Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelor Thesis

Yannick Rietz

**Implementation of an HDR Tone Mapping Operator
Based on Human Perception**

Fakultät Design, Medien und Information
Department Medientechnik

Faculty of Design, Media and Information
Department of Media Technology

Yannick Rietz

Implementation of an HDR Tone Mapping Operator Based on Human Perception

Bachelor Thesis for attaining the academic degree Bachelor of Science

in the degree course Bachelor of Science in Media Systems
at the Department of Media Technology
of the Faculty of Design, Media and Information
of the Hamburg University of Applied Sciences

First examiner: Prof. Dr. Edmund Weitz
Second examiner: Marc Briede

Handed in on: August 31, 2015

Yannick Rietz

Thema der Arbeit
Implementierung eines HDR Tone Mapping Operators
basierend auf der menschlichen Wahrnehmung

Stichworte
Tone Mapping Operator, High Dynamic Range Rendering, Visuelle Wahrnehmung, Simulation der menschlichen Wahrnehmung, Game Engine

Kurzzusammenfassung
Das Ziel dieser Bachelor-Thesis ist es, einen bestimmten HDR Tone Mapping Operator für eine Echtzeit 3D Game Engine zu implementieren. Ein besonderes Augenmerk liegt dabei auf der Simulation der Sichtbarkeit von Details. Der Text setzt sich mit den relevanten Eigenschaften des menschlichen Wahrnehmungsapparates auseinander und erklärt sowohl die Funktionsweise der Operators, als auch die Spezifika der neuen Implementierung.

Yannick Rietz

Title of the paper
Implementation of an HDR Tone Mapping Operator
Based on Human Perception

Keywords
Tone mapping operator, high dynamic range rendering, visual perception, visual system simulation, game engine

Abstract
The goal of this bachelor thesis is the implementation of a specific HDR tone mapping operator for a real-time 3D game engine. Particular emphasis is placed on simulation of detail visibility. The text describes the relevant characteristics of the human visual system. It details the mode of operation of the operator and includes a walkthough of the specifics of the new implementation.

Contents

List of Figures

Please note that the images in this document are best viewed on a monitor as they will not display the same detail and luminance in print.

1 Introduction

For more than half a decade, the aim in the production of realistic computer graphics has been to render virtual worlds which are as convincing as possible. This means that they should evoke the same response in the viewer as it would in a real world scene. As computational power becomes more easily available, resolutions and polygon counts go up steadily. Yet to feel truly realistic, a virtual world cannot just consist of an accumulation of realistically modeled objects, but rather the parts must also interact with each other in a realistic manner.

Humans are used to moving through a world that is completely dictated by the laws of physics. For that reason, it is easy for us to spot even the slightest divergence from those laws. As a result, developers must stick to the laws of physics as much as possible when simulating and rendering virtual worlds. As a matter of course, this also applies to simulating lighting and illumination. This is why *high dynamic range rendering* has become an essential tool in rendering realistic virtual worlds, as well as tone mapping, which enables the results to be shown on conventional displays.

This thesis discusses the functional principles of these tools and discusses the implementation of a tone mapping operator by Irawan, Ferwerdam and Marschner.

This first chapter introduces the tone mapping problem, and what the aims of the proposed solution are. Moreover, previous work on this topic is investigated. The second chapter details how the human visual system perceives differences in brightness, and which mechanisms are responsible for adaptation to different lighting conditions. The third chapter explains the mode of operation of the operator proposed by Irawan et al. The fourth chapter introduces a few prerequisites of graphics programming and compute shaders. The fifth chapter goes into detail concerning the implementation of the operator. Last but not least, the sixth chapter considers performance and further refinements.

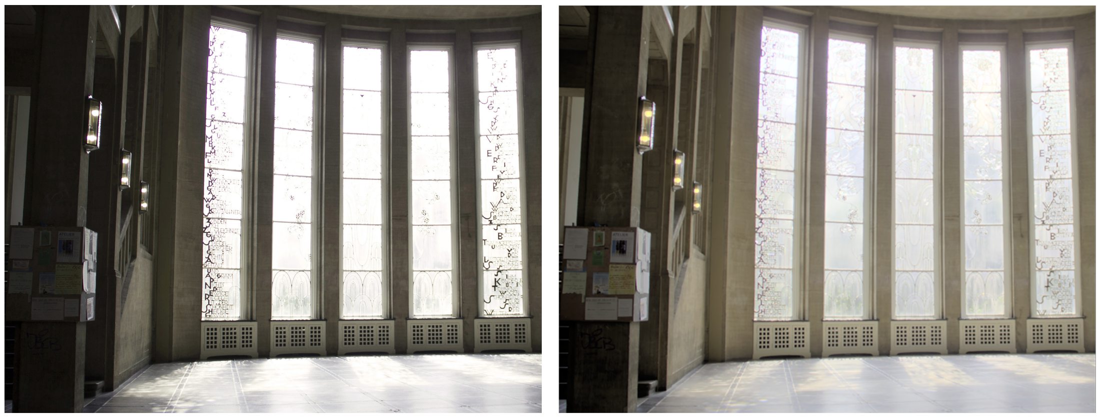


Figure 1.1: Left: a standard photo, as captured by a modern digital camera. Right: an HDR photo generated by combining photos of various exposures, tone mapped using the method described in this thesis. Note that in the image on the right, there are details visible in the bright areas that are clipped to white in the left image.

1.1 Dynamic Range

The *luminance* of a surface is the luminous intensity per unit area of light and is typically measured in candela per square meter (cd/m^2). The term *dynamic range* in this thesis refers to the range from the lowest to the highest luminance value of a given image. The range of luminances distinguishable by the *human visual system* spans from approximately 10^{-6}cd/m^2 to 10^8cd/m^2, which constitutes a total of 14 orders of magnitude [Delvin 2002]. For example, a white piece of paper is more than a million times brighter on a sunny day at noon than on a moonlit night. The visual system cannot perceive the full visual range of luminances at the same time, but it can adapt to different lighting conditions by means of multiple adaptation mechanisms, which will be discussed in detail in Chapter 2.

In reality, the complete distinguishable range of luminances is almost never present at the same time. Still, an ordinary scene can easily feature luminances that span 4 to 6 orders of magnitude [Reinhard *et al.* 2010:237]. Typical display devices have a dynamic range which spans approximately 2 orders of magnitude [Reinhard *et al.* 2010:233]: see Figure 1.2 for a visual comparison. Images that have a greater dynamic range than 2 orders of magnitude are usually referred to as having a *high dynamic range* (HDR). Correspondingly, images and scenes which feature a lower dynamic range are referred to as having a *low dynamic range* (LDR).

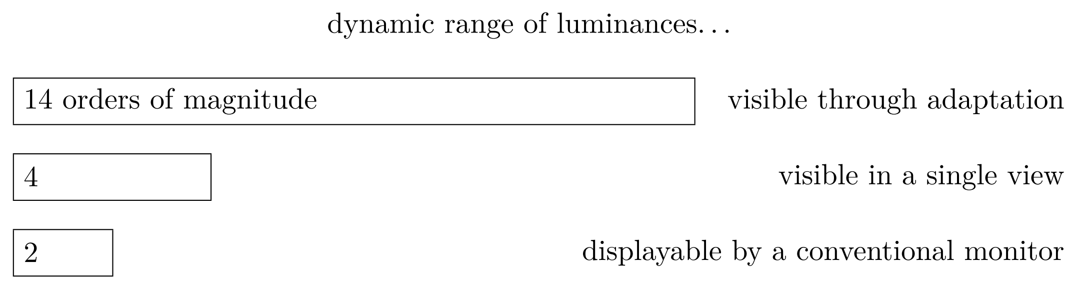

dynamic range of luminances…
14 orders of magnitude
visible through adaptation
4
visible in a single view
2
displayable by a conventional monitor

Figure 1.2: Comparison of different dynamic ranges in orders of magnitude.

While LDR material is suitable for displaying directly on conventional LDR display devices, HDR material can not be displayed directly on such devices, but instead must be converted to match the dynamic range of the display device in question. This is a lossy process called *tone mapping*. As the name suggests, the HDR values get mapped to the LDR during tone mapping. The function responsible for this is called a *tone mapping operator* (TMO). HDR display devices, that are capable of displaying HDR material directly, will become more affordable and mature during the next few years, but standard LDR display devices will continue to play a fundamental role in image display in the foreseeable future.

1.2 Taxonomy of Tone Mapping Operators

Whenever a HDR scene must be displayed on a conventional display device, the need for some sort of tone mapping remains. It is an integral part of digital camera systems, and some of the methods used in computer graphics have been influenced by those used in camera systems.

There are many different kinds of TMOs, which are often classified according to the following properties. The first distinction is between *global* (also known as *spatially uniform* or *single-scale*) and *local* (also known as *spatially varying* or *multi-scale*) operators. While global operators apply the same mapping function across an entire image, local operators aim to divide the image into segments of similar luminances and apply separate, adjusted mapping functions to each of these. In this manner, local operators can often reproduce more detail in both bright and dark parts of the image, and they can also distort the sensation of differences in luminance across the image.

A further distinction is made between *static* and *dynamic* TMOs. Static operators are designed to process only single images. Dynamic operators take the course of time into consideration and are therefore suitable for processing moving images. Some dynamic operators are also capable of simulating adaptation to different light conditions over time.

A third distinction can be made in accordance with the intent of the operator. Some TMOs strive to produce pleasing images depending on personal preferences or artistic goals (*best subjective quality*). Other TMOs are optimized to reproduce, as far as possible, the original appearance of the image, including contrast, sharpness and colors of the scene, properties which could get heavily distorted without the use of a proper TMO (*scene reproduction*). These operators are sometimes referred to as *tone reproduction operators*. Some operators take this a step further and also simulate limitations limitations of the visual system (*visual system simulators*). These limitations include limited contrast perception, glare or loss of color perception in dark conditions [Eilertsen *et al.* 2013].

The TMO used for the purposes of this thesis is a global, dynamic visual system simulator.

1.3 Goals of the New Operator

The TMO that will be discussed in this thesis was developed by Irawan, Ferwerda and Marschner in 2005, and is detailed in their publication *Perceptually Based Tone Mapping of High Dynamic Range Image Streams* [Irawan *et al.* 2005].

The main aim of this operator is to accurately represent scene visibility, or, in other words, to reproduce as much of the visible contrast and detail as possible. It includes a detailed simulation of the adaptation state of the visual system and how it changes over time. The authors consulted several psychophysical studies in order to model the limitations of the visual system.

There are many applications which could benefit from this tone mapping method. Lighting conditions in the simulation of architecture can be evaluated more accurately than with a more simple TMO. In driving simulations the visibility of obstacles can be displayed more realistically. The simulation of *low vision* can be useful in medical simulations.

The goal of this thesis is to implement Irawan's operator for a game engine. Game engines traditionally achieve the highest graphical fidelity possible in real-time on consumer hardware, which makes them a good fit for any 3D simulation which requires a certain amount of visual realism. They have also become increasingly affordable and often include support for virtual reality hardware, which is also of interest concerning 3D simulations.

1.4 Previous Work on This Topic

HDR rendering is already a well-established feature of 3D game engines and developers are experimenting with more sophisticated tone mapping operators, often with impressive results. These new operators often incorporate various effects of visual perception, or try to mimic the behavior of film to achieve a more filmic appearance. However, very fast performance speeds and a pleasing image appearance are also prioritized by all of these operators.

Tone mapping operators that try to achieve a critical degree of perceptual realism are less commonly found, and are less often implemented by popular game engines. They are instead found in less flexible or less open 3D frameworks. No freely available implementation of a global, dynamic visual system simulator was found for any of the popular game engines; thus, such an implementation will be attempted in this thesis.

2 The Human Visual System

The visual system consists of the *eye*, notably including the *retina*, the *optic nerve* and parts of the *brain*. The process of visual perception is a complex interplay between these parts, that includes the interpretation of the stimulus, as well as the comparison to known patterns and objects [Clifford *et al.* 2007]. Researchers have yet to fully understand the inner workings of visual perception, and new findings are made regularly. This chapter details the mechanisms of the visual system that are responsible for detecting differences in light intensity, as they are assumed and modeled by Irawan et al. in their paper.

2.1 Contrast Perception

Although LDR images exhibit a narrower dynamic range and lower peak luminances, they can still evoke a similar response in the observer as real HDR scenes. The reason for this is that the visual system is more sensitive to relative rather than to absolute luminances [Delvin 2002]. To understand how the impression of contrast forms, one has to look at the way the visual system manages to cover a dynamic range that spans over 10 orders of magnitude.

2.2 Mechanisms of Visual Adaptation

The visual system adapts in part globally to the *background intensity* of a scene, which can be approximated as the mean luminance of the whole field of vision, and in part locally for the different areas of the retina.

A special area of the retina is the *fovea*. This is the part of the retina where visual acuity is the highest, and the center of which is used to focus on objects [Hood and Finkelstein 1986:5-5]. It is especially important to model the adaptation of this area when predicting scene visibility.

The eyes are never completely at rest. Instead, they do semi-random movements at any time and never fixate on a single point for more than 500 ms. This is called *saccadic* *eye movements*. Typically, it takes between 200 ms and 400 ms for the eye to fixate on the next point [Hoffman and Baskaran 1995]. Therefore, the adaptation state is also constantly changing and is usually less than optimal [Irawan *et al.* 2005]. The state of the visual system not being optimally adapted is called *maladaptation*. The visual system possesses multiple mechanisms that let it adapt to different lighting conditions. These are detailed in the following subsections.

2.2.1 The Pupil

The first is probably also the most well-known. By opening or closing the circular structure in the eye that is the iris, the size of the pupil changes, and thus, the amount of light that enters the eye can be regulated. The size of the pupil varies between about 2 mm in bright conditions to 8 mm in the dark. This enables a reduction in light intensity entering the eye by a factor of 16, which is not much considering the range of 10 billion to 1 that the visual system is covering [Reinhard *et al.* 2010:239]. This is why the effects of the pupil are not taken into account by Irawans TMO.

2.2.2 The Rod and Cone System

The human retina has two types of photoreceptors that are responsible for vision: *Rods* and *cones* are dividing the high dynamic range that the visual system is covering into two smaller ranges. Figure 2.1 illustrates this relationship. Rods are more sensible to light and responsible for vision in dark conditions. The range of luminances they are working in is called *scotopic range*. Cones are less sensible to light and work in normal to bright conditions. This range is called *photopic range*. The photopic and scotopic ranges overlap in a region that is referred to as the *mesopic range*. In the mesopic range, both types of photoreceptors are active and working. Rods and Cones are individually adapting to changing light conditions [Reinhard *et al.* 2010:240f]. Under bright light, when rods are less sensitive than cones, the rods are referred to as being *saturated*.

Note that both curves of Figure 2.1 are flat at low luminance levels and become log-linear over the range where the visual system adapts well. This range is referred to as the *Weber Range*. In this range *Weber's Law* applies, which states that the ratio of a stimulus and the respective *just noticeable difference* (JND) is a constant.

There are three different kinds of cone cells, each for different wavelengths of light. This way, cone cells enable color vision in the photopic range, while in the scotopic range, with only rods active, we are unable to differentiate between colors.

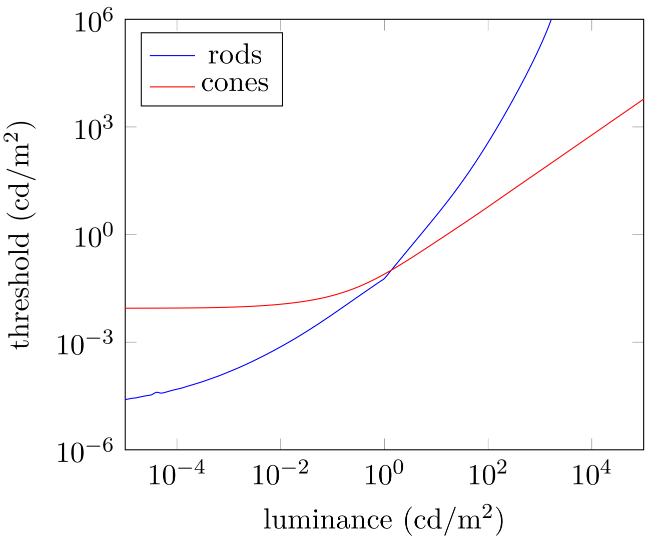


Figure 2.1: The just noticeable difference threshold for an always optimally adapted visual system, as computed by the new implementation of the TMO. It shows that at about 1 cd/m^2, the cone system gets more sensitive than the rod system. It takes over the task of contrast perception. At about 100 cd/m^2 the rod system is saturated.

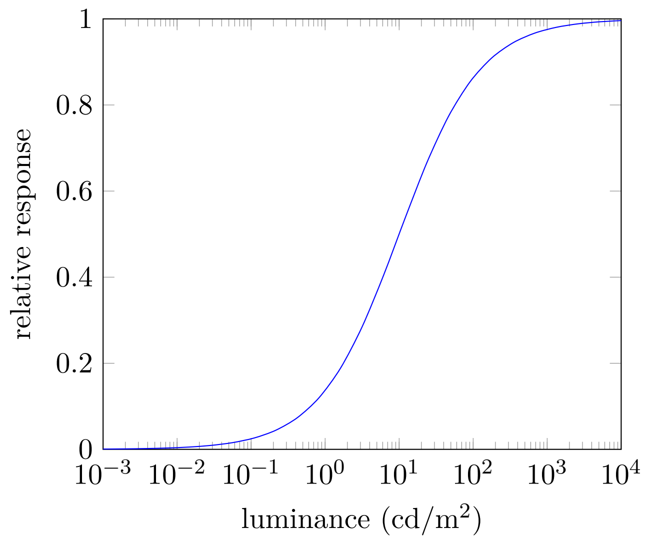


Figure 2.2: The Response of dark-adapted rod cells to short exposures to various luminance values. [Reinhard *et al.* 2010:244]

2.2.3 Photopigment Depletion and Regeneration

Photoreceptor cells contain photopigments, that get broken down when they absorb photons. These pigments enable the photoreceptors to fire a neural signal when exposed to light. The pigments appear more bright after this process, which is why it is called *bleaching* [Kaiser 2009]. Regeneration of photopigment is a relatively slow process that takes longer for rod cells than for cone cells. A lower concentration of photopigment can render photoreceptor cells less sensitive to light. But the role of photopigment depletion in visual adaptation is not considered very big, as researchers have found that even after long exposures to high luminances, the concentration of photopigment in cones is not reduced significantly. Some researchers believe that, in the photopic range, nearly all of the photopigment in rods is depleted, which would render them inoperable[Reinhard *et al.* 2010:243]. However, Irawan et al. worked with the assumption that after adaptation the fraction of unbleached pigment is the same in rods as in cones [Irawan *et al.* 2005]. Regardless of this, the rod system is still saturated above the mesopic range in their calculations.

2.2.4 Neural Adaptation

Photoreceptor cells turn the absorbed light into neural responses. When measured under the influence of short flashes of light of varying luminance, the responses form an S-shaped plot as shown in Figure 2.2. This response curve can be approximated by the following equation:

$$R(L, \sigma(L_a)) = \frac{L^n}{L^n + \sigma(L_a)^n}, \tag{2.1}$$

where R is the photoreceptor *response* in relative units between 0 and 1. L is the luminance the cell is flashed with. $\sigma(L_a)$ is defined as the adaptation state when the system is optimally adapted to luminance L_a. This is the semisaturation constant of this equation and provides the luminance that causes a response of 50%. Finally, n changes the sensitivity of the system, or in other words, the steepness of the curve, and has a value generally between 0.7 and 1.0 [Irawan *et al.* 2005].

The Equation 2.1 was proposed by Naka and Rushton [Naka and Rushton 1966]. As visible in the graph, photoreceptors respond logarithmically to increasing stimuli. In the equation $\sigma(L_a)$ controls the position of the curve on the horizontal intensity axis. This is necessary, because the response curve is only log-linear in a narrow portion of the dynamic range. The value of $\sigma(L_a)$ mainly depends on two processes of neural adaptation: *fast* and *slow neural adaptation*. These are modulations in the way the signal is being processed, and their effect is comparable to the sensitivity or gain control settings of modern cameras.

3 Data Flow of the New Operator

Recall that the goal of the new operator is to predict the visibility of details and realistically simulate the limitations of the visual system. To accomplish this, the authors built on the histogram adjustment method proposed by Ward et al., and extended it with a new adaptation model which builds on the temporal dynamics of visual adaptation as described by Pattanaik et al. [Ward-Larson *et al.* 1997][Pattanaik *et al.* 2000]. This chapter details their approach. Figure 3.1 provides an overview of the necessary steps.

3.1 Histogram Adjustment

This method was first described by Ward et al., and because of its similarity to *histogram equalization* it was termed *histogram adjustment*. The key idea here is to use the cumulative distribution function of the luminance histogram of an HDR image as the *mapping function*. Compared to the typical S-shaped mapping functions which are commonly used by TMOs, this allows for more fine-grained adjustments. For an example see Figure 3.2. Here, the second row of diagrams shows the normalized cumulative distribution functions of the histograms in the first row.

Using this technique, it is possible to use the available dynamic range of the output image more efficiently. Luminances that are more prevalent in the image get assigned a proportionally greater number of display luminances, while contrast in more sparsely populated regions of the image's histogram get compressed more heavily. This way, more detail is preserved in the resulting image, and it is possible to convey the impression of an over all higher contrast.

Since the fovea adapts to areas of approximately $1°$ of viewing angle, the operator reduces the resolution of the input image until each pixel represents the mean luminance of the area of one of those *foveal points*. The calculation of the mapping function is continued with the resulting *foveal image*.

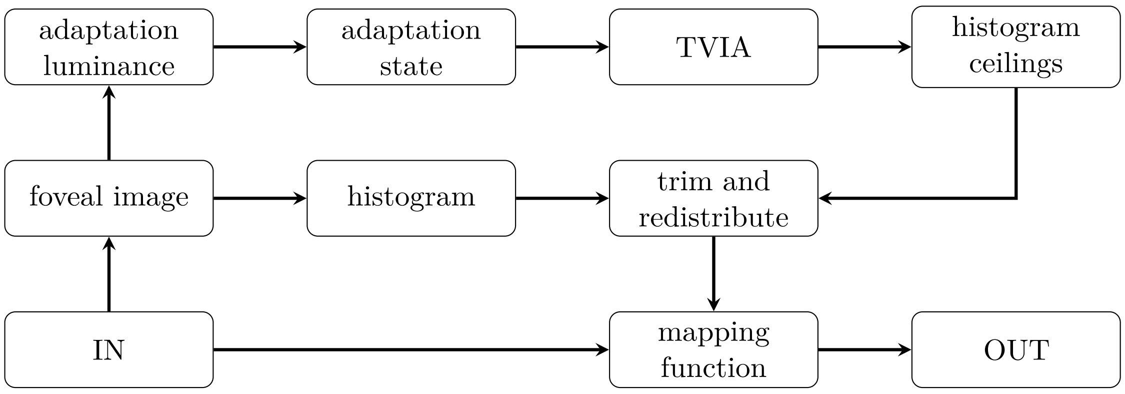


Figure 3.1: Block diagram of the new operator as it was implemented for this thesis.

If $h(b_i)$ counts the number of foveal points that lie in histogram bin b_i, the cumulative histogram $P(b)$, normalized by the total number of foveal points T, is defined as the following [Reinhard *et al.* 2010:378]:

$$P(b) = \sum_{b_i<b} h(b_i)/T \tag{3.1}$$

To use $P(b)$ as a mapping function and attain the $\log_{10}$ of the target display luminances, one can use the following equation, where $L_\mathrm{d}(x,y)$ is the target display luminance at position (x,y), $L_\mathrm{d,max}$ is the maximum display luminance, $L_\mathrm{d,min}$ is the minimum display luminance and $L_\mathrm{w}(x,y)$ is the world luminance at position (x,y).

$$\log(L_\mathrm{d}(x,y)) = \log(L_\mathrm{d,min}) + (\log(L_\mathrm{d,max}) - \log(L_\mathrm{d,min}))P(\log(L_\mathrm{w}(x,y))) \tag{3.2}$$

When just naïvely using this method, there is a high chance that contrast will be exaggerated in some areas of the image, see the left side of Figure 3.2 for a demonstration of this effect. For this reason, Ward et al. limited the slope of the mapping function to the ratio of contrast visibility thresholds for the display and world luminances using a *threshold vs. intensity* (TVI) function as it is pictured in Figure 2.1. This way, two world luminances, which are not visibly different, will map to two display luminances which are also not visibly different [Irawan *et al.* 2005]. The TVI function $\Delta L(L_\mathrm{a})$ takes an adaptation level L_a, where L_a is the luminance value that the visual system is processing and is optimally adapted to.

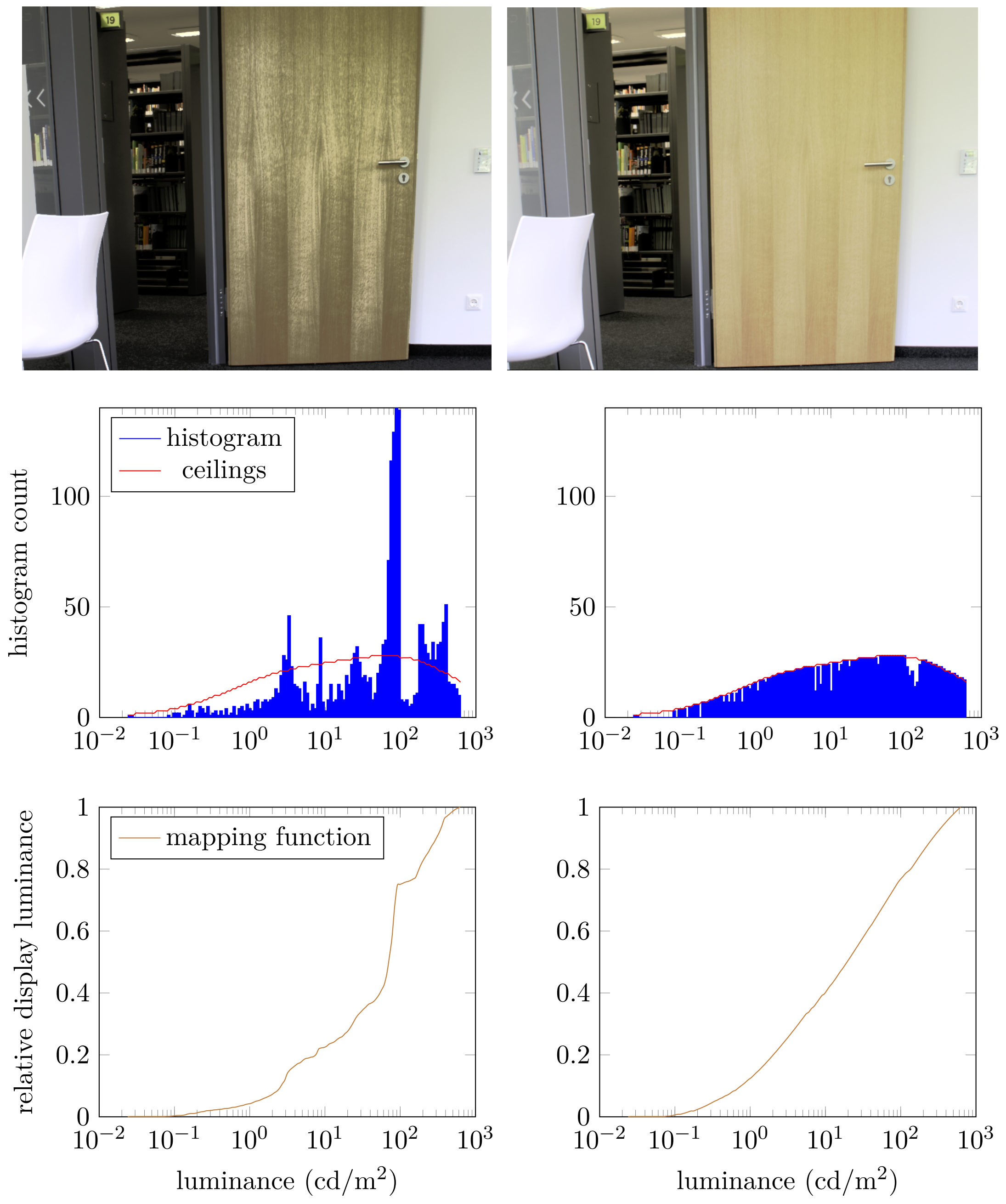


Figure 3.2: Left side: naïve histogram equalization, with TVIA ceilings included for orientation. Right side: histogram adjustment method, trimmed with TVIA ceilings as proposed by Irawan et al. Note that the left image exhibits exaggerated contrasts in the area of the wooden door. In comparison, the right image appears correct.

$$\frac{dL_\mathrm{d}}{dL_\mathrm{w}} \leq \frac{\Delta L(L_\mathrm{d})}{\Delta L(L_\mathrm{w})} \tag{3.3}$$

From this inequality Ward derives a constraint that will be used to trim the histogram values, prior to calculating the cumulative distribution function [Ward-Larson *et al.* 1997].

$$h(b_i) \leq \frac{T}{N} * \frac{\log(L_{\mathrm{w,max}}) - \log(L_{\mathrm{w,min}})}{\log(L_{\mathrm{d,max}}) - \log(L_{\mathrm{d,min}})} * \frac{\Delta L(L_{\mathrm{d}_i})/L_{\mathrm{d}_i}}{\Delta L(L_{\mathrm{w}_i})/L_{\mathrm{w}_i}}, \tag{3.4}$$

where N is the total number of histogram bins used, L_{w_i} is the world luminance for histogram bin b_i and L_{d_i} is the display luminance for histogram bin d_i. The value for N can be chosen freely and does not need to be very high.

Ward simply truncates the histogram values that exceed this *ceiling*. This changes the number of adaptation samples T and thereby changes the ceilings, creating a nonlinear problem. Ward et al. solve this problem by iteratively truncating counts and recomputing the ceilings until an arbitrary tolerance is reached [Ward-Larson *et al.* 1997].

Irawan et al. improve on this solution by redistributing the trimmed histogram counts to other bins proportional to the existing counts, without exceeding the ceiling for those bins. If not all trimmings can be redistributed this way, Irawan et al. distribute the rest uniformly over empty bins, provided that there are any. This way, T does not change during the process of constraining the histogram and the results of the TMO become more consistent. If not all trimmings can be redistributed without exceeding the ceiling, the input material is not HDR. See the next section for how this case is handled in Irawan's TMO.

Ward's operator provides a technique to accurately predict the visibility of details. However, because it uses the classic TVI function to constrain the visibility of contrast, it works under the assumption that the visual system is optimally adapted to any luminance value it processes, which is not realistic. Furthermore, Ward's operator is only optimised for single image processing and can yield inconsistent results when used on continuous image streams.

3.2 Handling of Low Dynamic Range

When, after constraining the visibility of details, the dynamic range of the image does not need to be compressed, we consider it to be LDR. When used on streams of moving

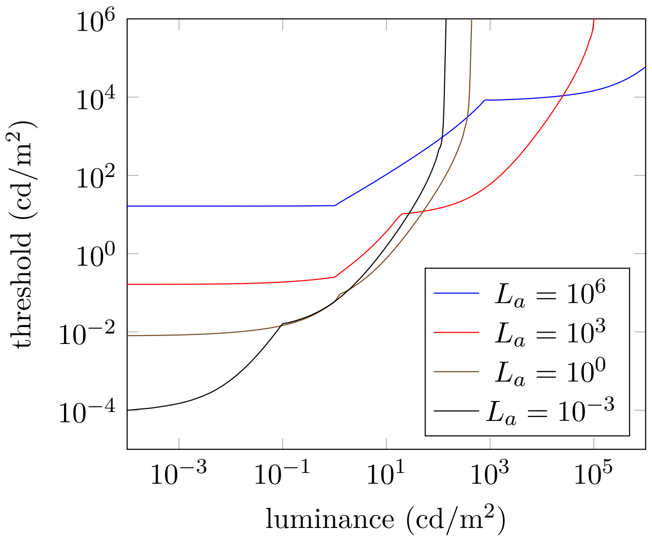


Figure 3.3: Slices of the TVIA for several values of the adaptation luminance L_a, as computed by the new implementation of Irawans operator. The lower envelope of these curves is approximately the TVI pictured in Figure 2.1.

images, the main reason for Ward's TMO to produce inconsistent results is that it does not handle LDR material the same way as HDR material. Instead, it switches to a completely different mode. As mentioned in the last section, to detect this case, Irawan et al. check if the sum of all the ceilings is not less than T.

$$\sum_{i=1}^{N} \frac{T}{N} * \frac{\log(L_{\mathrm{w,max}}) - \log(L_{\mathrm{w,min}})}{\log(L_{\mathrm{d,max}}) - \log(L_{\mathrm{d,min}})} * \frac{\Delta L(L_{\mathrm{d}_i})/L_{\mathrm{d}_i}}{\Delta L(L_{\mathrm{w}_i})/L_{\mathrm{w}_i}} \geq T \tag{3.5}$$

In this case, Irawan et al. reduce $\log(L_{\mathrm{d,max}}) - \log(L_{d,min})$ so that Equation 3.5 is met exactly. This way, after redistribution, the sample count in each bin will be exactly the ceiling for this bin. Afterwards, the operator needs to decide what part of the available display dynamic range the image should occupy, since it does not span the full display dynamic range. Irawan et al. compare the highest and lowest relative responses generated by the world luminances and the display luminances to solve this problem.

3.3 The Threshold Vs. Intensity and Adaptation Function

To account for the range of adaptation states of the visual system, Irawan et al. replace the TVI equation used in Ward's operator by their own *threshold vs. intensity and* *adaptation* (TVIA) function $\Delta L(L, \sigma(L_\mathrm{a}))$. It takes a luminance and an adaptation state, and it gives the JND for any combination of these. To find this function, they consulted the Naka-Rushton Equation (2.1). The idea is that when there is a constant ΔR that is the just noticeable difference in receptor response, then a just noticeable difference in luminance is:

$$R(L, \sigma(L_\mathrm{a})) + \Delta R = \frac{(L + \Delta L(L, \sigma(L_\mathrm{a})))^n}{(L + \Delta L(L, \sigma(L_\mathrm{a})))^n + \sigma(L_\mathrm{a})^n} \tag{3.6}$$

$$\Delta L(L, \sigma(L_\mathrm{a})) = \sigma(L_\mathrm{a}) \left(\frac{R(L, \sigma(L_\mathrm{a})) + \Delta R}{1 - (R(L, \sigma(L_\mathrm{a})) + \Delta R)} \right)^{\frac{1}{n}} - L \tag{3.7}$$

Equation 3.7 is Equation 3.6 rearranged.

As there cannot be a response values above 1, the operator has to check if $R(L, \sigma(L_\mathrm{a})) + \Delta R > 1$. In that case, the visual system reaches saturation and cannot discriminate any higher luminance values, and therefore, the threshold is infinite. If $R(L, \sigma(L_\mathrm{a})) + \Delta R \leq 1$, the operator can calculate the TVIA, see Equation 3.7. A few slices of this function are plotted in Figure 3.3.

3.4 The Adaptation State σ

As already mentioned in Section 2.2.4, the function $\sigma(L_\mathrm{a})$ shifts the response function of the photoreceptors left and right along the horizontal axis. This way, luminances that are close to $\sigma(L_\mathrm{a})$ fall near the center of the response function, where it is the steepest, and the required change in luminance to differ the response by ΔR is small. The bigger the difference between L and $\sigma(L_\mathrm{a})$, the higher the relative threshold.

To define the slope of $\sigma(L_\mathrm{a})$ Irawan et al. take Equation 3.6 and look at the case where $L = L_\mathrm{a}$. Since $\Delta L(L, \sigma(L))$ is exactly the known TVI function, the equation can now be solved numerically for $\sigma(L_\mathrm{a})$. As Irawan et al. put it:

> Graphically, this calculation is equivalent to shifting the response function left and right until the difference between $R(L, \sigma(L))$ and $R(L + \Delta L(L, \sigma(L)), \sigma(L))$ is exactly ΔR.

To account for the temporal dynamics of the different mechanisms of adaptation, the authors split $\sigma(L_\mathrm{a})$ into three factors, one for each of the major mechanisms: $\sigma_\mathrm{b}(L_\mathrm{a})$ for photopigment bleaching, $\sigma_\mathrm{n}(L_\mathrm{a})$ for fast neural adaptation and $\sigma_\mathrm{c}(L_\mathrm{a})$ for slow neural adaptation.

$$\sigma(L_\mathrm{a}) = \sigma_\mathrm{b}(L_\mathrm{a}) * \sigma_\mathrm{c}(L_\mathrm{a}) * \sigma_\mathrm{n}(L_\mathrm{a}) \tag{3.8}$$

The relative portion of unbleached pigment $p(L)$ follows the following formula [Hood and Finkelstein 1986:5-42]:

$$p(L) = \frac{I_0}{I_0 + L} \tag{3.9}$$

The amount of signal transmitted by receptors is assumed to be proportional to $L * p(L)$ [Irawan *et al.* 2005], which leads to

$$\sigma_\mathrm{b}(L_\mathrm{a}) = \frac{1}{p(L_\mathrm{a})} \tag{3.10}$$

Irawan et al. modeled the time course of pigment depletion and regeneration with the following exponential decay function:

$$p = p(L_\mathrm{a}) + (p_0 - p(L_\mathrm{a})) * e^{\frac{-t}{t_0 * p(L_\mathrm{a})}}, \tag{3.11}$$

where t is the time in seconds since the luminance changed from L_0 to L_a and the portion of unbleached pigment was p_0. The value of the denominator $t_0 * p(L_\mathrm{a})$ determines the abruptness of the function. A different time constant t_0 is used for the rod and for the cone system [Irawan *et al.* 2005]. Because in bright surroundings $p(L_\mathrm{a}) < 1$, p increases slower than it decreases, just like pigment regeneration happens slower than pigment depletion.

The adaptation values for fast neural adaptation $\sigma_\mathrm{n}(L_\mathrm{a})$ and slow neural adaptation $\sigma_\mathrm{c}(L_\mathrm{a})$ are modeled by Irawan et al. through the following equations:

$$\log(\sigma_\mathrm{n}(L_\mathrm{a})) = \frac{2.027 L^{0.6406}}{L^{0.6406} + 5.859^{0.6406}} + 0.01711 \tag{3.12}$$

$$\log(\sigma_\mathrm{c}(L_\mathrm{a})) = \frac{1.929 L^{0.8471}}{L^{0.8471} + 1048^{0.8471}} + 0.01820 \tag{3.13}$$

$$\log(\sigma_\mathrm{n}(L_\mathrm{a})) = \frac{2.311 L^{0.3604}}{L^{0.3604} + 5.859^{0.3604}} - 2.749 \tag{3.14}$$

$$\log(\sigma_\mathrm{c}(L_\mathrm{a})) = \frac{1.735 L^{0.9524}}{L^{0.9524} + 1.277^{0.9524}} + 0.005684 \tag{3.15}$$

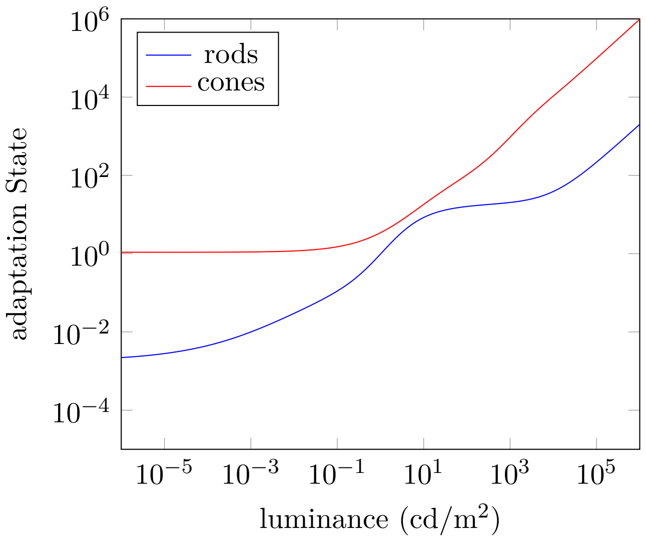


Figure 3.4: The steady-state adaptation state $\sigma(L_\mathrm{a})$ for the rod and cone system, as computed by the new implementation of the operator. The graph reflects the fact that the two adaptation systems are each suited for a different dynamic range of luminances, as the optimal case would be $\sigma(L_\mathrm{a}) = L$ for all L.

Equations 3.12 and 3.13 are for the cone system and Equations 3.14 and 3.15 are for the rod system. To model the time course of the neural adaptation, Irawan et al. use another exponential decay function:

$$L = L_\mathrm{a} + (L_0 - L_\mathrm{a}) * e^{\frac{-t}{t_0}} \tag{3.16}$$

The steady-state adaptation state for rods and cones is plotted in Figure 3.4. These two functions combined resemble the TVI function pictured in Figure 2.1. A few samples of the resulting thresholds during dark adaptation over the course of time are plotted in Figure 3.5.

3.5 Partial Adaptation

As mentioned in Section 2.2, the eyes are constantly doing saccadic movements. Adaptation times as short as a few milliseconds can already cause big differences in contrast recognition. Because the foveal area of the visual system is very small, its position over the image would have to be known at any time to correctly account for those

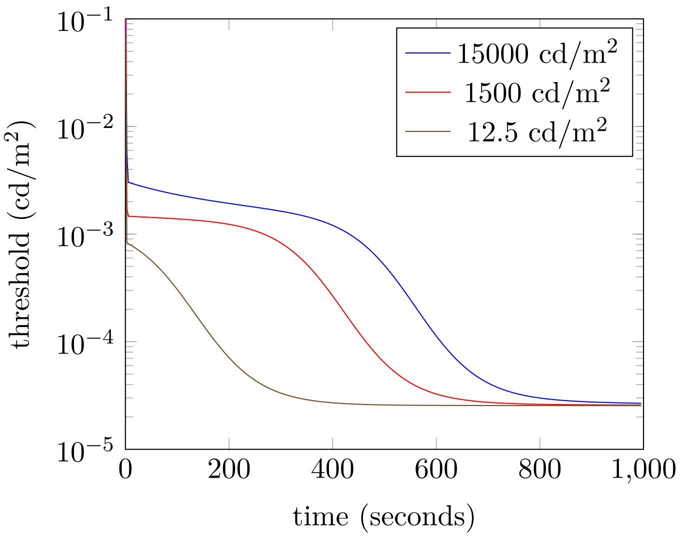


Figure 3.5: The time course of dark adaptation, as computed by the new implementation of the operator, for three different starting adaptation luminances. See the legend for the exact values. The jump due to fast neural adaptation during the first seconds is barely visible in this representation.

micro-adaptations. Since this is not feasible without extensive use of highly sophisticated additional equipment, the more sensible solution for this factor is to simply assume an average *partial adaptation time*.

Between frames, Irawan's operator only keeps track of the adaptation state over the average luminance of the image. But to account for partial adaptation, an individual adaptation state is calculated for each world luminance, as if the observer started in the average adaptation state and then adapted for a fixation time of t_f to luminance L [Irawan *et al.* 2005].

4 Technical Fundamentals

This chapter provides the reader with the fundamentals of graphics programming and compute shaders. Furthermore, the basics of the XYZ color space and the choice of game engine are explained in this chapter.

4.1 Graphics Hardware

Rendering graphics requires the processing of large amounts of data. To accelerate this, specialized graphics hardware has been introduced to the market in the 1990s. It contains an additional processor that was termed *graphics processing unity* (GPU) by the Nvidia Corporation [Varcholik 2014:8]. It complements the CPU in modern computing devices.

CPUs are optimized to quickly execute tasks in sequence and are usually capable of doing only a couple of things simultaneously. However, many graphics calculations can be done in parallel. The GPU is designed to take advantage of this fact. While the CPU is fast at doing things in sequence, GPUs are fast at doing the same task many times in parallel. Programs that are executed on the GPU usually are referred to as *shaders*.

Graphics APIs define specific sequences of steps for rendering 3D graphics on the screen. This is called the *graphics pipeline* and comprises stages for different kinds of shaders [Varcholik 2014:9ff]. Tone mapping is done towards the end of this pipeline, after rasterization, usually by means of one or more *pixel shaders*. This way, as many shaders and effects as possible can benefit from the realistic dynamic range of luminances before the TMO converts the image to LDR.

4.2 Parallel Programming and Compute Shaders

Prior to the 21st century, GPUs could only execute a *fixed* set of functions. Today, most stages of the graphics pipeline are freely programmable. There is also the possibility to use the GPU independently of the graphics pipeline for any kind of calculation by means of so-called *compute shaders*. Such shaders have been used for this thesis.

To take advantage of the capabilities of the GPU, *parallel programming* is required, in contrast to the conventional *sequential programming* for the CPU. A compute shader comprises one or more functions. These are called *kernels*.

GPUs contain one or more processing units. NVIDIA termed these *streaming multiprocessors* (SM). SMs execute many instances of a kernel at a time in parallel. Each instance is called a *thread*. Threads are processed in *thread blocks* or *groups*. The maximum number of threads in a group depends on the hardware, as the groups need to "fit into" the SMs. At the time of writing, this number typically is 1024 for modern desktop computers [Varcholik 2014:545f].

When *launching* a kernel on the GPU, each thread gets assigned a *thread ID* and a *group ID*. These IDs can have up to three dimensions for both threads and groups. The way threads are divided among these dimensions can be freely chosen. Except for the total number of threads and groups, this has no performance implications. The IDs are visible to the threads themselves and can be used to assign different portions of memory for each thread to process. Each thread has its own *local memory* and each SM has a block of memory that is visible to all of its threads. This is called *group shared memory*. In addition, there is *global memory* that is visible to all threads. These three types of memory reside in the GPU. All data that needs to be processed first needs to be transfered from the CPU to the GPU [Wilt 2013].

In general, the GPU code must be written in a way that lets the threads operate as independently as possible. This means that they should not operate on the same memory. This is not always possible. Since threads are not guaranteed to execute at the same speed, multiple threads operating on the same memory would introduce race conditions. As a solution, there are multiple ways for threads to be synchronized. One is an API-specific synchronization function, which stops all threads within a group until each of them has reached this function call. The other one is atomic operations which guarantee that no two threads are accessing the same memory address at the same time [Wilt 2013].

4.3 The XYZ Color Space

Most 3D environments internally represent color information as a combination of the primary colors red, green and blue. This is called *RGB color space* and R, G and B are called the *tristimulus values* of this color space. There are many different RGB color spaces, each defining the primary colors and the available intensities a little differently.

It is not possible to describe all colors observable by human vision in such a color space. For this purpose, the *Commission internationale de l'éclairage* created the *CIE XYZ color space* in 1931 [Greule 2015:76].

Since it contains all colors of the natural world, it is possible to describe any color or color space within the XYZ color space independent of display device or medium. An RGB color space is defined within the gamut of the XYZ space using an $x_W y_W$ chromaticity pair and a maximum luminance Y_W. It is possible to convert from tristimulus color space to another using a 3×3 matrix transformation [Reinhard *et al.* 2010:34].

The tristimulus values of the XYZ color space are imaginary in the sense that they cannot be generated by any light spectrum. The function for Y was designed to be equal to the *luminosity function* which describes the typical perceived brightness of different wavelengths, or *spectral sensitivity.* One can extract the color information, or *chromaticity*, from the XYZ tristimulus values using the following equations [Greule 2015:79]:

$$x = \frac{X}{X+Y+Z} \tag{4.1}$$

$$y = \frac{Y}{X+Y+Z} \tag{4.2}$$

$$z = \frac{Z}{X+Y+Z} = 1 - x - y \tag{4.3}$$

Since $z = 1 - x - y$, only x and y need to be kept. These are the chromaticity coordinates that form the CIE chromaticity diagram shown in Figure 4.1. Adding the luminance information Y to the chromaticity coordinates yields the *xyY color space.* This is widely used in TMOs. They take advantage of the fact that luminance information is kept separate from chromaticity because most tone mapping operations happen only on the luminance information.

4.4 The Graphics Framework

As mentioned before, the goal of this thesis was to implement a visibility predicting TMO within a game engine because game engines traditionally archive the highest fidelity 3D graphics in real time on a wide range of hardware configurations.

To decide which engine to use as the framework for the new real time implementation of Irawans TMO, different aspects were considered. Support for HDR rendering is a

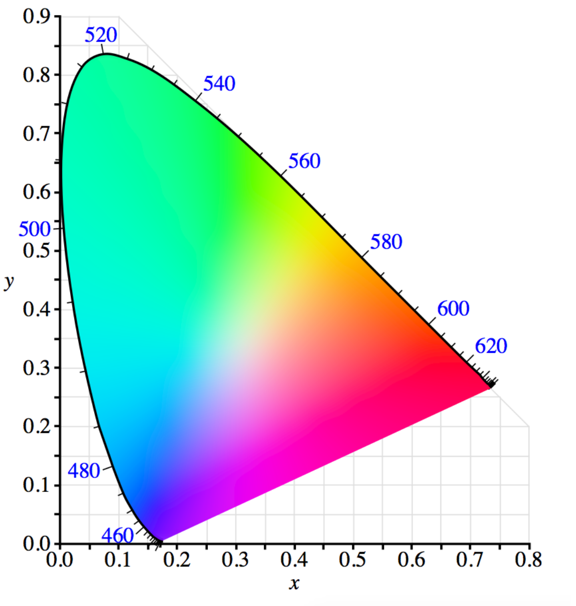


Figure 4.1: The CIE *xy* chromaticity diagram. The curved edge represents all monochromatic wavelengths in nanometers and is called *spectral locus*. The straight, lower edge represents all additive mixtures of short- and long-wave stimuli and is called the *purple line* [Reinhard *et al.* 2010:32f].

standard feature in most modern 3D engines. Another required feature is the possibility of using custom pixel or compute shader code, written in Microsofts DirectX High-Level Shader Language (HLSL), to allow for sufficient freedom and hardware access for the implementation. It is beneficial if the engine is at a mature stage of development and already has a large user base. This makes it easier to find documentation resources and other aids when developing for the engine. It also broadens the circle of potential beneficiaries. Three particularly widely used engines were examined more closely for this thesis: CryEngine developed by Crytek, Unity developed by Unity Technologies and Unreal Engine developed by epic games.

While the CryEngine delivers very convincing graphics, it only offers ways of influencing the TMO via a GUI. Therefore, it would be a more involved process to implement a custom TMO for this engine and source code access would be mandatory for this.

More recently, both Unreal Engine and Unity have added support for custom HLSL shaders in addition to standard tone mapping methods. The process of actually writing and debugging a TMO turned out to be the most convenient in Unity. It includes both a couple of example TMOs as well as sufficient documentation and support for this task. For these reasons, Unity was chosen for the implementation.

5 Implementation Details

This chapter walks the reader through the specifics of this implementation of the TMO. The source code for the C# UNITY script and the HLSL compute shader code can be found attached to this thesis.

UNITY supports multiple languages for scripting. This implementation uses C# for its `Tonemapper` class. The script's computations are executed on the CPU. For calculations on the GPU, a compute shader is used. UNITY compute shaders are written in HLSL syntax and have the file extension `.compute`. They need to contain `#pragma` compilation directives in the beginning to declare which functions are compute shader kernels.

UNITY compute shaders are invoked using the `ComputeShader` class provided by the engine. It includes functions like `SetBuffer` to declare data that needs to be sent to the GPU. To launch a kernel, the `Dispatch` function is used, which takes the index of the kernel and the number of groups that should be launched for each of the three dimensions as arguments.

UNITY includes a range of standard assets. Among them is a utility class called `PostEffectsBase`. This class can be extended to build full screen image post-processing effects which can be attached to `Components` of the type `Camera`. An effect like this only needs to provide an `OnRenderImage` function with two parameters of the type `RenderTexture`, one for the source image and one for the resulting image.

An `ImageEffectTransformsToLDR` instruction exists to inform the engine about the fact that the effect acts as a TMO for HDR images.

5.1 Color Space Conversion

On line 93, the first shader kernel is launched in the C# script. It uses one thread per pixel of the source image. The vector containing the RGB information is of type `float4`. It is multiplied by a 3×3 standard RGB to XYZ conversion matrix, specified by the INTERNATIONAL TELECOMMUNICATION UNION [Reinhard *et al.* 2010:35]. Since neither alpha nor depth values are used by the TMO, the fourth value of the vector may be dropped. Next, x and y are computed using Equations 4.1 and 4.2. The $\log 10$ is computed for the luminance values in preparation for the histogram adjustment.

The data is stored in three separate two-dimensional data structures of the type `RWTexture2D`, typed with `<float>`. In DirectX terminology, this is called an *unordered access view* (UAV) which allows for simultaneous read and write operations by multiple threads, without the need for atomic operations. It is the most flexible when typed with single-component 32-bit element types.

5.2 Reduction to Foveal Image

The second shader kernel follows immediately after the first. Here, the thread group size is critical for the result. Like for the first kernel, one thread is used for each pixel of the source image and each group covers a quadratic area that is `NUM_THREADS_X` wide. This value has an impact on the performance, since it determines the resulting count of histogram samples. Reasonable values are 16 or 32.

To avoid weighting calculations for non-full groups at the edges of the image, the operator only uses full groups for the foveal image.

The kernel calculates the mean value for each group and stores it in a buffer of UAV type `RWStructuredBuffer` for the following code to read out. This task is not feasible without inter-thread communication. The HLSL function `GroupMemoryBarrierWithGroupSync` is used for memory synchronization. The kernel works by first dividing the data by the group size and copying it into a `groupshared` array. This is followed by a parallel add-reduce algorithm.

5.3 Histogram Creation

All following steps until the final tone mapping are performed in the C# script, or in other words, on the CPU. GPU code needs a certain amount of parallelism and data throughput to be more time efficient than a sequential implementation [Konstantinidis 2014]. Since the operator only requires the histogram adjustment to be performed on the reduced set of luminance values of the foveal image, the maximum number of parallel computations possible is either limited to the sample count or the histogram bin count. For an image resolution of 1920×1080 and a group height of 32 the sample count would be $(1920/32) * (1080/32) = 2025$. If this is enough to fully utilize the capacities of the GPU is highly hardware dependent.

	rods	cones
fast neural adaptation	0.15	0.08
slow neural adaptation	60.00	30.00
pigment bleaching	400.00	110.00

Figure 5.1: The different time constants t_0 used in seconds

To create the histogram, the minimum and maximum luminance values need to be known, as the histogram will be spanned between those. In the same `for` loop, the mean luminance value is computed for adaptation modeling. The minimum luminance value has a lower bound of 10^{-4}cd/m^2, as there needs to be a lower bound of human vision [Reinhard *et al.* 2010:378].

In the next loop, the histogram bin values are computed. The number of histogram bins does not need to be high for sufficient accuracy. This implementation uses a bin count of $N = 128$.

5.4 TVIA Function and Adaptation State

On line 135, a separate function is used to update the adaptation values for the various adaptation mechanisms, assuming the mean log luminance to be the background luminance of the image. UNITY provides a static variable `Time.deltaTime` which is the time in seconds it took to complete the last frame. This is used as time parameter t in Equations 3.11 and 3.16. The values used as time constants t_0 can be found in Figure 5.1. Most of these are specified in Irawans paper. For slow neural adaptation it was necessary to work with Figure 11 in the paper to approximate the values for t_0 that were used by Irawan et al.

In the `for` loop on line 142, the ceilings are computed using Equation 3.4 combined with the TVIA Function 3.7. For each luminance value a new temporary adaptation state is calculated in the `calculateTVIA` function, according to the set partial adaptation time, which should be around 300ms, as this is roughly the average duration of saccadic eye movements.

Depending on whether the rod or the cones are better adapted to the particular luminance, `calculateTVIA` returns the JND threshold luminance for the respective system. Subsequently, the implementation makes a small modification, in that it slightly increases the threshold for the rods for luminances above 10cd/m^2. This is to accommodate the fact that most sources state that the rod system is presumably not active in normal to bright light conditions [Reinhard *et al.* 2010:243][Hood and Finkelstein 1986:5-30].

Data from Walraven was used to calculate the Weber constant and the just noticeable change in response ΔR [Walraven *et al.* 1993], since these values are not specified in Irawans paper. The well documented TVI function $\Delta L(L) \hat{=} \Delta L(L, \sigma(L))$ is needed for these calculations. For the Weber constant, one has to simply compute the ratio of the luminance of a typical display device to the respective TVI value $L_{\mathrm{d}}/\Delta L(L_{\mathrm{d}})$. For the just noticeable change in response ΔR Equation 2.1 is used and $\sigma(L) = L$ for all L is assumed. Then the difference between $R(L, L)$ and $R(L + \Delta L(L), L)$ is computed for a range of luminance values. The smallest of those is used for ΔR [Irawan *et al.* 2005]. These calculations resulted in a value of 0.06 for the Weber constant and a value of 0.013 for ΔR.

5.5 Low Dynamic Range Detection and Handling

An intermediate result is kept during the process of calculating the ceilings to be able to adjust $\log(L_{\mathrm{d,max}}) - \log(L_{d,min})$ without completely recomputing the ceilings. On line 156, the check for low dynamic range can be found. If it returns true, the display dynamic range is recalculated to exactly meet the condition of Equation 3.5. Then the ceilings are adjusted accordingly.

The next step is to decide what part of the display dynamic range the image should allocate. To determine by how much the range needs to be reduced, the ratio of the real and the modified display dynamic range is taken. Then the world dynamic range is compared to the unmodified display dynamic range, to assess were to place the luminances.

The following histogram adjustment step can be completely omitted in the LDR case. Instead, the ceilings may be used as histogram for the cumulative distribution function because, by definition, the redistributed histogram will always be equal to the adjusted ceilings.

5.6 Histogram Adjustment

For histogram adjustment the histogram is trimmed in the first `for` loop. In the same loop the sum of all ceilings of empty bins and the sum of all unmodified bin counts are computed. The latter value is used in the next step to redistribute the trimmed values to untrimmed bins proportionally their initial values. If all untrimmed bins are filled to their corresponding ceilings, the rest gets distributed over the empty bins, this time proportionally to their respective ceilings.

5.7 Mapping Function and Tone Mapping

Obtaining the cumulative distribution function is trivial in sequential programming. During this step, the relative dynamic range reduction is applied for the LDR case.

The resulting array defines the slope of the desired tone mapping function. It is transferred to the GPU, where it is used in the third kernel. Here, the corresponding histogram bins are computed. Values in between bins are interpolated linearly. In the case of the Unity Engine, the target range of values for an LDR image is $[0, 1]$. The results are normalized to fit this interval.

Finally, the luminance values are recombined with the chromaticity information and converted back to Unitys RGB color space using the inverted standard RGB to XYZ conversion matrix. The resulting texture is read out via the C# script and copied into the `destination` texture.

This concludes the walkthrough of the new implementation. The image is now converted to LDR and ready for further processing and finally being displayed on a screen.

6 Results and Further Exploration

6.1 Results

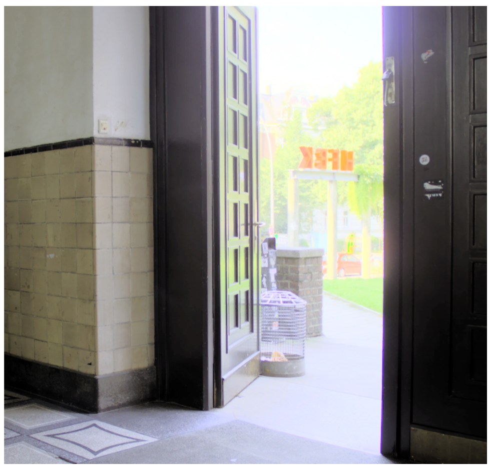
For the purpose of this thesis, the operator was tested with HDR photographs that were generated by combining various exposures of the same scene into one file of the `.hdr` *radiance* file format. Luminance values obtained this way can only be approximations, because no exact measuring hardware was used. The operator works best in an environment with realistic, physically based lighting calculations.

Still, the resulting images are well suited to demonstrate the effects and benefits of the TMO. It effectively maximizes the visibility of details in HDR scenes or limits it in very dark or empty scenes. Because of its ability to boost the apparent contrast of an image, the results for HDR scenes come out as visually pleasing. Overall, the resulting images appear balanced and natural.

The operator's modeling of adaptation and maladaptation is convincing, see Figure 6.2. Dark adaptation takes more time than light adaptation. The detailed reduction of distinguishable differences in brightness in the case of maladaptation is a big strength of the operator, and is rarely found even in visibility simulating tone mapping methods.

However, at the time of writing, a few artifacts can still be noticed with the state of the implementation. In some cases, the switch between the HDR and the LDR case can create a noticeable jump in image brightness. This happens when the operator is not able to determine the correct range of display luminance values for the LDR image. This problem is not trivially solvable when there is no guarantee for physically correct luminance values in the source image.

Another source of flickering and other irregularities can be changing minimum and maximum luminance values within the scene. Different kinds of interpolation of changes in the mapping function are possible solutions to this problem. For example, the calculation of the minimum, maximum and mean luminances could be weighted, so that center pixels have more influence than those at the edges of the image. This way, very



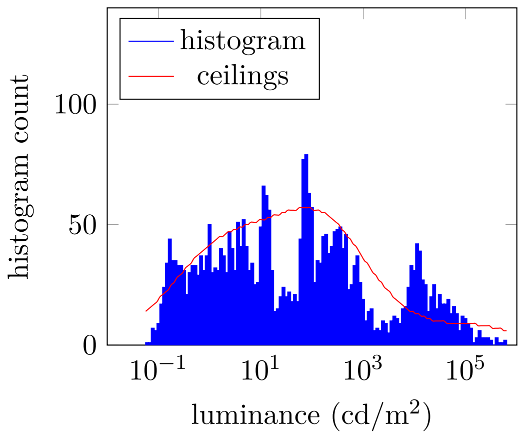


Figure 6.1: HDR door scene with histogram and ceilings in a steady adaptation state.

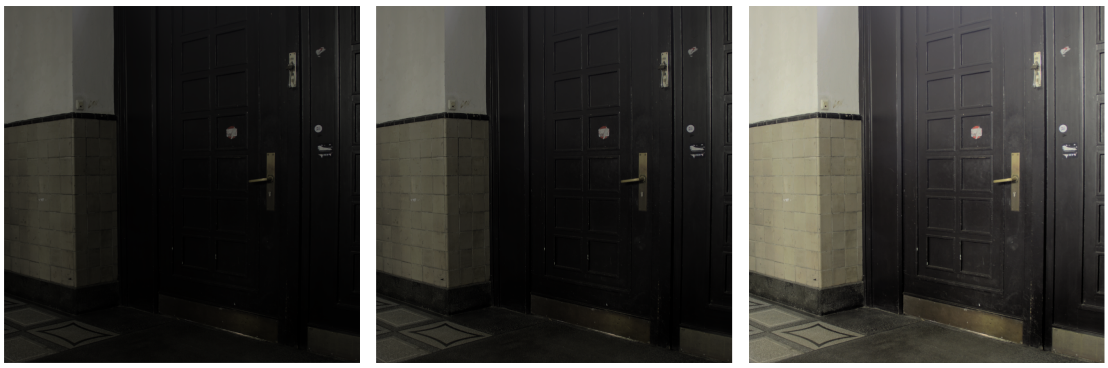
bright or very dark objects, entering the image from the sides, will not cause sudden jumps in luminance mapping.

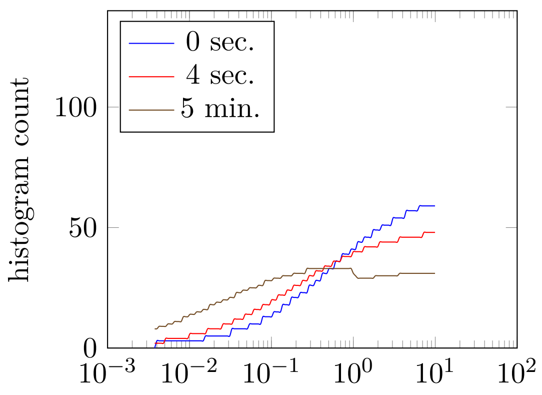
6.2 Performance Considerations

Performance is critical for real-time graphics programming and there likely is great potential for performance optimization in the new implementation. Still, so far, all tests indicated that it is already well suited for real-time application.

The operator was tested on a computer containing an Nvidia Geforce GT 640M graphics card with 512 MB of graphics memory and an Intel Core i5-3335S CPU with 4 processing units and a clock rate of 2.7 GHz. The tools provided by the engine were used to measure performance. The TMO takes about 8 ms to render a 1920×1080 image. The number of histogram bins `NUM_HISTOGRAM_BINS` does not seem to have a great impact on either the performance or the resulting images. Values between 32 and 256 worked well. The width of a thread block `NUM_THREADS_X` enables the best rendering times at a value of 16 on this hardware configuration. Image resolution was found to have a linear relation to rendering time, with the rendering time in milliseconds being roughly 5 times the number of megapixels in the image.

Performance optimization of graphics code is a highly sophisticated process and requires comprehensive knowledge of current graphics hardware. As described and justified in Section 5.3, only parts of the operator were implemented using GPU code.





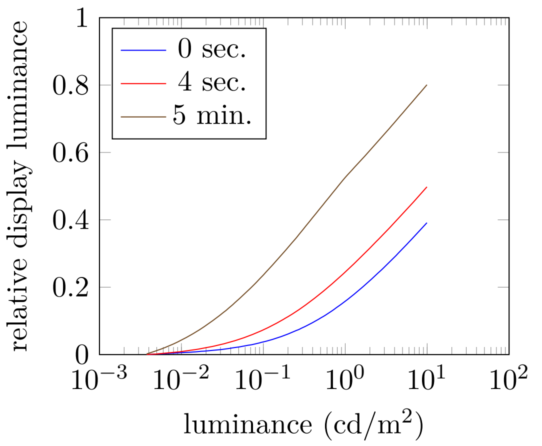


Figure 6.2: Demonstration of dark adaptation over time using the door scene from Figure 6.1 with the door closed. The system starts optimally adapted to the scene with the door open. Top: resulting images after 0 seconds, 4 seconds and 5 minutes of adaptation time. Middle: shapes of the respective ceilings. Bottom: resulting mapping functions. Note that the mapping functions do not reach the maximum value because the scene is LDR.

But it is completely possible that another split of the different tasks between CPU and GPU is more efficient on common hardware configurations.

Delegating the calculation of the minimum, maximum and mean luminances of the foveal image, or even the histogram creation, over to the GPU could have a positive performance impact. Since transferring data back and forth between CPU and GPU poses a performance bottleneck, an implementation that works entirely on the GPU could also be considered. Although, a parallel implementation of histogram redistribution presumably would be much more complex. It would require a lot of memory synchronization and utilize only a small part of the capacity of the GPU.

Overall, such involved tone mapping methods like the one by Irawan et al. will always have a considerable performance impact compared to simple single kernel operators.

6.3 Further Refinements

This thesis focuses on the main task of a tone mapping operator: the redistribution of luminance values. But since the TMO proposed by Irawan et al. is in fact a visual system simulator, an all-encompassing version would include additional steps to model even more effects of visual perception.

When simulating visual perception, an important effect is *glare*. It is caused by bright light that is being refracted by small particles in the lens and vitreous body of the eye. This reduces the perceived contrast in areas surrounding bright lights. The effect is especially noticeable when the visual system is dark adapted. Since peak luminances of standard LDR displays are not bright enough to trigger glare effects in the average observer, such patterns are directly printed to the image to improve the perceived realism. Because humans are so used to seeing glare effects around bright objects, such an artificial version can have a considerable impact on the perceived brightness of objects, which is a desirable property when seeking perceptual realism [Reinhard *et al.* 2010:367ff].

Other noteworthy effects are those of night vision. When the rods become the dominating photoreceptors in the dark, their properties influence the way the images are perceived. Because there are fewer rods present in the region of the fovea, visual acuity drops when the cones stop operating at low light intensities [Greule 2015:66]. Furthermore, since rods alone are not able to discriminate colors, color vision is lost in the dark. In addition, color perception in the mesopic range is different from the photopic range, because blue photoreceptors are more sensitive to low luminance values than red and green photoreceptors [Greule 2015:64]. This also causes night scenes to appear blueish. This fact can be utilized to create the impression of night and darkness, despite the relatively high luminance of standard display devices.

These effects usually are applied in the context of tone mapping because access to the HDR luminance information and the adaptation state is needed for their implementation. A good report about these and other effects of visual perception, and how they can be implemented for a game engine, can be found in the thesis by Hellsten [Hellsten 2007].

6.4 Conclusion

Overall, the TMO described by Irawan et al. proved to be a satisfying visibility simulation tool that accounts for various effects of luminance perception. It works by modeling and approximating mechanisms of visual perception. However, evaluating to what extent the results meet their goal is a hard problem. The process of visual perception is not yet fully understood and comprises more than just discrimination of luminance values. Hence, mathematically modeling photochemical processes might not be the single best way to achieve perceptual realism. From this point of view the complexity of the implemented operator might not be fully justified and a more simple method might generate equally realistic results.

In the long term, the ultimate way to realistically render virtual worlds lies with the emerging HDR display devices. Prior to that, the operator implemented for this thesis is a good fit for different purposes in realistic 3D rendering.

Bibliography

Clifford, C. W. G., Webster, M. A., Stanley, G. B., Stocker, A. A., Kohn, A., Sharpee, T. O., and Schwartz, O. (2007). Visual adaptation: Neural, psychological and computational aspects.

Delvin, K. (2002). A review of tone reproduction techniques.

Eilertsen, G., Wanat, R., Mantiuk, R. K., and Unger, J. (2013). Evaluation of tone mapping operators for hdr-video. *Pacific Graphics*, **32**.

Greule, R. (2015). *Licht und Beleuchtung im Medienbereich*. Hanser, first edition.

Hellsten, J. (2007). Evaluation of tone mapping operators for use in real time environments.

Hoffman, J. E. and Baskaran, S. (1995). The role of visual attention in saccadic eye movements. *Perception and Psychophysics*.

Hood, D. C. and Finkelstein, M. A. (1986). Sensitivity to light. In K. R. Boff, L. Kaufman, and J. P. Thomas, editors, *Handbook of Perception and Human Performance: Sensory Processes and Perception*. John Wiley and Sons, Inc.

Irawan, P., Ferwerda, J. A., and Marschner, S. R. (2005). Perceptually based tone mapping of high dynamic range image streams. *Eurographics Symposium on Rndering*.

Kaiser, P. K. (2009). The joy of visual perception: A web book. www.yorku.ca/eye/darkadap.htm. Last fetched on 08/02/2015.

Konstantinidis, E. (2014). Least required gpu parallelism for kernel executions. http://parallelplusplus.blogspot.de/2014/10/least-required-gpu-parallelism-for.html. Last fetched on 08/13/2015.

Naka, K. I. and Rushton, W. A. H. (1966). S-potentials from luminosity units in the retina of fish (cyprinidae). *Journal of Physiology*, **185**.

Pattanaik, S. N., Tumblin, J., Yee, H., and Greenberg, D. P. (2000). Time-dependent visual adaptation for fast realistic image display.

Reinhard, E., Ward, G., Pattanaik, S., Debevec, P., Heidrich, W., and Myszkowski, K. (2010). *High Dynamic Range Imaging*. Morgan Kaufmann, second edition.

Varcholik, P. (2014). *Real-Time 3D Rendering with DirectX and HLSL*. Addison-Wesley, first edition.

Walraven, J., Enroth-Cugell, C., Hood, D. C., MacLeod, D. I. A., and Schnapf, J. L. (1993). The control of visual sensitivity: Receptoral and postreceptoral processes. In L. Spillman and J. S. Werner, editors, *Visual Perception: The Neurophysiological Foundations*. Academic Press.

Ward-Larson, G. W., Rushmeier, H., and Piatko, C. (1997). A visibility matching tone reproduction operator for high dynamic range scenes.

Wilt, N. (2013). *The CUDA Handbook*, chapter 8. Addison-Wesley Professional.

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, August 31, 2015 Yannick Rietz