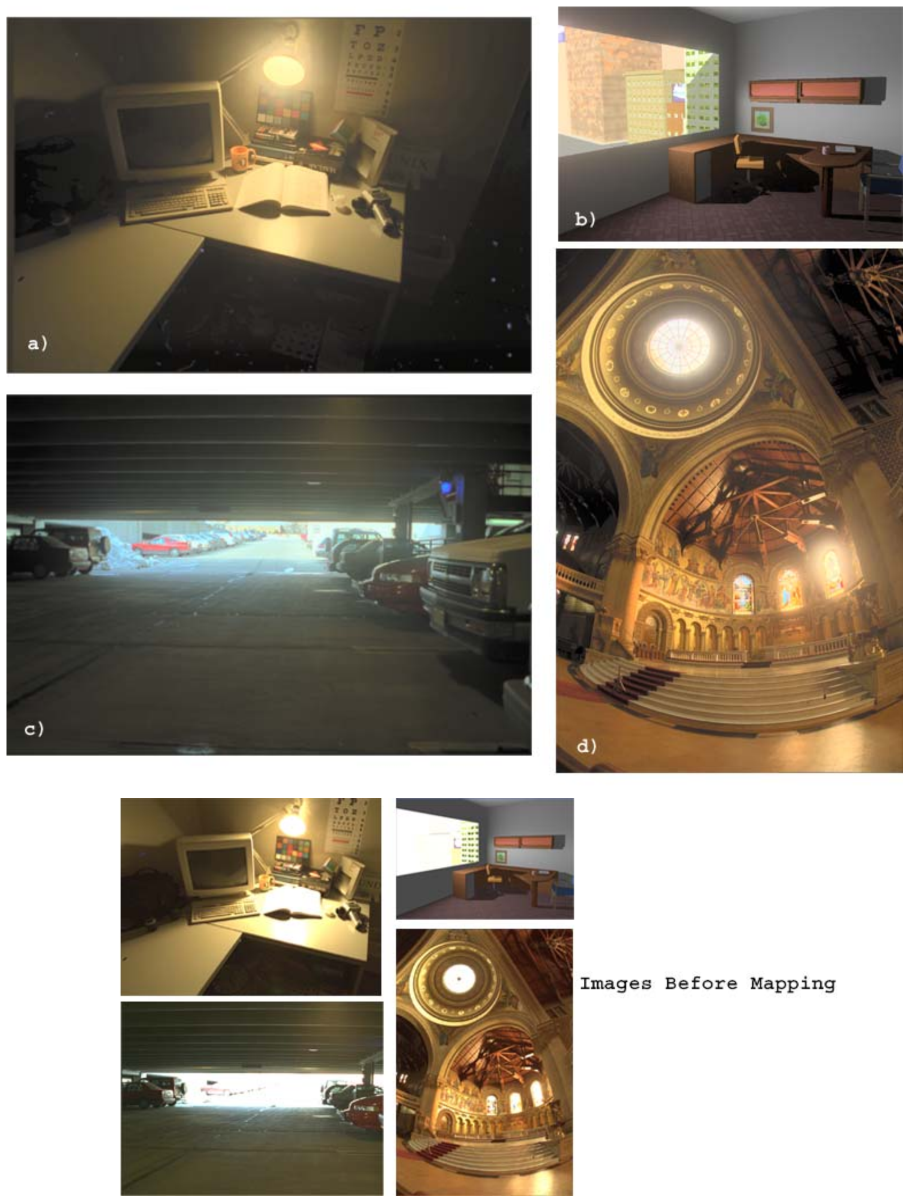


Figure 7.3: The application of our operator to scenes with varying resolution, field of view and dynamic range. (a) and (c) are the desk and parking garage scenes from Pattanaik [1998]. Both images are 1536 by 1024, have a field of view of 37 by 25 degrees and an original dynamic range of roughly 10,000:1. (b) is the office scene from Ward [1997], which is 1000 by 676, has a field of view of 63 by 45 degrees and originally spans a range of 1000:1. (d) is the Stanford Memorial Church (courtesy of Paul Debevec), which is 512 by 768, has a field of view of 100 by 150 degrees and has an unmapped dynamic range of 250,000:1.

subsequent tests we approximated the Church field of view with a 1/3 smaller field of view and the results are nearly indistinguishable and can be computed on the order of 6-7 fps. The convolution speed is the main limiting factor for the performance of our algorithm. For applications where liberty can be taken with the accuracy of glare and visibility computations very high frame rates (60-90fps) can be acheived by downsizing the foveal image and using a small glare kernel.

7.4 Light and Dark Adaptation

Figure 7.4 shows an image sequence captured from our real time operator that shows a simulation of the time course of dark adaptation. In the first frame the average luminance has been set to 1000 cd/m^2, which approximates daylight levels of illumination. In the second frame we have jumped instantaneously to a starlight adaptation level of 0.001 cd/m^2. Initially, only the largest luminance contrasts are visible because the threshold predicted by the dark adaptation curve (Figure 6.1) is very high compared to the final adapted state. As time progresses, the threshold for visible contrasts gets smaller along the dark adaptation curve and progressively smaller contrasts become visible. Because our time course model treats each adaptation level in the image separately we can simulate the effects of local dark adaptation. This is clear from the differing rates of progress for adaptation throughout the image. After 20 minutes very little change in adaptation can be observed and we have reached our steady state conditions for an adaptation level of 0.001 cd/m^2, well into the scotopic range of vision and close to the limits of vision for lower contrasts in the scene.

Figure 7.5 shows an image sequence captured from our operator that simulates the time course of light adaptation. In the first frame the average luminance has been

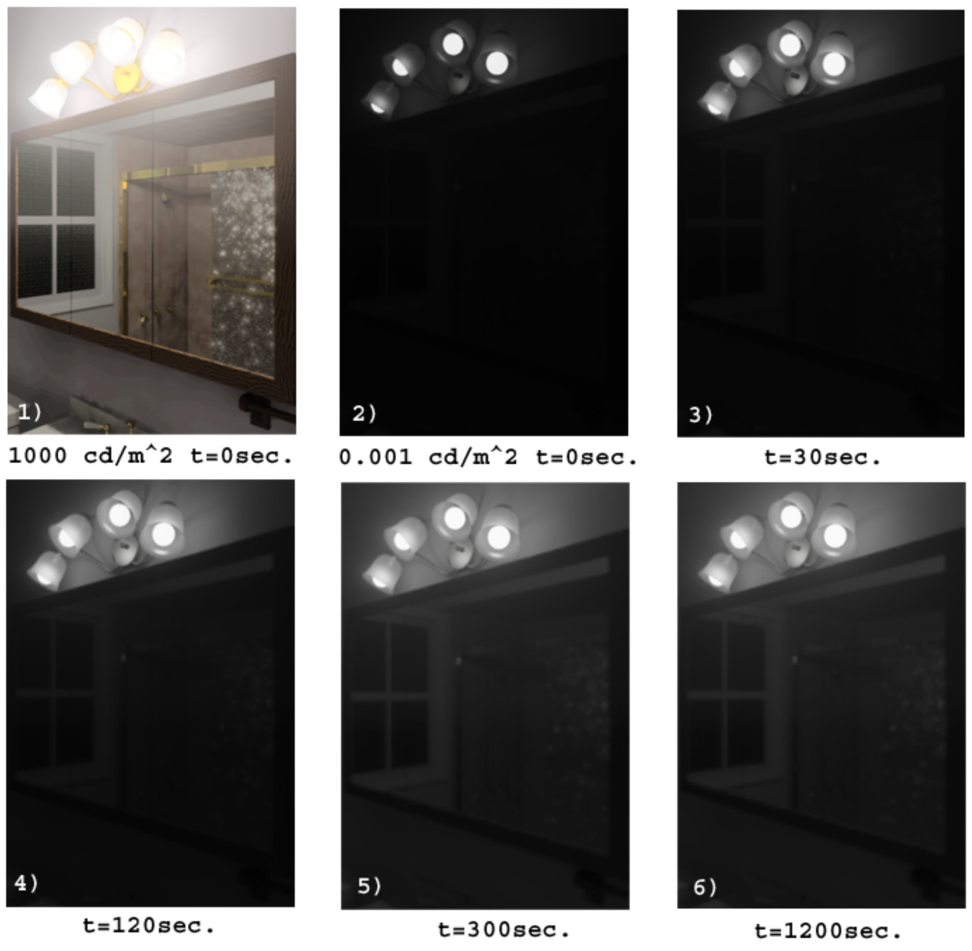


Figure 7.4: Image sequence showing the time course of dark adaptation starting at a daylight adaptation level (average luminance) of 1000 cd/m^{2}and jumping down to a starlight adaptation level of 0.001 cd/m^2. We follow the progress of adaptation out to 20 minutes, well into the rod dominated scotopic range of adaptation. The data used is from Riggs [1971].

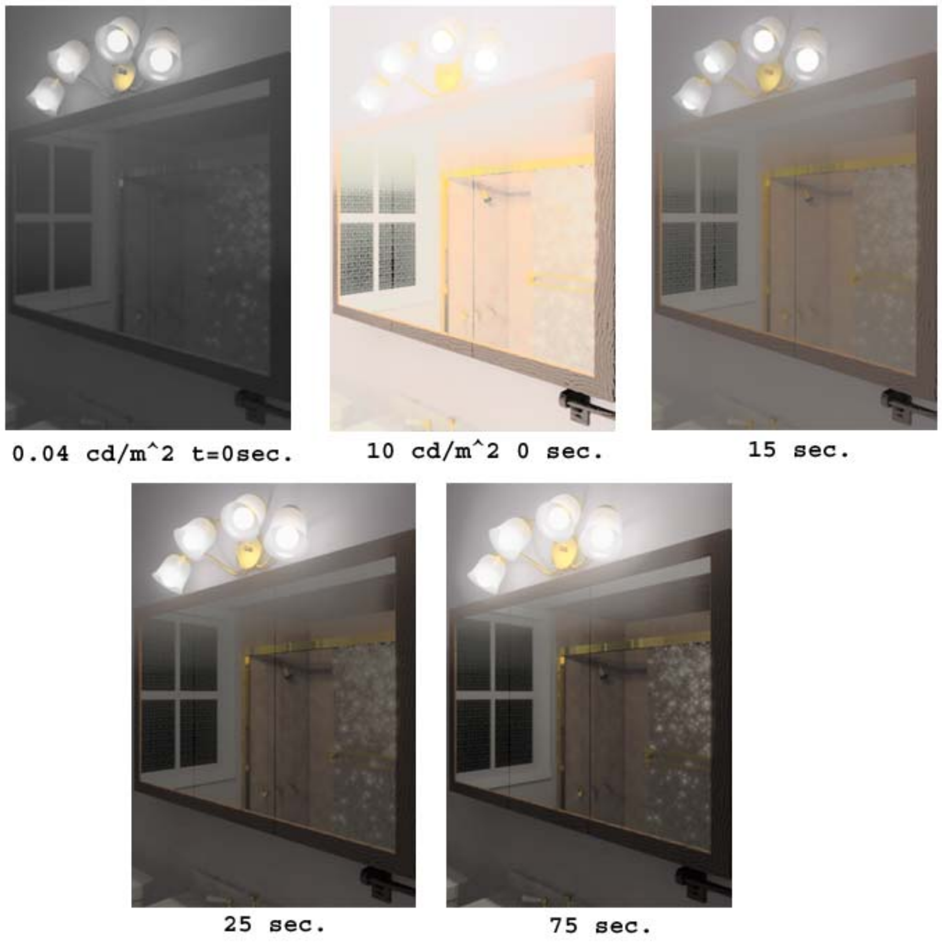


Figure 7.5: Image sequence showing the time course of light adaptation starting at a moonlight adaptation level of 0.04 cd/m^2 and jumping to an interior light level of 10 cd/m^2.

set to 0.04 cd/m^2 which approximates moonlight conditions. In the second frame we have switched to 10 cd/m^2 which approximates dim interior conditions. At first the scene is washed out and excessively bright due to glare and maladaptation. Subsequent frames show the return of sensitivity until the final frame at 75 seconds at which point normal sensitivity has returned. Figure 7.6 illustrates light adaptation in a HDR video sequence. Initially, the headlight is turned off and the scene is at bright interior conditions. When the light is turned on these images show the effects of significant glare and temporary reduction in contrast as light adaptation occurs. Sensitivity quickly returns over tens of seconds. Our operator ran at 46.6 fps for this sequence. The results of using JND's to drive light adaptation lead to less of an extreme effect than predicted by Pattanaik [2000]. Pattanaik's work predicts total wash out of the scene during light adaptation compromising visibility over most of the image. Our operator predicts much less loss of visibility. The visual effect is closer to the work of Ferwerda [1996] but with variation for each separately considered luminance population in the brightness histogram.

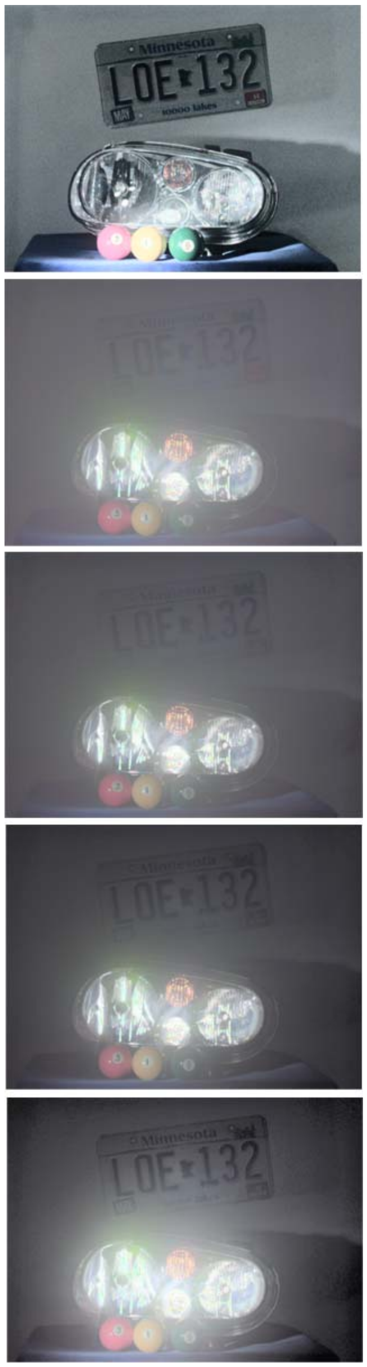


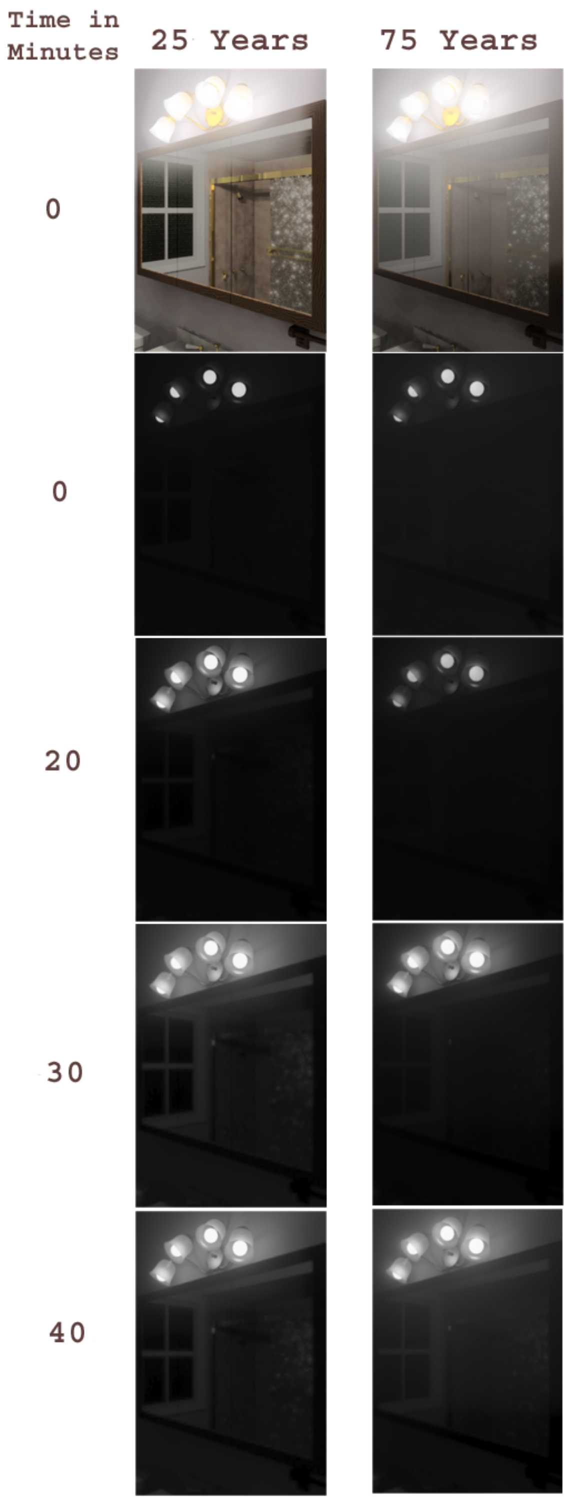
Figure 7.6: Captured frames from a HDR sequence in which a headlight is turned on facing the viewer. The scene starts at normal interior conditions in frame one. Then the headlight is turned on and light adaptation ensues. Notice the initial lose of sensitivity when the headlight comes on and then subsequent return of sensitivity as light adaptation occurs.

7.5 Normal Versus Aged Observers

Figure 7.7 and 7.8 illustrates the application of our operator to low vision simulation, specifically, the images simulate some of the deficits to vision that occur with aging. To create these simulations we have substituted psychophysical data on visual performance from an elderly population for normal data from young observers. We must change the contrast sensitivity curve, dark adaptation time course and contribution from glare. Nothing else in our operator needs to be modified. Compared to a person with normal vision an older person (we chose 75 years) will have a detection thresholds shifted up by a factor of 3.51 [McGwin99]. The glare contribution factor *K* in Figure 5.1 will be increased by a factor of 2.8. Finally, the dark adaptation curve will be different. For an older person the threshold in the initial stages of adaptation is higher, the time until adaptation finishes is longer and the final threshold is higher. We used data from McGwin [1999] for a 25 year old and 75 year old subject.

Figure 7.7 shows a jump from a daylight adaptation level of 1000 cd/m^2 to a starlight adaptation level of 0.001 cd/m^2 in the Ward bathroom scene for a 25 and 75 year old individual . Notice that the older individual has slightly less sensitivity to start with and recovers much more slowly than the younger individual. Also, the older individual reaches a final steady state with less sensitivity. In addition to these limitations glare is also significantly worse for the older person. Figure 7.8 shows frames selected from an HDR video sequence of a 25 year old and 75 year old entering and exiting a tunnel. The tunnel sequence is from Pattanaik's [2000] paper on time dependent tone reproduction. The scene has a dynamic range of approximately 1000:1. The top row simulates the appearance for a 25 year old and the bottom represents appearance for a 75 year old. The older person suffers from more severe

Figure 7.7: Comparison of dark adaptation in a 25 and 75 year old subject using the dark adaptation data of [McGwin99]. The jump is from a daylight adaptation level (average luminance of 1000 cd/m^2) to a starlight adaptation level of 0.001 cd/m^2. Notice that the older individual has slightly less sensitivity to start with and recovers much more slowly than the younger individual. Also, the older individual reaches a steady state with less sensitivity. In addition to these limitations glare is also significantly worse for the older person.


Time in Minutes
25 Years
75 Years
0
0
20
30
40

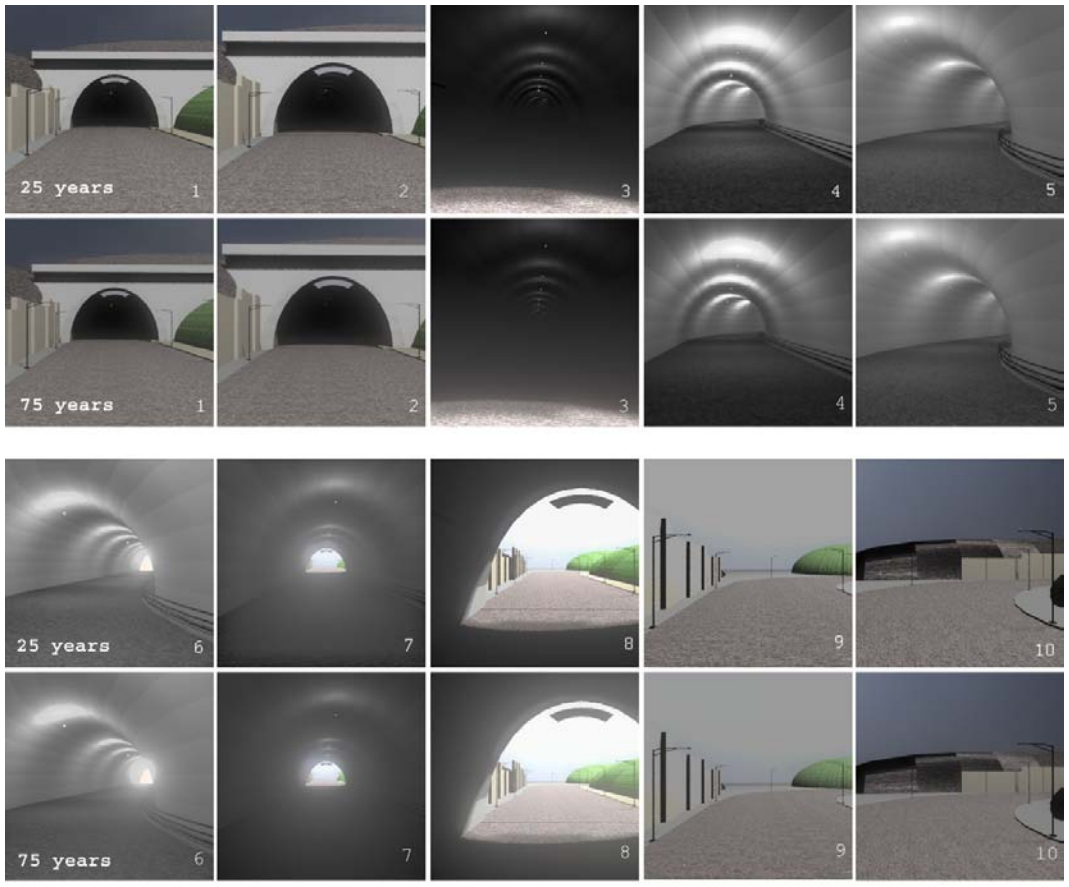


Figure 7.8: Frames selected from an HDR video sequence of a 25 year old and 75 year old entering and exiting a tunnel. The tunnel sequence is from Pattanaik's [2000] time dependent adaptation paper and has a dynamic range of 1000:1. The top row is for a 25 year old and the bottom is for a 75 year old. The older person suffers from more severe visibility problems entering and exiting the tunnel and recovers sensitivity more slowly than the younger person once inside the tunnel. This is highlighted in frames 3,4 and 7,8.

veiling luminance throughout the sequence, which compromises visibility significantly when entering and exiting the tunnel. In addition, dark adaptation is slowed and there is a higher threshold at the start of dark adaptation for the 75 year old. Thus when entering the tunnel, visibility is reduced and sensitivity recovers more slowly than for the 25 year old. However, once adaptation has completed, as shown in frame five, visibility is about the same because the luminance level inside the tunnel is not low enough to bring out the differences in the contrast sensitivity curve of the 75 and 25 year old observers.

The tunnel sequence shown in Figure 7.7 is the most general case for our operator. It is not just a shift up or down in absolute luminance (as in Figure 7.4,7.5 and 7.7) or a static scene with dynamic pixel intensities (as in Figure 7.6), it is a fully dynamic scene. As such it brings out some of the limitations of our approach. Because we base our operator on Ward's histogram adjustment operator, and since it is intended to preserve visibility and not brightness, surfaces in HDR videos may not always remain the same brightness from frame to frame. This occurs in the tunnel sequence just before entering the tunnel and is a property of the histogram adjustment algorithm. Only with a general sequence does this become noticeable. Moreover, our light adaptation approach based on JND's and derived from psychophysical data is sensitive to outliers in the data. If from one frame to the next a high valued patch emerges, the appearance will change dramatically because the entire mapping curve for the image is affected. The approach is very well suited to absolute changes in adaptation level in which the dynamic range stays constant, but falters with noisy data and oscillating changes in dynamic range. Too much sensitivity to luminance changes is the price paid for using a single global operator rather than a local operator in light adaptation.

Chapter 8

Conclusion and Future Work

In this thesis we have implemented a real-time hardware accelerated tone reproduction operator that increases current performance by one or two orders of magnitude, includes more phenomena than past work and is widely applicable. We applied our operator to a wide range of test images and sequences and observed quality results and good performance. Our operator depends on commodity graphics hardware and uses some approximations for its speed, yet it remains predictive. The use of hardware based fragment shaders for image processing proved valuable for performance and highlighted the growing usefulness of graphics hardware for applications beyond standard hardware accelerated computer graphics.

We introduced a new approach to light and dark adaptation that leads to an altered version of Ward's histogram adjustment algorithm that is time varying and uses psychophysical detection threshold data directly. Using the core operator and our time course model we successfully applied our operator to the simulation of vision for aged individuals and processed high dynamic range images and videos comparing a young and old viewer.

Our operator focused on visibility as the most important measure of correctness. Due to this, perception of brightness, colorfulness and apparent contrast and other supra-threshold phenomena were not considered. Sometimes, with dynamic scenes, the apparent brightness of surfaces in the scene would change, leading to a time varying surface brightness. These problems are heightened if input data is noisy and has outliers. In addition our time course models will not always reproduce the subjective appearance of light or dark adaptation correctly. For instance, during light adaptation, contrast may be reduced to preserve visibility at the cost of dimming the scene when it should appear overly bright. Future work should definitely consider threshold and supra-threshold measures.

Finally, it is clear that there is more potential for tone mapping to gain wide usability and good performance from the capabilities of graphics hardware. The more general the fragment programs become the more algorithms can benefit from using them. Local algorithms have been shown to give better results than global ones. A real-time local, predictive tone reproduction operator that takes into account threshold and supra-threshold measures does not yet exist and remains as a major challenge for future researchers in computer graphics.

APPENDIX A

Appendix A1

Below is code illustrating the formation of a floating point texture in OpenGL. To set the texture using an array of floats the input type is set to GL_FLOAT (see the OpenGL programming manual for more information) and the format is set to GL_RGB16 or GL_RGBA16 , which provides 16-bits of precision per channel.

```
void glTexImage2D(GLenum target,
                  GLint level,
                  GLint components,
                  GLsizei width,
                  GLsizei height,
                  GLint border,
                  GLenum format,
                  GLenum type,
                  const GLvoid *pixels)
```

Example:

```
      glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB16, tex_x_res,
      tex_y_res, 0, GL_RGB, GL_FLOAT, hdrImageData);
```

Appendix A2

Below is OpenGL code for creating a P-Buffer (Pixel Buffer). To create a P-Buffer in Windows, the current device context (a reference to the graphics hardware being used) must be retrieved along with the current OpenGL rendering context. Next, parameters for the P-Buffer such as the number of channels, bit depth of each channel, and modes of operation are selected. This is done by specifying the desired parameters and then querying the current device context to see if what you want is available. If the P-Buffer format desired exists a new P-Buffer will be created. Then a device context is retrieved for the new P-Buffer and a new rendering context is created based on the device context. Once this has been done all OpenGL calls will apply to either the standard or the P-Buffer rendering context depending on which is selected at the time. Switching between rendering contexts is done using the wglMakeCurrent function. Textures can be shared between contexts using the wglShareLists function.

```
HDC hpbufdc;

HGLRC hpbufglrc;

HGLRC hwinglrc;

HDC hdc;

HPBUFFERARB hbuf;

// Get current device context

hdc = wglGetCurrentDC();

// Get current OpenGL rendering context

hwinglrc = wglGetCurrentContext();

// Check pixel formats
```

```
float fAttributes[] = {0,0};

int pixelFormat = 1000;

UINT numFormats;

int iAttributes[] = {WGL_SUPPORT_OPENGL_ARB, TRUE,

                     WGL_DRAW_TO_PBUFFER_ARB, TRUE,

                     WGL_BIND_TO_TEXTURE_RGB_ARB, TRUE,

                     WGL_COLOR_BITS_EXT,48,

                     WGL_RED_BITS_EXT, 16,

                     WGL_GREEN_BITS_EXT,16,

                     WGL_BLUE_BITS_EXT,16,

                     WGL_ACCELERATION_ARB,

                     WGL_FULL_ACCELERATION_ARB,0,0};


bool status = wglChoosePixelFormatARB(hdc, iAttributes, fAttributes,

1, &pixelFormat, &numFormats);

cout << "Found a matching pixel format?(0 or 1): " << status << endl;

cout << "And numFormats is not zero: " << numFormats << endl;

cout << "Pixel Format ID: " << pixelFormat << endl;



// Create the pbuffer

int itAttrib[]={WGL_TEXTURE_FORMAT_ARB, WGL_TEXTURE_RGB_ARB,

                WGL_TEXTURE_TARGET_ARB,WGL_TEXTURE_2D_ARB,0,0};

hbuf = wglCreatePbufferARB( hdc, pixelFormat, 512, 512, itAttrib );

// get PBuffer device context

hpbufdc = wglGetPbufferDCARB( hbuf );
```

```
// Get new rendering context
hpbufglrc = wglCreateContext( hpbufdc );
wglShareLists(hwinglrc, hpbufglrc );
int w,h;
wglQueryPbufferARB( hbuf, WGL_PBUFFER_WIDTH_ARB, &w );
wglQueryPbufferARB( hbuf, WGL_PBUFFER_WIDTH_ARB, &h );
cout << "The PBuffer Size I got is: " << w << " " << h << endl;
if( !wglMakeCurrent( hpbufdc, hpbufglrc ) ) {
      cout << "wglMakeCurrent:Failed" << endl;
}
```

Appendix A3

Below is OpenGL code for creating a texture with automatic hardware mipmap generation enabled. Further details can be found under 'Automatic Mipmap Generation' in the OpenGL 1.4 Specification. It has been promoted from the GL_SGIS_generate_mipmap extension which is used in the code below so that all one needs to do now is set the texture parameter GENERATE_MIPMAP to TRUE. First a texture object is created and bound as the current texture. Next standard magnification and minification filters, and texture wrap parameters are set for the texture. Then, assuming that we are using the GL_SGIS_generate_mipmap extension, we first request `GL_NICEST` as a hint (suggestion) to the card to make the highest quality mipmaps possible. Then we set the base level, an integer identifier that refers to the bottom level of the mipmap and the maximum level which gives the final level to be calculated during mipmap formation. Finally, the texture parameter `GL_GENERATE_MIPMAP_SGIS` is set to `TRUE` and the texture is formed.

```
glGenTextures(1, &TexName);
glBindTexture(GL_TEXTURE_2D, TexName);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
GL_LINEAR_MIPMAP_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,
GL_LINEAR_MIPMAP_LINEAR);
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP);
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP);
// Automatic Hardware Mipmap getneration
glHint(GL_GENERATE_MIPMAP_HINT_SGIS, GL_NICEST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_BASE_LEVEL_SGIS, 0);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAX_LEVEL_SGIS, 6);
glTexParameteri(GL_TEXTURE_2D, GL_GENERATE_MIPMAP_SGIS, TRUE);
// Load HDR texture data
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB16, TEX_MAX_RES, TEX_MAX_RES, 0,
                     GL_RGB, GL_FLOAT, hdrImage);
```

Appendix A4

Below is a sample fragment shader program which showcases the two pass capability of current fragment shaders. The second pass begins with more texture sampling operations. This mechanism allows for a powerful *dependent texture read* in which the result of an earlier operation can be used to sample a texture. This allows lookup of functions for any number of purposes. Dependent texture reads have been used for per pixel lighting effects and BRDF approximations [HWLighting]. It's important to note that our work is outdated with respect to the use of fragment shaders. It is more desirable to use a higher level API like Cg which will compile higher level code into the kind of assembly seen in our example. When this work was begun no such APIs were available.

```
// Get a fragment program ID
glGenProgramsARB(1, &shaderNameARB0);
// Make the new fragment program id be the current fragment program
glBindProgramARB(GL_FRAGMENT_PROGRAM_ARB, shaderNameARB0);
// Set the program local variable 0 to rgba = (1.0,1.0,1.0,1)
glProgramLocalParameter4fARB(GL_FRAGMENT_PROGRAM_ARB, 0, 1.0,
                             1.0, 1.0, 1.0 );
// String to hold fragment program code
int proglen = 12;
char** p = new char*[proglen];
// Signifies the start of the program
p[0] = "!!ARBfp1.0";
// Pass one //////////////
// Name some of the incoming fragment attributes and constants
p[1] =  "ATTRIB tex = fragment.texcoord[0];";
p[2] =  "ATTRIB fcol = fragment.color;";
p[3] =  "PARAM ones = program.local[1];";
// Output from program goes here
p[4] = "OUTPUT outColor = result.color;";
// Temporary variables for use in program
```

```
p[5] = "TEMP colin, bias;";

// use the texture coordinates to set the LOD bias

// of the incoming texture

p[6] = "MOV bias, tex;";

// sample texture from texture unit 0 and bias

// the mipmap level of detail

p[7] = "TXB colin, bias, texture[0], 2D;";

// Pass two ///////////////

// Separation between passes is signified by dependent texture

// or regular texture read

p[8] = "TEMP lookup;";

// Use the input color to lookup into the texture

// unit 1 texture.

p[9] = "TEX lookup, colin, texture[1], 2D;";

// Compute two to the power of the x component of the vector lookup

p[10] = "EX2 outColor, lookup.x;";

// Signifies the end of the fragment program

p[11] = "END";

char* string = new char[200];

for( int j=0; j <200; j++ ) string[j] = NULL;

for( int i=0; i < proglen; i++ ) {

      strcat(string, p[i]);

}

int len = strlen(string);

glProgramStringARB(GL_FRAGMENT_PROGRAM_ARB,

GL_PROGRAM_FORMAT_ASCII_ARB, len, string);
```

After setting up the fragment program it can be used by making the following calls in the OpenGL display loop:

```
glEnable(GL_FRAGMENT_PROGRAM_ARB);

glBindProgramARB(GL_FRAGMENT_PROGRAM_ARB, shaderNameARB0);

……rendering code……

glEnable(GL_FRAGMENT_PROGRAM_ARB);
```

Bibliography

[Adams80] A. Adams. The camera. The Ansel Adams Photography Series. Little, Brown and Company, 1980.

[Adams81] A. Adams. The negative. The Ansel Adams Photography Series. Little, Brown and Company, 1981.

[Adams83] A. Adams. The print. The Ansel Adams Photography Series. Little, Brown and Company, 1983.

[Ashikmin02] M. Ashikhmin. A tone mapping algorithm for high contrast images. In 13th Eurographics Workshop on Rendering. Eurographics, June 2002.

[ATI] http://www.ati.com, 2003.

[Bolz03] J. Bolz, I. Farmer, E, Grinspun, P. Schröder. Sparse Matrix Solvers on the GPU: Conjugate Gradients and Multigrid. The Proceedings of SIGGRAPH, 2003.

[Blommaert90] F.J.J. Blommaert, and J.-B. Mertens. An object-oriented model for brightness perception. Spatial Vision 5, 1, pages 15-41, 1990..

[Bruce96] V. Bruce. The role of the face in communication: Implication for videophone design. Interacting with computers, 8, pages 166-176. 1996.

[Chiu93] K. Chiu, M. Herf, P. Shirley, S. Swamy, C. Wang, and K. Zimmerman. Spatially nonuniform scaling functions for high contrast images. In Graphics Interface

’93, pages 245–253, Toronto, Ontario, Canada, May 1993. Canadian Information Processing Society.

[Cohen02] J. Cohen, C. Tchou, T. Hawkins, and P. Debevec. Real-Time high dynamic range texture mapping. In 12th Eurographics Workshop on Rendering, pages 313–320. Eurographics, June 2002.

[Debevec97] P. E. Debevec, and J. Malik. Recovering high dynamic range radiance maps from photographs. In SIGGRAPH 97 Conference Proceedings, Addison Wesley, T. Whitted, Ed., Annual Conference Series, ACM SIGGRAPH, pages369-378, 1997.

[Devlin02] K. Devlin. A review of tone reproduction techniques. Technical Report CSTR-02-005, Department of Computer Science, University of Bristol, November 2002.

[Dicarlo00] J. DiCarlo and B. Wandell. Rendering high dynamic range images. Proceedings of the SPIE: Image Sensors 3965, pages 392-401, 2000.

[Durand02] F. Durand and J. Dorsey. Fast bilateral filtering for the display of high dynamic range image. In John Hughes, editor, SIGGRAPH 2002 Conference Graphics Proceedings, Annual Conference Series, pages 257–265. ACM Press/ACM SIGGRAPH, 2002.

[Durand00] F. Durand and J. Dorsey. Interactive tone mapping. In Rendering Techniques 2000: 11th Eurographics Workshop on Rendering, pages 219–230. Eurographics, June 2000.

[Fairchild95] Fairchild and Reniff . Time course of chromatic adaptation for color-appearance judgments. JOSA A, 12(5):824, 1995.

[Fattal02] R. Fattal, D. Lischinski, and M. Werman. Gradient domain high dynamic range compression. In Proceedings of ACM SIGGRAPH 2002, Computer Graphics Proceedings, Annual Conference Series. ACM Press / ACM SIGGRAPH, July 2002.

[Ferwerda98] J.A. Ferwerda. Visual Models for Realistic Image Synthesis. PhD thesis, Program of Computer Graphics, Cornell University, Ithaca, 1998.

[Ferwerda96] J.A. Ferwerda, S. Pattanaik, P. S. Shirley, and D. P. Greenberg. A model of visual adaptation for realistic image synthesis. In Proceedings of SIGGRAPH 96, Computer Graphics Proceedings, Annual Conference Series, pages 249–258, New Orleans, Louisiana, August 1996. ACM SIGGRAPH / AddisonWesley.

[Geigel97] J. Geigel and K. Musgrave. A model for simulating the photographic development process on digital images. Proceedings of SIGGRAPH ’97, pages 135-143, 1997.

[Glassner95] A. Glassner. Principles of Digital Image Synthesis. Morgan Kaufman, San Francisco, 1995.

[HLSL] http://msdn.microsoft.com/library/default.asp?url=/library/enus/dnhlsl/html/shaderx2_introductionto.asp, 2003.

[Holm00] J. Holm, K. Parlulski, and E. Edwards. Extended colour encoding requirements for photographic applications. CIE Expert Symposium, Extended Range Color Spaces, November 11, 2000.

[Hunt95] R.W.G. Hunt. The Reproduction of Colour, Chapter, Fountain Press, England, 1995.

[Hunt52] R.W.G. Hunt. Light and dark adaptation and the perception of color. JOSA A, 42(3):190, 1952.

[Hurvich81] L.M. Hurvich. Color Vision. Sinaur Associates Inc. Sunderland, Massachusetts, 1981.

[Hurvich56] L. M. Hurvich and D. Jameson. Some quantitative Aspects of an Opponent-Colors Theory. IV. A Psychological Color Specification System. Journal of the Optical Society of America, Vol. 46, No. 6, June 1956, pp. 416-421.

[HWLighting] http://developer.nvidia.com/object/docs_lighting_materials.html, 2003

[IMS03] http://www.ims-chips.de, 2003

[Jameson56] D. Jameson and L.M. Hurvich. Some Quantitative Aspects of an Opponent-Colors Theory. I. Chromatic Responses and Chromatic Saturation. Journal of the Optical Society of America, Vol. 45, No. 7, July 1955, pp. 546-552.

[Jensen00] H.W Jensen, S. Premoze, P. Shirley, M.M. Stark, W.B. Thompson and J.A. Ferwerda. Night Rendering. Tech. Rep. UUCS-00-016, Computer Science Department, University of Utah, August 2000

[Jobson97] D.J. Jobson, Z. Rahman, and G.A. Woodell. A multiscale retinex for bridging the gap between color images and the human observation of scenes. IEEE Transactions on Image Processing, 6(7):965–976, July 1997.

[Kruger03] J. Krüger, R. Westermann. Linear Algebra Operators for GPU Implementation of Numerical Algorithms. Proceedings of SIGGRAPH, 2003.

[Land77] E. H. Land. The Retinex Theory of Color Vision. (The) Scientific American, 12:108-128, December 1977.

[London98] D. London and J. Upton. Photography. Sixth ed. Longman, 1998.

[Mark03] W.R. Mark, R.S. Glanville, K. Akeley, M.J. Kilgard. Cg: A System for Programming Graphics Hardware in a C-like Language. The Proceedings of SIGGRAPH, 2003.

[Marr82] D. Marr. Vision. Wiley, New York, NY, 1982.

[McGwin99] G. McGwin, Jr., G.R. Jackson and C. Owsley. Using nonlinear regression to estimate parameters of dark adaptation. Behavior Research Methods, Instruments, & Computers, 31(4), pages 712-717, 1999.

[Miller84] N.J. Miller, P.Y. Ngai, and D.D. Miller. The application of computer graphics in lighting design. Journal of the IES, 14:6–26, 1984.

[Millerson91] Millerson. Lighting for Television and Films, 3^{rd} ed. Focal Press, 1991.

[Moon45] P. Moon and D. Spencer. The Visual Effect of Non-Uniform Surrounds. Journal of the Optical Society of America, vol. 35, No 3, pp. 233-248, 1945.

[Natural] http://www.ati.com/developer/demos/r9700.html, 2003.

[Neumann98] L. Neumann, K. Matkovic and W. Purgathofer. Automatic exposure in computer graphics based on the minimum information loss principle. Proceedings of Computer Graphics International ’98, pages 1-19, 1998.

[Nikon00] Nikon. http://www.nikon.ca, 2000.

[NVidia] http://www.nvidia.com, 2003.

[OpenGL] http://www.opengl.org, 2003.

[OpenGLSpec] http://www.opengl.org/developers/documentation/specs.html, 2003.

[Oppenheim68] A. Oppenheim, R. Schafer, and T. Stockham. Nonlinear filtering of multiplied and convolved signals. In Proceedings of the IEEE, volume 56, pages 1264–1291, August 1968.

[Pattanaik02] S.N. Pattanaik and H. Yee. Adaptive gain control for high dynamic range image display. Proceedings of the Spring Conference in Computer Graphics (SCCG2002).

[Pattanaik00] S.N. Pattanaik, J.E. Tumblin, H. Yee, and D. P. Greenberg. Time-dependent visual adaptation for realistic image display. In Proceedings of ACM SIGGRAPH 2000, Computer Graphics Proceedings, Annual Conference Series, pages 47–54. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, July 2000.

[Pattanaik98] S.N. Pattanaik, J.A. Ferwerda, M.D. Fairchild, and D.P. Greenberg. A multiscale model of adaptation and spatial vision for realistic image display. In Proceedings of SIGGRAPH 98, Computer Graphics Proceedings, Annual Conference Series, pages 287–298, Orlando, Florida, July 1998. ACM SIGGRAPH / AddisonWesley.

[Reinhard02] E. Reinhard, M. Stark, P. Shirley, and J. Ferwerda. Photographic tone reproduction for digital images. In Proceedings of ACM SIGGRAPH 2002, Computer Graphics Proceedings, Annual Conference Series. ACM Press / ACM SIGGRAPH, July 2002.

[RGBE] http://radsite.lbl.gov/radiance/refer/Notes/picture_format.html, 2003.

[Scheel00] A. Scheel, M. Stamminger, and H-P. Seidel. Tone reproduction for interactive walkthroughs. Computer Graphics Forum, 19(3):301–312, August 2000.

[Schlick94] C. Schlick. Quantization Technique for Visualization of High Dynamic Range Pictures. Proceedings 5th Eurographics Workshop on Rendering, pp.7-20., 1994.

[Shaler37] S. Shaler. The relation between visual acuity and illumination. Journal of General Physiology, 21, pages 165-188, 1937.

[Spencer95] G. Spencer, P.S. Shirley, K. Zimmerman, and D.P. Greenberg. Physically-based glare effects for digital images. In Proceedings of SIGGRAPH 95, Computer Graphics Proceedings, Annual Conference Series, pages 325–334, Los Angeles, California, August 1995. ACM SIGGRAPH / AddisonWesley.

[Stevens60] S.S. Stevens and J.C. Stevens. Brightness Function: Parametric Effects of Adaptation and Contrast, Program of the 1960 Annual Meeting, Journal of the Optical Society of America, Volume 53, #ll, page 1139, November, 1960.

[Stockam72] T.G. Stockham. Image processing in the context of a visual model. Proceedings of the IEEE, 60, pages 828-842, 1972.

[Thompson02] W. Thompson, P. Shirley, and J.A. Ferwerda. A spatial post-processing algorithm for images of night scenes. Journal of Graphics Tools 7(1), pages 1-12, 2002.

[Tumblin99] Tumblin, J., Hodgins, J. K., and Guenter, B. K. Two methods for display of high contrast images. ACM Transactions on Graphics 18, 1 (January 1999), 56--94. ISSN 0730-0301.

[Tumblin99] J. Tumblin and G. Turk. Lcis: A boundary hierarchy for detail-preserving contrast reduction. In Proceedings of SIGGRAPH 99, Computer Graphics Proceedings, Annual Conference Series, pages 83–90, Los Angeles, California, August 1999. ACM SIGGRAPH / Addison Wesley Longman.

[Tumblin93] J. Tumblin and H.E. Rushmeier. Tone reproduction for realistic images. IEEE Computer Graphics & Applications, 13(6):42–48, November 1993.

[Upstill85] S.D. Upstill. The Realistic Presentation of Synthetic Images. PhD thesis, Computer Science Division, University of California, Berkeley, 1985.

[Ward94] G. Ward. A contrast-based scale factor for luminance display. In Graphics Gems IV, pages 415–421. Academic Press, Boston, 1994.

[Ward97] G.W. Larson, H.E. Rushmeier, and C. Piatko. A visibility matching tone reproduction operator for high dynamic range scenes. IEEE Transactions on Visualization and Computer Graphics, 3(4):291–306, October – December 1997.

[Williams83] L. Williams, Pyramidal Parametrics. Computer Graphics, v.17,n.3, July 1983.