Fast Local Tone Mapping, Summed-Area Tables and Mesopic Vision Simulation

Marcos Slomp, Michihiro Mikamo and Kazufumi Kaneda
Hiroshima University
Japan

1. Introduction

High dynamic range (HDR) imaging is becoming an increasingly popular practice in computer graphics, bringing unprecedented levels of realism to computer-generated imagery and rich detail preservation to photographs and films. HDR imagery can embrace more accurately the wide range of light intensities found in real scenes than its counterpart, low dynamic range (LDR), which are tailored to the limited intensities of display devices. Simply put in computer terminology, think of HDR as a *very large*, *continuous* range of intensities encoded in a floating-point representation (but not necessarily), while LDR translates to *coarse*, *quantized* ranges, usually encoded as 8-bit integers and thus limited to 256 *discrete* intensities.

An important term in HDR imaging is the *dynamic range* or *contrast ratio* of a scene, representing the distance between the lowest and highest intensity values. Luminance in the real world typically covers 14 orders of magnitude (dynamic range of $10^{14} : 1$), ranging from direct sunlight (10^5 up to $10^8 cd/m^2$) to shallow starlight (10^{-3} down to $10^{-6} cd/m^2$), while typical image formats and commodity display devices can cope with only 2 up to 4 orders of magnitude (maximum contrast ratio of $10^4 : 1$) [Ledda et al. (2005)].

A challenging task on HDR imaging consists in the proper presentation of the large range of intensities within HDR imagery in the much narrower range supported by display devices while still preserving contrastive details. This process involves intelligent luminance (dynamic range) compression techniques, referred to as *tone reproduction operators* or, more commonly, *tone mapping operators* (TMO). An example is depicted in Figure 1.

The process of tone mapping shares similarities with the light adaptation mechanism performed by the Human Visual System (HVS), which is also unable to instantly cope with the wide range of luminosity present in the real world. Even though the HVS is only capable of handling a small range of about 4 or 5 orders of magnitude at any given time, it is capable of dynamically and gradually shift the perceptible range up or down, appropriately, in order to better enclose the luminosity range of the observed scene [Ledda et al. (2004)]. Display devices, on the other hand, are much more restrictive, since there is no way to dynamically improve or alter their inherently fixed dynamic range capabilities.

Tone mapping operators can be classified as either *global* (spatially-uniform) or *local* (spatially-varying). Global operators process all pixels uniformly with the same parameters, while local operators attempt to find an optimal set of parameters for each pixel individually, often considering a variable-size neighborhood around every pixel; in other words, the

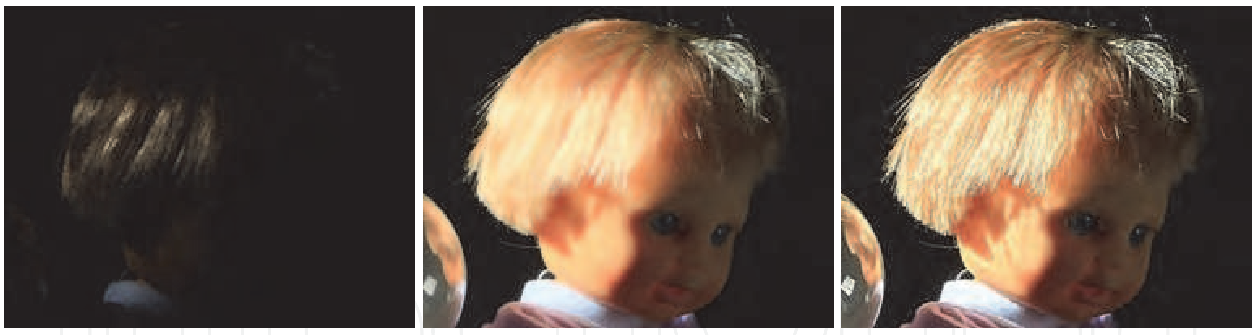

(a) No Tone-Mapping (clamped) (b) Global Operator (c) Local Operator

Fig. 1. A comparison between the global and the local variants of the photographic operator of Reinhard et al. (2002). Local operators are capable of preserving more contrastive detail.

amount of luminance compression is locally adapted to the different regions of the image. Determining such local vicinities is not straight-forward and may introduce strong haloing artifacts if not handled carefully. Although all local tone-mapping operators are prone to this kind of artifacts, good operators try to minimize their occurrence. Overall, local operators retain superior contrast preservation over global operators, but at much higher computational costs. Figure 1 illustrates the visual differences between each class of operator.

Due to the prohibitive performance overhead of local operators, current HDR-based real-time applications rely on either global operators or in more simplistic exposure control techniques. Such exposure mechanisms can be faster than global operators, but are not as automatic, often requiring extensive manual intervention from artists to tailor the parameters for each scene and vantage point; worse yet, if objects or light sources are modified, this entire tedious process would have to be repeated.

Apart from contrast preservation and performance issues, most tone-mapping operators focus exclusively on luminance compression, ignoring chromatic assets. The HVS, however, alters color perception according to the overall level of luminosity, categorized as *photopic*, *mesopic* and *scotopic*. The transition between these ranges is held by the HVS, stimulating visual cells – cones and rods – according to the overall lighting conditions. Cones are less numerous and less responsive to light than rods, but are sensitive to colors, quickly adapt to abrupt lighting transitions and provide sharper visual acuity than rods.

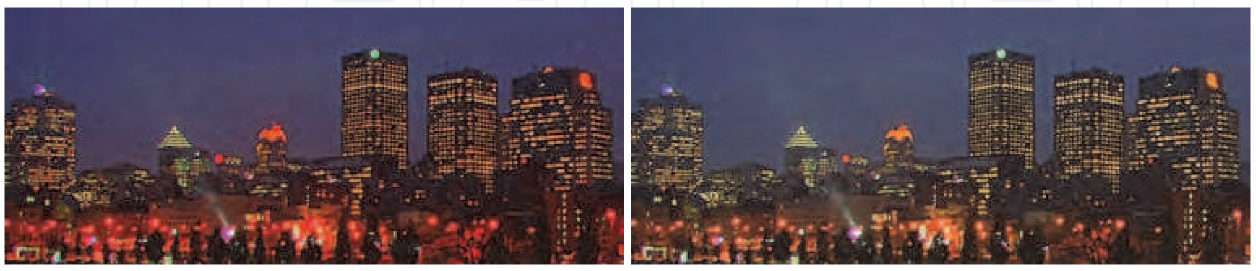

(a) without mesopic vision simulation (b) with mesopic vision simulation (proposed)

Fig. 2. An evening urban scene (a) without mesopic vision simulation and (b) with the mesopic vision strategy later described in Section 4. As it can be seen the sky changes from purple to a more blueish tone, while distant artificial lights shift from red to orange/yellow.

At photopic conditions (think of an outdoor scene at daylight) color perception is accurate since *only* cones are being stimulated. When the lighting conditions turn to scotopic (think of starlight), colors can no longer be discerned because cones become completely inhibited while rods become fully active. *Mesopic vision* is a transitory range in-between where rods and cones are both stimulated simultaneously. At this stage colors can still be perceived, albeit in a distorted fashion: the responses from red intensities tend to fade faster, thus producing a peculiar *blue-shift* phenomenon known as Purkinje effect [Minnaert (1954)].

Moonlit scenes, for example, present a blueish appearance even though the light being reflected by the moon from the sun is not anywhere close to blue in nature, but it is actually *redder* than sunlight [Khan & Pattanaik (2004); van de Hulst (1957)]. Besides this overall blueish appearance at extreme mesopic vision conditions, the same phenomenon also causes otherwise red features to appear in a much darker tone, or in an *orangeish* or *yellowish* tone; similarly, purple tonalities tend to be noticed in dark blue colorations. Refer to Figure 2 for a depiction of a scene with and without the proposed mesopic vision filter.

The explanation of such effect comes from the fact that in mesopic conditions rods respond better to short wavelengths (blue) stimuli than long and medium wavelengths (red, yellow, green). As the overall luminosity conditions dim, but *before* rods completely take over the visual system (scotopic vision), color perception shifts towards the currently most sensible rod's wavelengths, that is, around blue.

Mesopic vision reproduction for computer-generated images has immediate application on artistic assets and architectural lighting design. Perhaps an even more relevant application is on road engineering and signalization planning, by reproducing the overall experience of drivers subjected to adverse lighting conditions.

This chapter focuses on an efficient GPU-based implementation of the local-variant of the photographic operator introduced by Reinhard et al. (2002) that achieves real-time performance without compromising image quality. The presented approach approximates the costly variable-size Gaussian convolutions with efficient box-filtering through Summed-Area Tables (SAT) [Crow (1984)], as suggested earlier by Slomp & Oliveira (2008). This work, however, utilizes a technique based on balanced-trees for the SAT generation [Blelloch (1990)] instead of the originally employed (and more widely spread) recursive-doubling algorithm [Dubois & Rodrigue (1977)]. This algorithmic change vastly improves performance, in particular for large-sized images, at the expense of a more involved implementation.

More importantly, a novel, fast and universal perceptually-based method to reproduce color shifts under mesopic vision conditions is introduced. Unlike previous works, the present work can deal with the *entire* mesopic vision range. The method builds upon initial investigations of Mikamo et al. (2009), who suggested the use of perceptual metrics for mesopic vision reproduction derived from psychophysical experiments performed by Ikeda & Ashizawa (1991). The proposed *uniform* mesopic filter is not bound exclusively to the aforementioned photographic operator, but can in fact suit *any* other existing TMO, since the chromatic adjustment stage is *decoupled* from the luminance compression stage. Finally, this chapter also describes how to exploit the foundations of the local photographic operator to further extend the proposed uniform filter in a *spatially-varying* manner and achieve more plausible mesopic vision results without performance penalties. Only a tiny performance footprint is incurred to the overall tone-mapping process, hence being friendly to real-time applications.

External resources

The demo program is available for download at the following URL:

`http://www.eml.hiroshima-u.ac.jp/demos/fast-mesopic-tmo`

2. Related work

A proper sound review on tone-reproduction operators would consume several pages of this document, since the available literature is vast and rich. Therefore, this background review section will focus on pertinent research related to *real-time local tone-mapping* as well as *mesopic vision simulation*. The curious reader is referred to Reinhard et al. (2010) for an extensive survey on tone-mapping and HDR imaging techniques.

2.1 Real-time local tone-mapping

Local operators, due to the varying-size filtering and halo-avoidance requirements, impose great challenge for faithful real-time implementations, even with parallel power of modern programmable graphics hardware (GPU). From all existing TMOs, the photographic operator introduced by Reinhard et al. (2002) has received special attention from researchers and practitioners, mainly due to its simplicity (few parameters), automaticity (no user intervention), robustness (extreme dynamic ranges) and perceptually-driven approach (local adaption is guided by an HVS-based brightness perception model). There are two variants of the operator, a global and a local one. The photographic operator is reviewed in Section 2.3.

The local variant of the photographic operator, at its core, makes use of differences of Gaussian-filtered (DoG) luminance images at various scales. Unfortunately, convolving an image with variable-size Gaussian kernels is a computationally demanding operation. Several attempts were made to accelerate this filtering to provide real-time frame rates.

Goodnight et al. (2003) investigated the implementation of local operators on GPU. Their best result was with the photographic operator, where they implemented the 2D Gaussian convolution using separable 1D kernels in a two-stage approach. Their implementation is by no means naïve, but makes clever use of the efficient 4-component vector dot product instruction provided by GPU architectures. This reduces the number of required filtering passes and thus achieves better performance. Despite all their optimization efforts, the technique only runs at interactive rates when using a small subset of the originally required adaptation scales (i.e., the first few Gaussian-filtered luminance images at small convolution profile sizes; see Section 2.3). A limited number of adaptation zones gradually causes the operator to fall-back to the global variant case, thus sacrificing important contrastive details (see Figure 3-b).

Krawczyk et al. (2005) proposed an approximation for the local photographic operator on the GPU. The Gaussian-filtered luminance images are downsampled to $1/4$, $1/16$, and $1/64$ of their original size. Convolutions are then performed using smaller approximate Gaussian kernels of fixed size (always 7 pixels wide), with intermediate filtered results being reused. The blurred images are then upsampled back to the original size prior to evaluating the DoG model. This strategy significantly speeds up the operator, but not without inherent limitations: there is excessive blurring across high contrast edges being caused by the downsampling-upsampling process, potentially introducing noticeable halo artifacts (see

Figure 3-c). Besides these shortcomings, their main contributions concentrate on reproducing perceptual effects such as temporal luminosity adaptation, glare and loss of vision acuity.

Slomp & Oliveira (2008) replaced the expensive variable-size Gaussian convolutions of the operator with box-filtering powered by Summed-Area Tables (SAT) [Crow (1984)]. Summed-Area Tables allow arbitrary rectangular portions of an image to be efficiently box-filtered in constant time $O(1)$. Although box-filtering provides a very crude approximation of Gaussian-filtering, within the context of the photographic operator results are nearly indistinguishable from the original operator, as demonstrated in Figure 3-d. It is worth mentioning that box-weighted kernels were also used in the context of tone mapping by Pattanaik & Yee (2002) in their bilateral-filtering algorithm. The process of SAT generation can be efficiently implemented on the GPU, thus not only mitigating the occurrence of halos, but also substantially accelerating the operator, even when compared against the fast method of Krawczyk et al. (2005). A review on SAT and their generation is provided in Sections 2.4, 3.2 and 3.3.

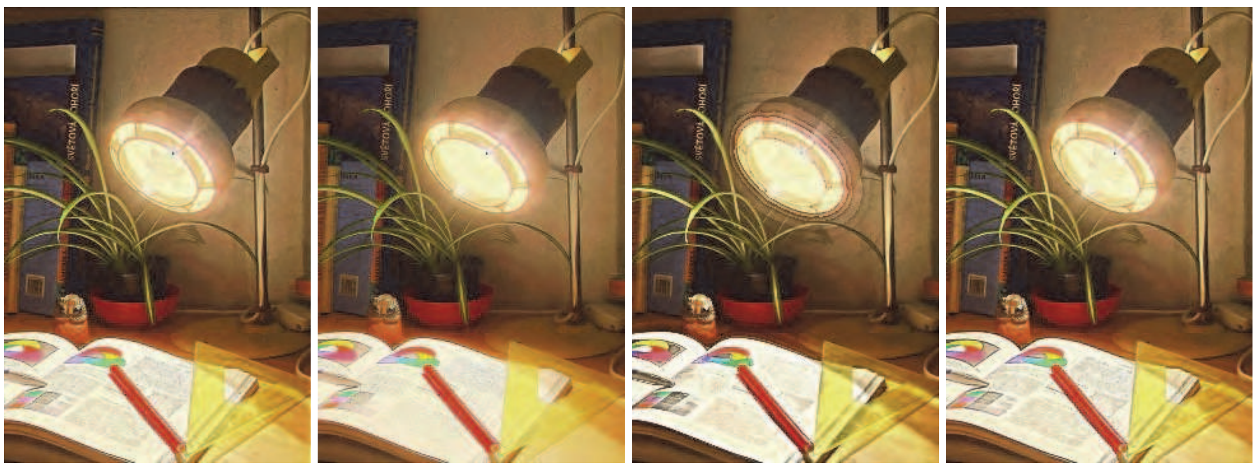

(a) Reinhard et al. (2002) (b) Goodnight et al. (2003) (c) Krawczyk et al. (2005) (d) Slomp & Oliveira (2008)

Fig. 3. Comparison between different implementations of the local photographic operator. Note that words vanished from the book in (b), and halos appeared around the lamp in (c).

This Chapter describes a more attractive method for SAT generation based on balanced-trees [Blelloch (1990)]. The originally employed recursive-doubling algorithm [Dubois & Rodrigue (1977)] requires less passes, but performs more arithmetic operations and is more sensitive to cache thrashing. In spite of the advantages and readily availability since long, the adoption of the balanced-tree approach by the computer graphics community was oddly timid.

2.2 Mesopic vision simulation

Despite luminance compression, many tone mapping operators have concentrated efforts on reproducing perceptual effects recurrent from the HVS, most notably: temporal luminosity adaption, scotopic vision simulation, loss of visual acuity, and glare. The literature on these topics is broad; refer to Reinhard et al. (2010) for examples of each category. However, and quite surprising, very little has been done on reproducing mesopic vision; below is a summary of the most remarkable research related to mesopic vision in tone reproduction.

Durand & Dorsey (2000) have specialized the rod-cone interaction model of Ferwerda et al. (1996) to better suit night scenes, adding support for chromatic adaptation and color shifts, among other effects. Even though the method does not explicitly address mesopic vision, the underlying framework can be tunned to handle specific *subranges* of mesopic vision with some degree of fidelity, but not without some user interaction. The technique is fast and runs at interactive rates since it inherits the same *global* characteristics of Ferwerda et al. (1996).

Khan & Pattanaik (2004) have also proposed a blue-shift filter based on rod-cone interaction. Their technique is designed primarily for moonlit scenes, without the intervention of artificial light sources. Overall, the method tends to introduce very strong blue hues on the scene, almost completely mitigating other color tones. This is due to their hypothesis that *only* short wavelength cones (blue) would respond to light in a naturally moonlit scene. The technique itself does not rely on HDR imagery and therefore can not be classified as a TMO.

Kirk & O'Brien (2011) have proposed a color shift model for mesopic vision, building upon the biological model and fitting experiments of Cao et al. (2008). Their model can be combined with existing TMO since chrominance adaptation is decoupled from luminance compression. The chrominance adaption strategy itself is spatially-uniform (global), but nonetheless computationally demanding and incompatible with interactive frame rates. The overhead comes from the fact that *spectral* information is required; in other words, the input images must be capable of approximating the continuous distribution of energy at every pixel using some higher dimensional representation. Moreover, the method assumes that the spectral sensitivity of the camera is known; if not provided by the manufacturer, a calibration procedure is required to estimate the sensitivity. Traditional HDR imagery can also be targeted, although not without first estimating the unknown spectral distribution.

The mesopic vision reproduction strategy to be introduced in this Chapter takes a more generic avenue than the methods above. Chrominance adaptation is also decoupled from luminance compression, thus suitable to co-exist with nearly any available TMO. The *entire* mesopic vision range is handled systematically without special conditions. The technique is targeted for widely available HDR imagery, hence unconventional spectral information is not required. The performance overhead introduced is most likely negligible. Two variants of the proposed filter are presented: a more generic spatially-uniform one and a spatially-varying one that uses the infrastructure available from the local photographic operator. The technique, however, should *not* be misunderstood as a definitive replacement to the more specialized aforementioned approaches, but rather as an accessible framework that does not incur into prohibitive performance penalties, acquisition and development costs or image quality degradation.

2.3 Review of the photographic tone reproduction operator

Reinhard et al. (2002) digital photographic operator uses a photographic technique called Zone Systems [Adams (1983)] as a conceptual framework to manage luminance compression. The goal is to map the *key-value* (subjective predominant intensity) of a scene to the *middle-gray* tone of the printing medium (*middle-intensity* of the display device in this case) and then linearly rescaling the remaining intensities accordingly. This can be intuitively thought as setting an exposure range in a digital camera. An overview of the operator is shown in Figure 4.

Given an HDR image, a good estimation for its key-value is the *geometric average* (*log-average*) of its luminances, which is less susceptible to small outliers than plain arithmetic average:

$$\widetilde{L} = exp\left(\frac{1}{N}\sum_{x,y} log(L(x,y) + \delta)\right) \tag{1}$$

where N is the number of pixels in the image, $L(x,y)$ is the luminance at the pixel with coordinates (x,y), and δ is a small constant (i.e., $\delta = 0.00001$) to prevent the undefined $log(0)$.

Each pixel luminance is then *scaled* based on the Zone System printing zones, with the estimated key-value $\widetilde{L}$ being mapped to the middle-grey range:

$$L_r(x,y) = L(x,y)\frac{\alpha}{\widetilde{L}} \tag{2}$$

where $\alpha = 0.18$ for average-key scenes (akin to automatic exposure control systems present in digital cameras [Goodnight et al. (2003)]). For high-key and low-key scenes, the parameter α has to be tweaked, but an automatic estimation strategy is described by Reinhard (2003).

For the global-variant of the operator, each relative luminance $L_r(x,y)$ is mapped to a normalized displayable range $L_d(x,y) \in [0,1)$ as follows:

$$L_d(x,y) = \frac{L_r(x,y)}{1 + L_r(x,y)} \tag{3}$$

This global operator is prone to conceal contrastive detail. A better approach is to locally adapt the contrast of each region, individually, similar to photographic *dodging-and-burning*:

$$L_d(x,y) = \frac{L_r(x,y)}{1 + L_r^{s_{max}}(x,y)} \tag{4}$$

where $L_r^{s_{max}}(x,y)$ is the Gaussian-weighted average of the largest isoluminant region s_{max} around each isolated pixel where no substantial luminance variation occur. The term $L_r^{s_{max}}(x,y)$ can be more intuitively thought as a measurement of *local area luminance* around a pixel. It is imperative to judiciously determine these isoluminant regions because otherwise strong haloing artifacts are prone to appear around high-contrast edges of the image.

The dodging-and-burning approach of the local photographic operator uses *differences of Gaussian*-filtered (DoG) portions of the scaled luminance image $L_r(x,y)$ at increasing sizes to iteratively search for these optimal isoluminant regions, according to the following expression:

$$V_s(x,y) = \frac{L_r^s(x,y) - L_r^{s+1}(x,y)}{2^{\phi}\alpha/s^2 + L_r^s(x,y)} \tag{5}$$

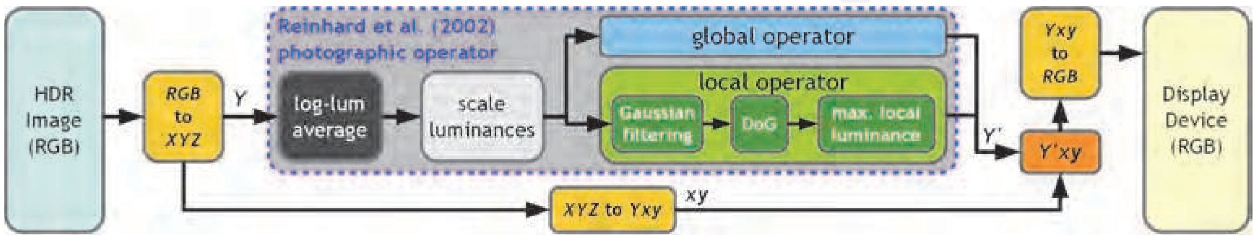


Fig. 4. Overview of the photographic tone mapping operator of Reinhard et al. (2002). The input Y is the HDR luminance ($L(x,y)$) and the output Y' is the compressed luminance ($L_d(x,y)$).

where ϕ is a sharpening factor, and defaults to $\phi = 8$. The term $L_r^s(x,y)$ corresponds to a center-surround Gaussian-blurred image, formally:

$$L_r^s(x,y) = L_r(x,y) \otimes Gaussian_s(x,y) \tag{6}$$

where the operator $\otimes$ denotes the kernel convolution operation and $Gaussian_s(x,y)$ is a Gaussian convolution profile of some scale s centered at pixel coordinates (x,y). The choice for this DoG-based model is not arbitrary and closely follows the human brightness perception model and psychophysical experiments of Blommaert & Martens (1990).

Finally, the largest isoluminant scale s_{max} is found by thresholding the Gaussian differences $V_s(x,y)$ obtained from Equation 5 against the following expression:

$$s_{max} : |V_{s_{max}}(x,y)| < \epsilon \tag{7}$$

where $\epsilon = 0.05$ proved to be a good thresholding choice through empirical experimentation, according to Reinhard et al. (2002).

Starting from the initial scaled luminance image $L_r(x,y)$ of Equation 2, subsequent blurred images $L_r^s(x,y)$ are produced according to Equation 6 with a kernel about 1.6 times larger than the previous one. As the differences $V_s(x,y)$ are computed according to Equation 5, they are thresholded against Equation 7, stopping as soon as the condition fails. The largest scale is selected if the threshold condition is never reached. In the end, the estimated local area luminance $L_r^{s_{max}}(x,y)$ is plugged back into Equation 4 for tone mapping.

In general, a total of eight scales (and therefore a total of seven DoG) is sufficient for most situations. The suggested kernel *length* (*not radius*) in pixels, at both horizontal and vertical directions, of the first eight center-surround profiles are: $1, 3, 5, 7, 11, 17, 27$ and 41.

Color information can be removed prior to luminance compression and inserted back afterwards by using the Yxy deviation of the $CIE\ XYZ$ color space [Hoffmann (2000)]. The Yxy color space is capable of separating luminance and chrominance components. This decolorization and recoloring process is also depicted in the diagram of Figure 4.

2.4 Review of Summed-Area Tables (SAT)

Although originally introduced as a texture-mapping enhancement over mip-mapping, precision constraints made Summed-Area Tables inviable for the graphics hardware to follow at that time. Since their original conception by Crow (1984), SAT were successfully employed on a wide range of tasks ranging from face and object recognition[1] [Viola & Jones (2004)], depth-of-field and glossy reflections [Hensley et al. (2005)], shadows [Díaz et al. (2010); Lauritzen (2007); Slomp et al. (2010)] and tone-mapping [Slomp & Oliveira (2008)].

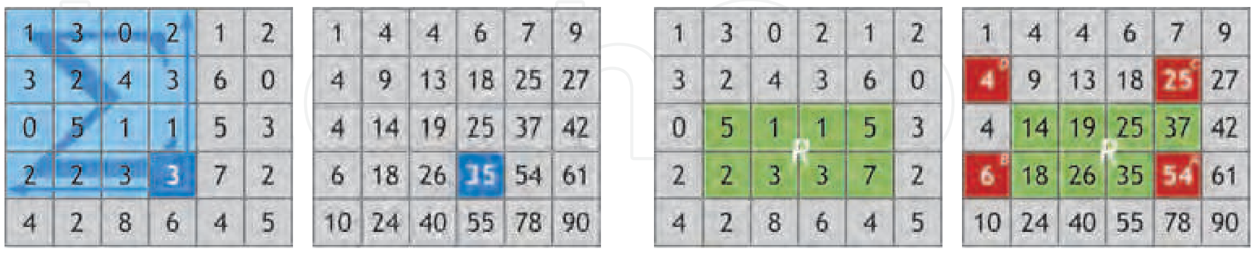
A SAT is simply a *cumulative table*, where each cell corresponds to *the sum of all elements above and to the left of it, inclusive*, in the original table, as depicted in Figure 5-ab. More formally:

$$SAT(x,y) = \sum_{i=1}^{x} \sum_{j=1}^{y} Table(i,j) \tag{8}$$

[1] Summed-Area Tables are also referred to as *integral images* in some image processing, computer vision and pattern recognition contexts.

where $1 \leq x \leq c$ and $1 \leq y \leq r$, with c and r representing the number of columns and rows of the source table, respectively. A SAT therefore has the *same* dimensions of its input table.

The usefulness of SAT comes from the fact that any *rectangular* region (axis-aligned) of the input table can be *box-filtered*[2] (or integrated) with *only four* lookups on the associated SAT (see Figure 5-cd). This gives the same *constant* filtering complexity $O(1)$ to *any* kernel size.



1	3	0	2	1	2
3	2	4	3	6	0
0	5	1	1	5	3
2	2	3	3	7	2
4	2	8	6	4	5

(a) input table

1	4	4	6	7	9
4	9	13	18	25	27
4	14	19	25	37	42
6	18	26	35	54	61
10	24	40	55	78	90

(b) Summed-Area Table

1	3	0	2	1	2
3	2	4	3	6	0
0	5	1	1	5	3
2	2	3	3	7	2
4	2	8	6	4	5

(c) filtering region

1	4	4	6	7	9
4	9	13	18	25	27
4	14	19	25	37	42
6	18	26	35	54	61
10	24	40	55	78	90

(d) SAT-based filtering

Fig. 5. A small 6×5 table (a) and its corresponding SAT (b). The highlighted blue cell on the SAT is the sum of all the highlighted blue cells in the input table (all cells up and to the left, inclusive). In order to filter the 8 elements marked in green in the input table (c), only the four red elements A, B, C and D need to be fetched from the SAT (d), a fact that holds true for arbitrary sizes. The filtering result is given by: $\frac{A-B-C+D}{area} = \frac{54-6-25+4}{4*2} = \frac{27}{8} = 3.375$.

Fetching cells outside the boundaries of the SAT, however, requires special attention: elements out of the upper or left boundary are assumed to evaluate to *zero*, while elements out of the bottom or right boundary should be redirected back to the closest element at the respective boundary (analogous to the *clamp-to-edge* mode in OpenGL). Bilinear filtering can also be used to sample the SAT at non-integer locations if necessary.

Note that the original table could be entirely discarded: the SAT alone is capable of restoring the original values of the table from which it was built from. Depending on the application, if the input table is to be used constantly along with the SAT, it is a good idea to keep the input table at hand. This is specially true if the underlying numerical storage representation is prone to introduce precision errors due to arithmetic operations (i.e., floating-point).

Summed-Area Tables can be efficiently generated in a purely sequential fashion in $O(n)$, as shown by Crow (1984). Parallel implementations on the GPU are reserved to the next Section.

3. Photographic local tone reproduction with summed-area tables

The approximation proposed by Slomp & Oliveira (2008) to the local photographic operator of Reinhard et al. (2002) suggests the replacement of the costly variable-size Gaussian-filtering by box-filtering. This means that the Equation 6 of Section 2.3 gets replaced by:

$$L_r^s(x,y) \approx L_r(x,y) \otimes Box_s(x,y) \tag{9}$$

Box-filtering can be efficiently performed through Summed-Area Tables at a fraction of the cost of Gaussian convolutions, requiring only four SAT lookups for any kernel scale s. The

[2] Higher-Order Summed-Area Tables [Heckbert (1986)] can extend plain SAT beyond box-filtering, allowing for triangular and spline-based filtering, at the expense of increased constant time overhead and numerical issues. Although compelling, a more in-depth discussion on the subject is out of the scope of this text since plain SAT are sufficient for the tone mapping technique used in this document.

input table is, in this case, L_r from Equation 2, and the corresponding SAT will be referred to as $SAT[L_r]$. Equation 9 is then rewritten as:

$$L_r^s(x,y) \approx SAT[L_r]_s(x,y) \tag{10}$$

where $SAT[L_r]_s(x,y)$ box-filters a square-shape region of L_r centered around the pixel location (x,y) at some scale s using only the contents of the $SAT[L_r]$ itself; in other words, the four pertinent cells of the SAT are fetched and filtering follows as depicted in Figure 5-cd.

The set of differences $V_s(x,y)$ from Equation 5 are performed without any structural alteration, the only change being that $L_r^s(x,y)$ and $L_r^{s+1}(x,y)$ now amount to box-filtered portions of the scaled luminance image instead of the Gaussian-filtered regions. An overview of the modified local photographic operator is shown in Figure 6.

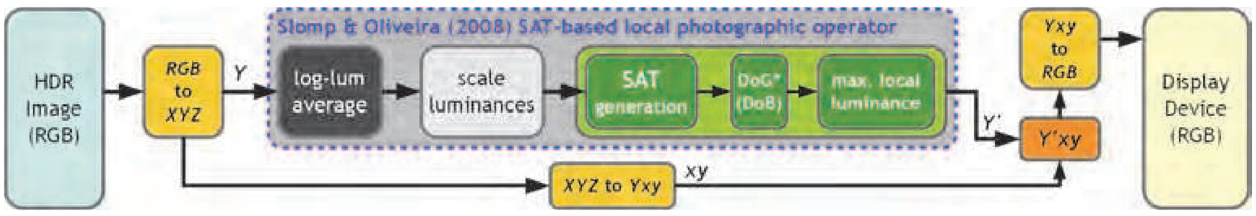


Fig. 6. Overview of the SAT-based local photographic operator of Slomp & Oliveira (2008).

Visually, results produced with this box-filtering approximation proved to be comparable to the original operator (see Figure 3-ad). Quantitative analysis using the S-CIELAB metric of Zhang & Wandell (1997) was also evaluated in the paper by Slomp & Oliveira (2008). Typically, the same originally devised filtering scales (listed at the end of Section 2.3) and threshold $\epsilon = 0.05$ from Equation 7 can be used. However, since box filters weight the contributions equally, they are more prone to noticeable halos than Gaussian filters of the same scale, which gradually reduces the weights towards the limits of the profiles. If such artifacts become apparent, the authors suggest reducing the threshold ϵ down to 0.0025.

3.1 Generating the scaled luminance image on the GPU

Prior to SAT generation, the scaled luminance image L_r should be computed. This process is fairly straight-forward and can be mapped entirely to the GPU. Initially, the input HDR image is placed into a *float-precision RGB* texture (in the case of synthesized 3D scenes, the entire scene is rendered in such texture target), as depicted in Figure 7-a.

Following that, the *RGB* texture is re-rendered into a *luminance-only* (single-channeled) float-precision texture using a full texture-aligned quad. At this stage, each *RGB* pixel is converted to the *XYZ* color space, but only the *logarithm* of the *luminance* component Y (plus some δ) is stored in this new texture. This corresponds precisely to the internal summation component of Equation 1. The summation and average can be evaluated with a full mip-map reduction of this log-luminance image: the single texel of the last mip-map level will be the key-value $\widetilde{L}$ from Equation 1, *except* for the *exponentiation*. These steps are shown in Figure 7-bc.

Finally, a final full texture pass is performed and targeted to yet another luminance-only float-precision texture, this time evaluating Equation 2. The term $L(x,y)$ is obtained by once again sampling the *RGB* texture and converting the pixels to the *XYZ* color space. The key-value $\widetilde{L}$ is accessible through the last log-luminance mip-map level, remembering to take

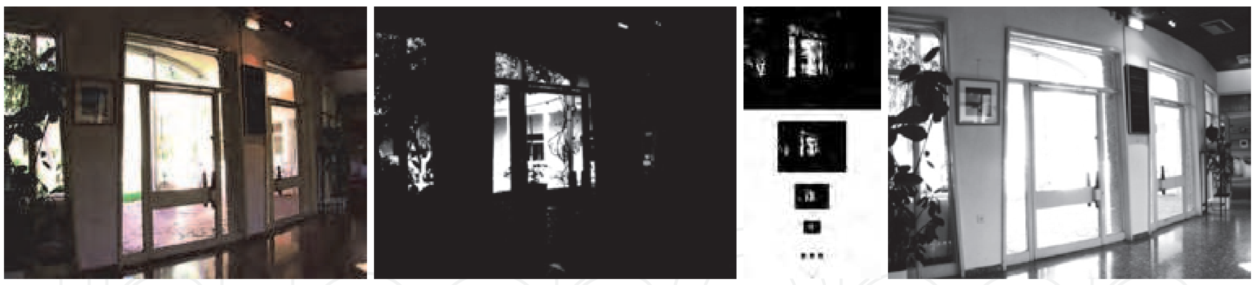

(a) input HDR RGB image (b) log-luminance (c) mipmap (d) scaled luminance L_r

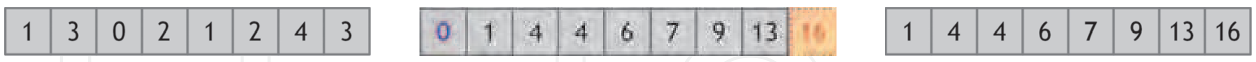
Fig. 7. Efficient generation of the scaled luminance image L_r on the GPU. The logarithm of the luminance (b) of the input image (a) is submitted to a mip-mapping reduction stage (c) and then used to produce the scaled luminance image (d).

the exponentiation once the texel is fetched. The parameter α is an application-controlled uniform shader parameter. In fact, the base log-luminance texture can be used here as the render target, since it is no longer required by the operator, thus alleviating memory requirements. Refer to Figure 7-d for a depiction of the resulting scaled luminance image.

At this point, the scaled luminance texture L_r can be either used directly on the global operator or submitted to an efficient SAT generation stage to be later used to approximate the local operator, a topic reserved for Section 3.2.

3.2 Fast Summed-Area Table generation on the GPU

Summed-Area Tables can be seen as the 2D equivalent of 1D array prefix-sums. A prefix-sum is a cumulative array, where each element is the sum of all elements to the left of it, inclusive, in the original array. There are two types[3] of prefix-sum: *prescan* and *scan*. A prescan differs from a scan by a leading zero in the array and a missing final accumulation; refer to Figure 8 for an example. As a matter of fact, prefix-sums are not limited to addition operation, but can be generalized to any other operation (neutral element is required for prescans). In the context of SAT generation, however, scans under the addition operation suffices.



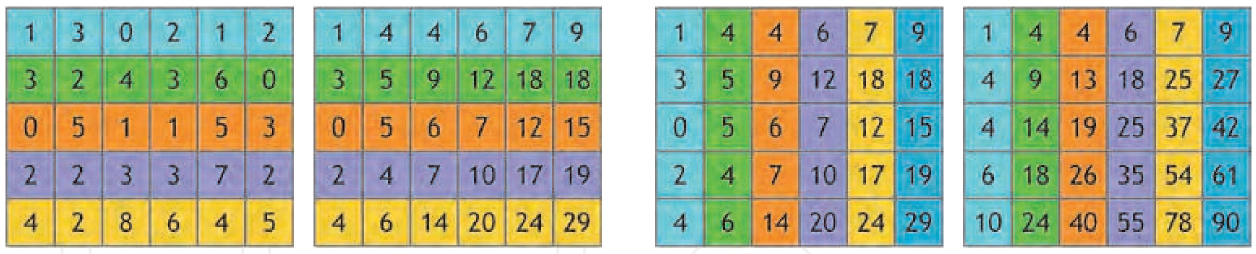
(a) input array (b) prescan (c) scan

Fig. 8. An example of a prescan (b) and a scan (c) under addition on some input array (a).

The process of generating a SAT can be break down into a two stage array scan. First, each row of the input table is independently submitted to a 1D array scan. The resulting table, to be referred to here as a *partial SAT*, is then submitted to another set of 1D scans, this time operating on each of its columns. The resulting table this time is the complete SAT itself. The process is illustrated in Figure 9.

Prefix-sum generation is a straight-forward $O(n)$ procedure using a purely sequential algorithm. Prefix-sums, however, comprise a versatile and fundamental building block for many parallel algorithms. Therefore, efficient methods that harness parallelism from

[3] A prescan may also be referred to as an *exclusive* prefix-sum, while a scan can be referred to as either an *inclusive* prefix-sum or as an *all*-prefix-sum [Blelloch (1990)].



1	3	0	2	1	2
3	2	4	3	6	0
0	5	1	1	5	3
2	2	3	3	7	2
4	2	8	6	4	5

(a) input table

1	4	4	6	7	9
3	5	9	12	18	18
0	5	6	7	12	15
2	4	7	10	17	19
4	6	14	20	24	29

(b) partial SAT

1	4	4	6	7	9
3	5	9	12	18	18
0	5	6	7	12	15
2	4	7	10	17	19
4	6	14	20	24	29

(c) partial SAT

1	4	4	6	7	9
4	9	13	18	25	27
4	14	19	25	37	42
6	18	26	35	54	61
10	24	40	55	78	90

(d) complete SAT

Fig. 9. SAT generation as a set of 1D array scans. Each row of the input table (a) is submitted to an 1D array scan, leading to a partial SAT (b). Each column of this partial SAT (c) is then submitted to another 1D array scan, resulting in the complete SAT (d).

prefix-sum generation also exist. Two of these algorithms are based on multi-pass parallel gathering patterns that map particularly well to the GPU: recursive-doubling [Dubois & Rodrigue (1977)] and balanced-trees [Blelloch (1990)].

The approach based on balanced-trees perform less arithmetic operations than recursive-doubling, but requires twice as much passes. This trade-off, however, quickly starts to pay-off for moderately larger inputs, with balanced-trees being much more work-efficient and faster than recursive-doubling on a GPU-based implementation. As far as the parallel complexity goes, the balanced-tree approach is $O(n/p + log(p))$ while recursive-doubling is $O(n/p\ log(n))$. The interested reader is redirected to Harris (2007) and Blelloch (1990) for a more in-depth complexity analysis.

Until recently the computer graphics community has curiously favored the recursive-doubling approach, despite the attractive performance gains and readily availability since long of the method based on balanced-trees. There are already plenty of resources available related to recursive-doubling implementations on the GPU [Harris (2007); Hensley et al. (2005); Lauritzen (2007); Slomp & Oliveira (2008)]. For this matter, this section will refrain in reviewing recursive-doubling and will instead focus on the balanced-tree-based approach.

3.3 Fast parallel scan generation on the GPU based on balanced-trees

Prefix-sum surveys on the literature that mention the balanced-tree approach usually describe it as a method for producing prescans only [Blelloch (1990); Harris (2007)]. Although prescans can be easily converted into scans in a few different ways (if the input is still available), this additional computation is not necessary since it is actually possible to modify the plain balanced-tree approach slightly in order to produce a scan directly. This section will focus only on this direct scan generation since it is more useful for SAT generation. Only 1D scan will be detailed, but its extension for SAT generation should be clear from Figure 9: all rows/columns can be processed simultaneously at the same pass with exactly the same shader.

There are two stages involved in the balanced-tree technique: a reduction stage (up-sweep) and an expansion stage (down-sweep). The reduction stage is straight-forward and consists on successively accumulating two nodes in $log_2(n)$ passes. Starting from the input array, each pass produces a set of partial sums on the input array; the last pass produces a single node (the root) comprising the sum of all elements of the input array. This resembles an 1D mip-map reduction, except that averages are not taken. The left side of Figure 10 depicts the process.

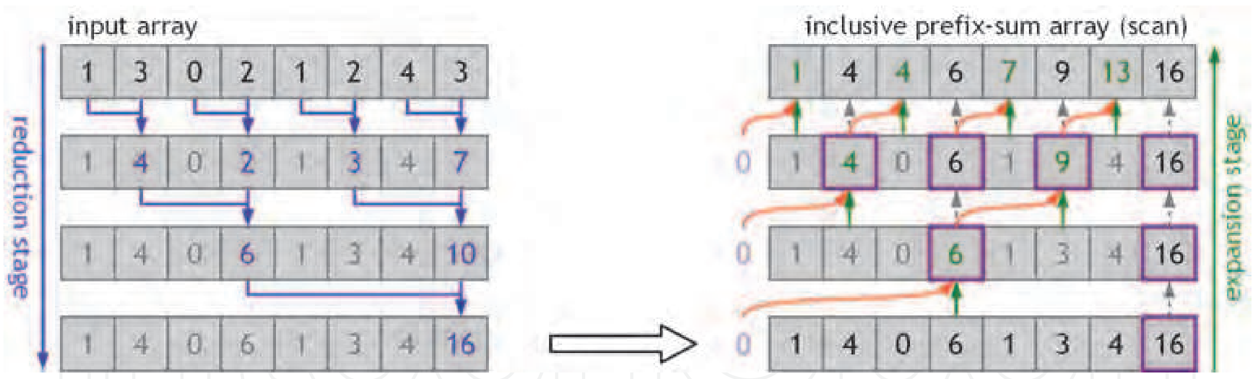


Fig. 10. An walk-through on scan generation using the balanced-tree approach.

The expansion stage is less intuitive and challenging to put into words. The reader is directed to Figure 10-right for the explanation to follow. Expansion starts from the root node of the reduction stage, referred to here as the first *generator* node (outlined in magenta). The generator node itself is its own rightmost child (dashed gray arrows). The left child is computed by adding its own value from the reduction tree (green arrows) with the value of its *uncle* generator node immediately to the left (orange arrows). If there is no such *uncle* node, a *ghost* zero-valued uncle is assumed (violet zero-valued nodes). Both children now become generator (parent) nodes (again outlined in magenta) for the next pass and the process repeats. After $log_2(n)$ passes the resulting array will be the scan of the input array.

The process illustrated in Figure 10 is memory-efficient since all computations are performed successively on the same input array without the need of any auxiliary memory. Unfortunately, a GPU-based implementation would suffer from the impossibility of performing simultaneous read and write operations on the same texture memory. The intermediate reduction and expansion stages, therefore, require additional memory to store their computations. A depiction of the suggested layout for this extra memory is presented in Figure 11.

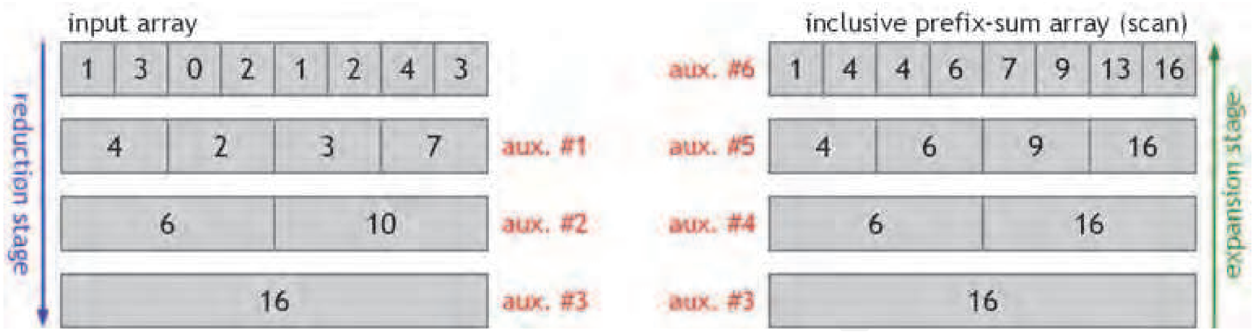


Fig. 11. Suggested layout for the auxiliary GPU memory during the scan. Simultaneous read/write from/to a same buffer never happens. In order to expand aux.#4 the expanded parent buffer aux.#3 and the reduced *sibling* buffer aux.#2 must be accessed. Similarly, expanding aux.#5 needs access to aux.#4 and aux.#1. The final expansion uses aux.#5 and the input buffer itself. The extra memory amounts to about three times the size of the input.

Even though the amount of necessary auxiliary memory with this layout is substantially large ($\approx 3n$), the memory access patterns becomes more cache-coherent than the ones from the memory-efficient version of Figure 10 since the data in each buffer is laid out together instead of sparsely distributed in a single array. This cache-coherence is also another attractive advantage that a GPU-based scan with balanced-trees has over a recursive-doubling one.

There is no need to limit the computations to two nodes per pass. Reducing and expanding a fixed number of k ($2 \leq k \leq n$) nodes per pass requires just a few modifications and can substantially improve the performance and lower the amount of extra memory. The optimal value for k is dependent on several architectural details of the GPU in question, such as memory latency and cache efficiency, and is open for experimentation. In the hardware profiles investigated it was found that $k = 4$ provided the overall best performance (see Section 5). GPU-based implementations of recursive-doubling can also benefit from a higher number of accumulations per pass as demonstrated by Hensley et al. (2005).

A reduction phase with $k > 2$ is straight-forward to implement, but the expansion phase is again more involved. Each generator node will now span k children per pass. To compute the expanded value of any child, the respective child value from the reduction tree is added together with its expanded uncle node, just as when $k = 2$. However, the values of *all* their *reduced siblings* immediately to the *left* have to be added together as well. For example, if $k = 8$ then the expanded value of the 5^{th} child will be the sum of its expanded uncle node with its own respective node in the reduction tree, plus the sum of all of its siblings to the left in the reduction tree, namely, the 1^{st}, 2^{nd}, 3^{rd} and the 4^{th}.

This process works seamlessly even when the array length n is not a multiple of k. Note that the value of the generator node is no longer propagated. One could propagate it to its k^{th} child (if any), but this is less systematic since special cases need to be accounted in the shader.

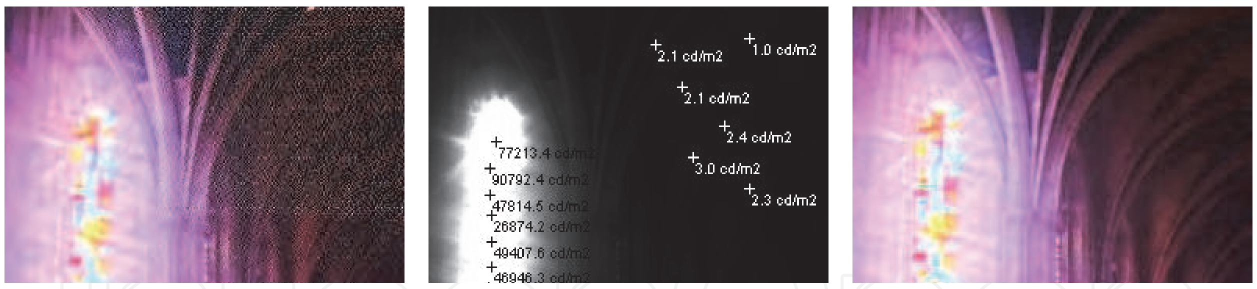
Equipped with such an algorithm, a GeForce GTX 280 is capable of generating a 2048x2048 SAT in about 2*ms*. In comparison, recursive-doubling would take nearly 6*ms*. In general, the overall speedup is of roughly 3x. Performance results are summarized in Section 5.

3.4 A Note on precision issues with Summed-Area Tables

Luminances are positive quantities, which makes the SAT of luminance images grow monotonically. The values in the SAT can quickly reach overflow limits or run out of fractional precision due to ever increasing accumulations. Depending on the dynamic range of the involved scene and the dimensions of the image, these critical situations may be hastily reached.

When such situations happen, high-frequency noise artifacts appear in the resulting tone mapped image, thus compromising the quality as shown in Figure 12-a. One way to mitigate this problem is to subtract the average value of the input table from the table itself prior to SAT generation [Hensley et al. (2005)]. This simple procedure has two main implications: first, the SAT allots an additional bit of precision, the signal bit, due to introduction of negative quantities, and second, the SAT is no longer monotonic and thus the range of the values within it should have much lower magnitude.

After filtering with this non-monotonic SAT, keep in mind that the average input table value should be added back. The average value can be computed using the same mip-mapping reduction strategy described earlier for the log-luminance average (see Section 3.1). The performance overhead incurred is small and well worth for the extra robustness. There is no need for additional memory for this average-subtracted scaled luminance image: the average can be subtracted onto the original scaled luminance texture using subtractive color blending.



(a) plain monotonic SAT (b) luminance magnitude range (c) non-monotonic SAT

Fig. 12. Summed-Area Tables of luminance images are inherently monotonic and prone to unpleasing noise artifacts (a) if the quantities involved have a wide dynamic range (b). Making the SAT non-monotonic by first subtracting the average luminance mitigates such artifacts.

4. Mesopic vision simulation

In order to reproduce Purkinje's blue-shift effect, it is necessary to have some quantitative estimation on how individual color responses tend to change under mesopic vision conditions. Following that, it is important to be able of reproducing such changes in some HVS-compatible and perceptually-uniform fashion. Therefore, before the proposed mesopic vision operators are properly introduced in Sections 4.3- 4.5, a discussion on psychophysical subjective luminosity perception and opponent-color spaces will be presented in Sections 4.1 and 4.2.

4.1 Equivalent lightness curve

Ikeda & Ashizawa (1991) have performed a number of subjective luminosity perception experiments under various lighting conditions. In these experiments, subjects were exposed to a special isoluminant room where different glossy colored cards[4] were presented to them. Once adapted to the different levels of luminosity, the subjects were asked to match the brightness of the colored cards against particular shades of gray distributed in a scale[4]. From the data analysis, several curves of *equivalent lightness* were plotted, depicting how the experienced responses of different colors varied with respect to the isoluminant room conditions. The results can be compiled into a single equivalent lightness response chart, as shown in Figure 13.

As can be observed in the curves, as the overall luminosity decreases, red intensities produce much lower lightness responses than blue intensities, which only slightly varies. This behavior implicitly adheres to the expected characteristics of the Purkinje's effect, that is, the tendency of the HVS to favor blue tonalities at low lighting conditions. Since blue responses are barely affected, finding adequate lowering factors mainly for red responses should be a sufficient approximation, which is the key idea behind the proposed mesopic filters.

The red response curve of Figure 13 can be approximated by the following expression:

$$E(I) = \frac{70}{1 + (10/I)^{0.383}} + 22 \tag{11}$$

[4] Standard, highly calibrated color/gray chips and scale produced by Japan Color Research Institute.

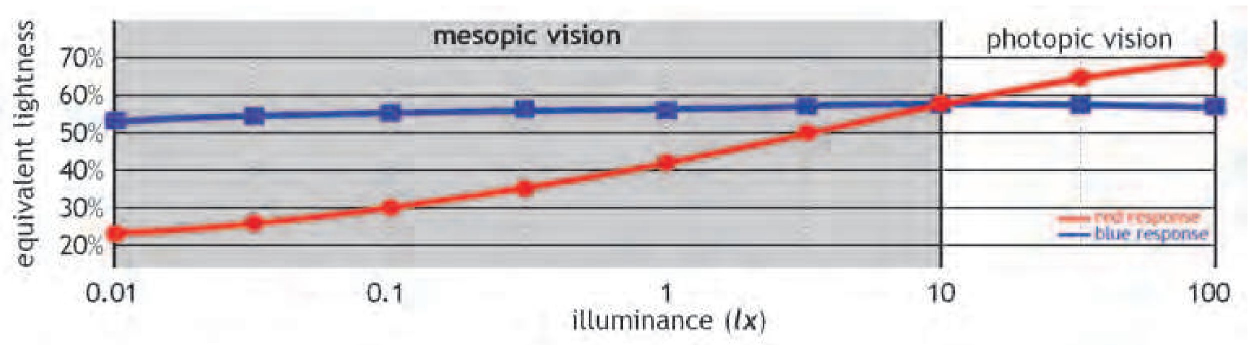


Fig. 13. Equivalent lightness curve for red and blue according to the experiments of Ikeda & Ashizawa (1991). These curves summarize the experienced relative brightness from several colored cards against gray-scale patterns in different isoluminant environment conditions.

The range of interest is the mesopic vision range, that is to say, between 0.01 lx and 10 lx. If the equivalent lightness of some overall luminosity $E(\lambda)$ is normalized against the equivalent lightness of the triggering luminance of the mesopic vision range $E(10)$, then the result will yield a coefficient ρ that indicates, in relative terms, how much the response coming from red intensities at such lighting conditions lowers with respect to the starting range $E(10)$:

$$\rho(\lambda) = \frac{E(\lambda)}{E(10)} \tag{12}$$

In other words, if $\rho(\lambda) \geq 1$, the overall luminosity offers photopic conditions and chrominances *do not* need to be altered. However, if $\rho(\lambda) < 1$ then the overall luminosity lies in the mesopic vision range, and $\rho(\lambda)$ gracefully provides a normalized relative measurement of how much red components should be scaled down in order to reproduce the expected experienced response. These two conditions can be handled uniformly by *clamping* $\rho(\lambda)$ to $[0, 1]$.

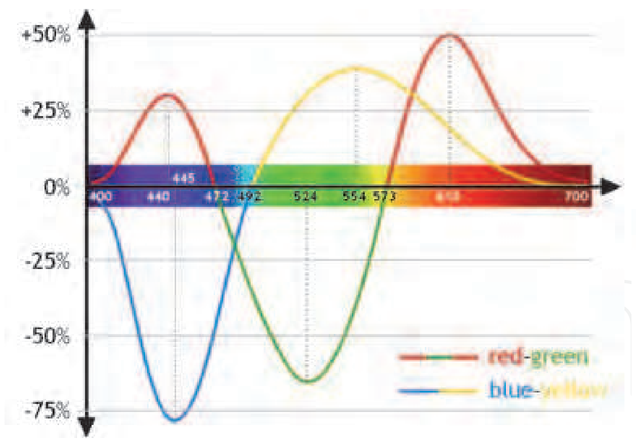
4.2 Opponent-color theory and the L*a*b* color space

Recent physiological experiments by Cao et al. (2008) have shown that mesopic color-shifts happen in the opponent-color systems of the Human Visual System, and that these shifts change linearly with the input stimuli. The initial hypothesis behind opponent-color theory in the HVS was proffered by German physiologist Karl Ewald Konstantin Hering and later validated by Hurvich & Jameson (1955). The key concept is that particular pairs of colors tend to nullify each other responses in the HVS and thus can not to be noticed simultaneously.

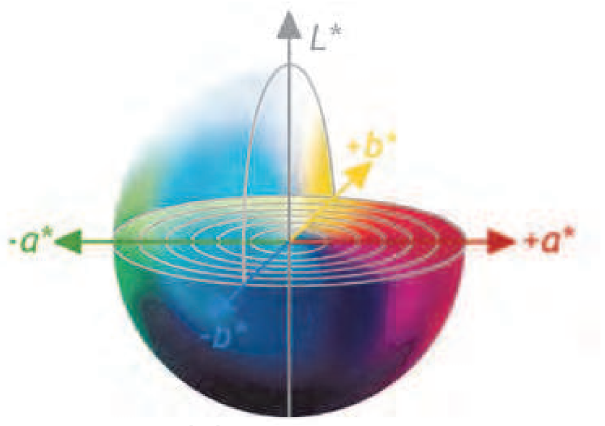
Color perception in the HVS is guided by the joint activity of two independent opponent systems: red versus green and yellow versus blue. A plot of these opponent response curves is shown in Figure 14-a, based on the experiments of Hurvich & Jameson (1955). Interestingly, luminance perception is also controlled by another opponent system: white versus black.

Physiologically speaking, these systems are dictated by opponent neurons which appropriately produce a chain of excitatory and inhibitory responses between the two components of each individual system. Numerically this models to positive feedbacks at some wavelengths being interfered concurrently by negative impulses at their counterparts.

Several color spaces were established based upon opponent-color schemes. The $CIE\ L^*a^*b^*$ (Figure 14-b) is remarkably one of the best known and widely used of such color spaces since



(a) HVS opponent-colors: wavelength × response



(b) L*a*b* opponent-color space

Fig. 14. Opponent chromatic response curves of the HVS (a), and the L*a*b* color space (b).

it gauges color distances in a compatible perceptually-uniform (linear) fashion with respect to the Human Visual System. The L^* component represents a *relative* measurement of luminance (thus can not be expressed in cd/m^2), while a^* relates to a red-green opponent system and b^* to a blue-yellow opponent system. For simplicity, throughout the remaining of this Section the $CIE\ L^*a^*b^*$ color space will be shortly referred to as Lab.

The proposed mesopic vision filters to be introduced in the following Sections make use of the Lab color space. The choice for an opponent-color system to apply the color-shifts is supported by the physiological observations of Cao et al. (2008); the Lab space itself was selected because it offers the desired perceptually linear characteristics. Even though external light stimuli is directly *received* by red-green-blue cone-shaped sensors in the retina, the experienced *perceived* color comes from the activity of opponent-color systems wired to these primaries.

4.3 Overview of the mesopic vision reproduction operator

The proposed mesopic vision filter for digital images is derived directly from the lightness response curve specialized for mesopic vision of Equation 11 and from the perceptually-uniform Lab color space. Given a pixel represented in some color space (i.e., RGB), the process begin by promoting the pixel to the Lab color space.

A Lab pixel holding a *positive* quantity at its a component is actually holding some red intensity (negative a means green, see Figure 14). Now, if the overall luminosity condition suggests mesopic vision (that is, $\rho(\lambda) < 1$) then this red (positive-a) component should modulated by the normalized coefficient $\rho(\lambda)$ obtained from Equation 12, as below:

$$a' = a\,\rho(\lambda) \tag{13}$$

and this modified quantity a' replaces the original a in the Lab triplet, yielding to $La'b$. This can then be converted back to the initial color space (i.e., RGB) for presentation purposes.

In order to integrate these chrominance adjustments with HDR luminance compression, another color space must be used as an intermediate, preferentially one that can decouple luminance from chrominance. The Lab space itself is one of such spaces, but since the component L comprises only a relative measurement of luminance, this may cause incompatibilities with tone mapping operators that require proportionally equivalent absolute quantities. The Yxy deviation of the $CIE\ XYZ$ color space is up to this task [Hoffmann (2000)].

The general algorithm for the filter can be summarized in the following steps:

1. obtain a measurement of the overall luminosity – λ – (Sections 4.4 and 4.5)
2. compute the red response coefficient for this luminosity – $\rho(\lambda)$ – (Equation 12)
3. transform the original HDR pixels to the Lab color space
4. perform the blue-shift by altering the red (positive-a) component – a' – (Equation 13)
5. transform the modified $La'b$ pixels to the Yxy color space
6. compress the HDR pixel luminance Y using some TMO, yielding to Y'
7. replace the HDR pixel luminance Y by the compressed pixel luminance Y'
8. transform the modified Yxy pixels to the color space of the display device

The proposed mesopic vision filter offers two variants to determine the average isoluminant for the step 1 above: one that is spatially-uniform (per-scene, global) and another that is spatially-varying (per-pixel, local). They are introduced in Sections 4.4 and 4.5, respectively.

4.4 Spatially-uniform mesopic vision reproduction operator

One way to determine the overall luminosity of a scene is through the log-average luminance, as reviewed in Section 2.3. This log-average is a suitable candidate, but it was observed that it has a tendency of placing relatively bright images very low into the mesopic range scale. Simple arithmetic average of the luminances was found to produce more plausible indications for the global mesopic scale, which is also more compatible with the equivalent lightness curve of Section 4.1 that is plotted based on absolute luminosity quantities.

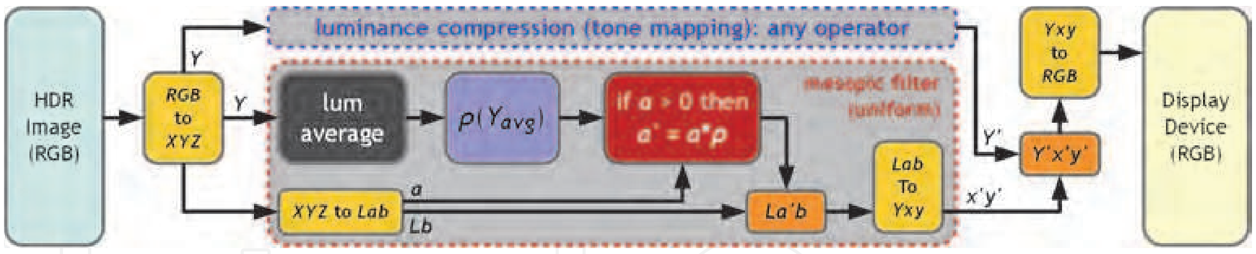
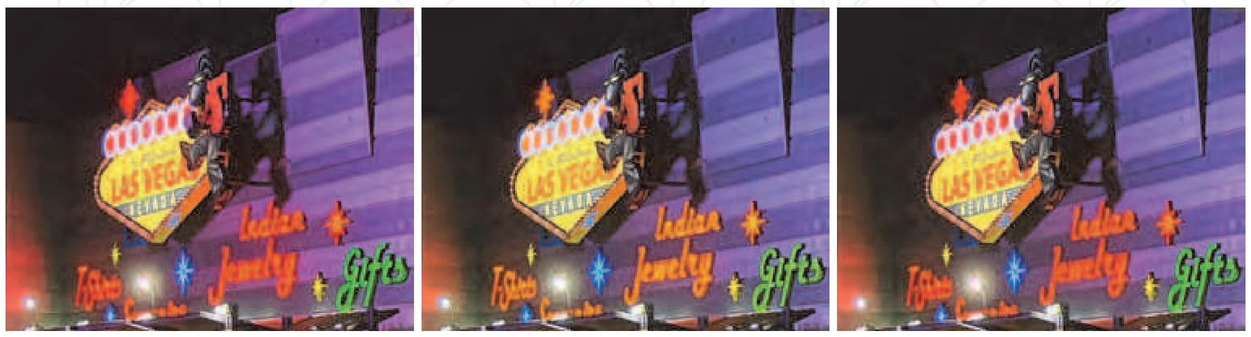


Fig. 15. Overview of the spatially-uniform mesopic vision reproduction filter.

Computing the arithmetic average of the luminances is straight-forward on GPU by using the same mip-map reduction strategy described in Section 3.1. It is possible to produce this average in advance along with the log-average by reformatting the base log-luminance texture with an additional channel to hold the absolute luminance (`GL_LUMINANCE_ALPHA32F`).

Once evaluated, the average serves as a measurement for the overall luminosity of the entire scene and the algorithm follows as listed in Section 4.3, applying the *same* response coefficient $\rho(Y_{avg})$ to all pixels, uniformly. The process is illustrated in Figure 15. An example of this spatially-uniform mesopic filter is shown in Figure 16-b. Refer to Section 5 for more examples.

The advantage of such uniform filter is that it is independent of luminance compression and hence can be integrated with any existing TMO. The disadvantage becomes clear when strong red-hued light sources are present in the scene, as is the case of Figure 16. When the overall luminosity suggests mesopic vision, the intensity coming from such light sources will be inadvertently suppressed, regardless of their own local intensity; even worse, the higher the intensity, the larger the shift will be. Red light traffic semaphores, neon lights and rear car lights, for example, still hold perceptually strong red intensities which are not noticed as yellow by an external observer, even at the dimmest surrounding lighting conditions. An even more extreme example would be a digital alarm clock equipped with red LEDs: even in complete darkness the LEDs are still perceived as red. This leads to the design of a variant filter that is capable of reproducing mesopic vision in a local, spatially-varying fashion.



(a) without mesopic simulation (b) spatially-uniform mesopic (c) spatially-varying mesopic

Fig. 16. Comparison between an image without mesopic simulation (a), with global mesopic filter (b) and per-pixel mesopic filter (c). All red intensities shift toward orange/yellow in (b), while only those not sufficiently bright enough change in (c) like the light reflex in the leftmost wall. Also note that purple tones shifted towards a more blueish hue in the mesopic images.

4.5 Spatially-varying mesopic vision reproduction operator

In order to counter-act the effects described at the end of Section 4.4, per-pixel local area luminance must be inspected. The good news is that such local measurement is already available from the local variant of the photographic tone mapping operator ($L_r^{s_{max}}(x,y)$). The bad news is that this quantity is based on the relative *scaled* luminance image L_r of Equation 2, and thus incompatible with the *absolute* scale of the equivalent lightness curve from Section 4.1.

Fortunately, the scale can be nullified with the inverse function of the scaled luminance L_r:

$$L_r^{-1}(x,y) = L_r(x,y)\frac{\widetilde{L}}{\alpha} = L(x,y) \tag{14}$$

The expression above can be generalized to filtered versions of $L_r(x,y)$ at any scale s:

$$L_r^s(x,y)\frac{\widetilde{L}}{\alpha} = L^s(x,y) \tag{15}$$

Hence, plugging the *scaled* local area luminance $L_r^{s_{max}}(x,y)$ in the expression above yields to $L^{s_{max}}(x,y)$, which is the *absolute* local area luminance. This quantity is now compatible with the equivalent lightness curve, and it is now possible to determine individual response coefficients $\rho(L^{s_{max}}(x,y))$ for each pixel (x,y), thus enabling localized mesopic adjustments.

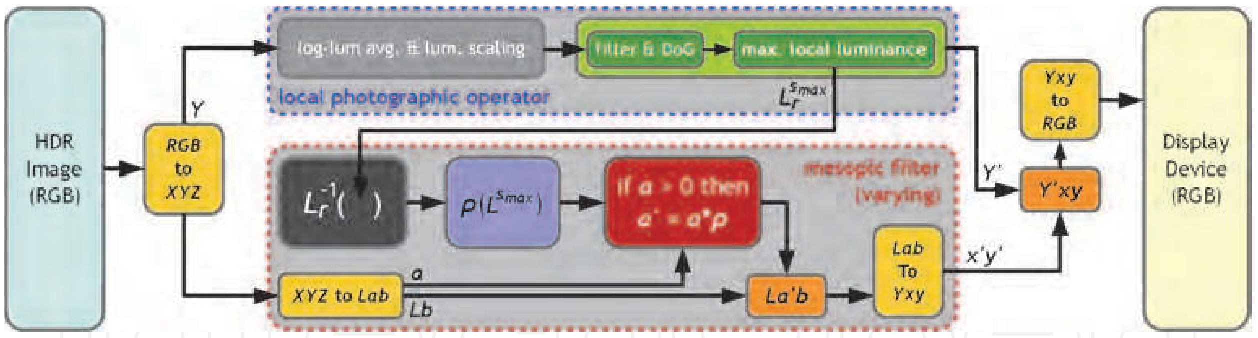


Fig. 17. Overview of the spatially-varying mesopic vision reproduction filter.

An overview of this spatially-varying mesopic filter is depicted in Figure 17. The application of the filter on a real image in shown in Figure 16-c; more examples are provided in Section 5.

The spatially-varying mesopic filter is structurally simpler than the spatially-uniform one since no additional data need to be assembled. The filter is, however, strongly attached to the framework provided by the photographic operator. Although no other local TMO was explored, most local operators perform estimations of local pixel averages and hence should be able to offer such information and feed it through the filter pipeline of Figure 17.

An advantage of using the local photographic operator over other local operators for the localized mesopic reproduction is the fact that per-pixel local area luminances are searched through an HVS-based brightness perception model [Blommaert & Martens (1990)]. This means that the chromatic adjustments follow some perceptual guidelines, while other operators may not rely at all on HVS features and thus become less suitable for a proper mesopic reproduction.

5. Results

This Section summarizes the performance achievements of balanced-trees over recursive-doubling for GPU-based SAT generation, as well as the performance of the mesopic filters. Additional examples of the mesopic filters are also available in Figures 23 and 24.

The system configurations and hardware profiles investigated are listed below:

1. Windows 7 Enterprise 32bit SP1 running on an Intel(R) Core(TM)2 Quad CPU Q9499 2.66GHz with 4GB RAM equipped with a NVIDIA GeForce GTX 280 with 1GB VRAM (240 shader cores at 1107MHz, memory at 1296MHz, WHQL Driver 280.26)
2. Windows 7 Enterprise 32bit SP1 running on an Intel(R) Core(TM)2 Quad CPU Q8200 2.33GHz with 4GB RAM equipped with a NVIDIA GeForce 9800 GT with 512MB VRAM (112 shader cores at 1500MHz, memory at 900MHz, WHQL Driver 280.26)
3. Windows XP Professional x64 Edition SP2 running on an Intel(R) Xeon(R) CPU W3520 2.67GHz with 8GB RAM equipped with an ATI FirePro 3D V3700 with 256MB VRAM (40 shader cores at 800MHz, memory at 950MHz, WHQL Catalyst Driver v8.85.7.1)

The main program was implemented in C++, compiled and linked with Visual C++ Professional 2010. The graphics API of choice was OpenGL and all shaders were implemented in conformance to the feature-set of the OpenGL Shading Language (GLSL) version 1.20. A

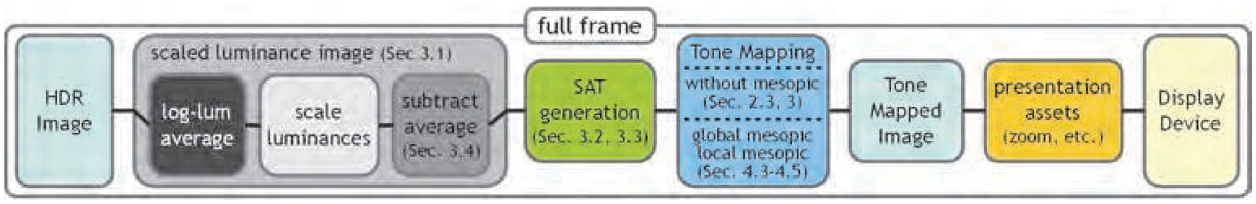

full frame
HDR Image
scaled luminance image (Sec 3.1)
log-lum average
scale luminances
subtract average (Sec. 3.4)
SAT generation (Sec. 3.2, 3.3)
Tone Mapping
without mesopic (Sec. 2.3, 3)
global mesopic
local mesopic
(Sec. 4.3-4.5)
Tone Mapped Image
presentation assets (zoom, etc.)
Display Device

Fig. 18. Overview of all the stages implemented in the program for a complete frame display.

diagram depicting all the stages of the program for the display of a complete frame on the screen is presented in Figure 18.

All performance times in this Section are given in *milliseconds*. Full-frame times were captured with performance counters from the Win32 API and double-checked with the free version of Fraps 3.4.6. Intra-frame performance was profiled using OpenGL Timer Query Objects (`GL_ARB_timer_query`). Performance results were recorded through multiple executions of the program from which outliers were removed and the average was taken.

Summed-Area Table generation times for typical screen/image resolutions using recursive-doubling and balanced-trees are presented in Figures 19-a and 19-b, respectively. The speed-up achieved with the balanced-tree approach is shown in Figure 20.

Performance times for the tone mapping stage alone with and without the mesopic simulation are shown in Figure 21-a. The overhead introduced to the operator by the spatially-varying mesopic filter is shown in Figure 22. Complete frame times are shown in Figure 21-b, accounting for all the stages depicted in Figure 18 with the spatially-varying mesopic filter activated.

From Figure 20 it can be seen that SAT generation with the balanced-tree outperforms recursive-doubling by a factor of 2.5x≈3x (or 4x on the ATI FirePro 3D V3700) as the image size increases. From Figure 22 one can see that the overhead introduced by the spatially-varying mesopic filter tends to amount to about 16% up to 19% of the original execution time of the operator without the filter (8% for the ATI FirePro 3D V3700). Most of this overhead is coming from the non-linear Yxy to *Lab* color conversions (and vice-versa) inside the tone mapping shader. Note that this overhead is being measured relative to the tone mapping stage alone; putting it on a full-frame scale the overhead is most likely negligible.

6. Limitations and future work

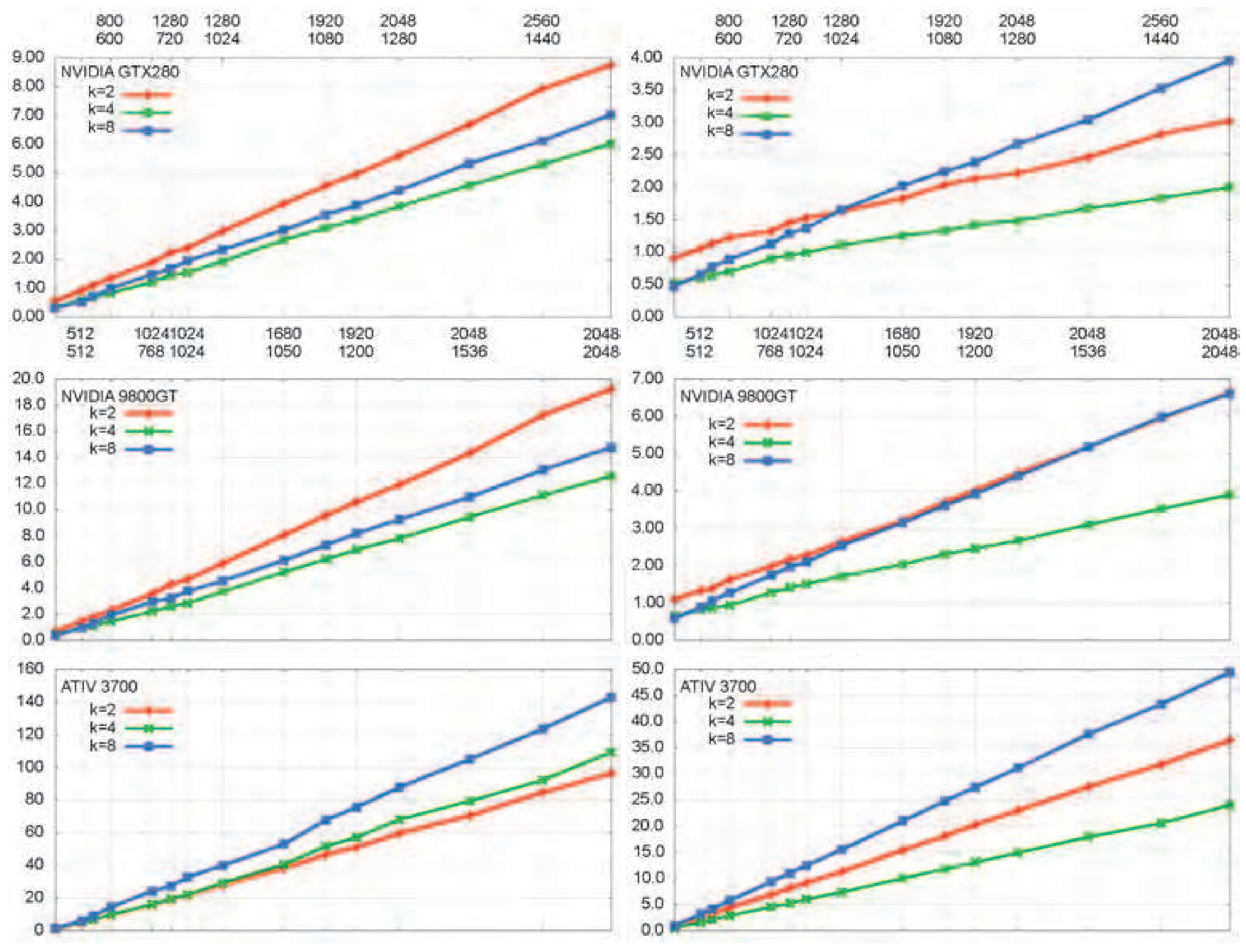
The main drawback with SAT generation on the GPU is the fact that it requires a considerable amount of intermediate memory. This is due to the simultaneous read-write texture (global) memory restrictions and the lack of adequate inter-fragment synchronization and communication directives of GPU architectures. As the expected clash between GPU and multi-core CPU architectures comes to a close, such memory access constraints tend to disappear. Current development on general-purpose GPU computing technologies such as CUDA, OpenCL, DirectCompute and C++Amp already started to address these limitations and are paving the road for exciting new prospects. The interoperability overhead between such technologies and regular graphics API is also expected to diminish with future advances.

It must be noticed that the introduced mesopic filters decrease responses from red intensities only, as suggested by the curves of equivalent lightness that were studied. A future opportunity is to investigate how the response coming from of other colors (green and yellow, namely, for the assumed opponent-system) behave according to different levels of luminosity, leading to a much more robust and believable mesopic vision experience.

Another limitation of the introduced mesopic filters is that color-shifts happen instantly. It is a known fact that chrominance changes in the HVS do not occur abruptly, but instead stabilize gradually on due time, just like with luminosity adaptation. Temporal luminance adaptation is a topic already studied extensively, but little is known about temporal chromatic adaptation, which still remains as a fascinating open field for further research.

The local averages used by the spatially-varying mesopic filter come directly from the brightness perception model used implicitly by the local photographic operator. Such local averages are convenient since they are already part of the luminance compression framework that was utilized and are efficient to be computed. The estimation of such local averages is based on psychophysical brightness perception experiments and the obtained results look plausible; however, they may not be the most suitable candidate for local averages under mesopic conditions. Additional physiological evidence must be researched in order to validate the accuracy of the employed brightness perception model for mesopic vision.

An implicit assumption made over the length of this document was that the source HDR imagery was properly calibrated. In other words, the HDR images were expected to be



a) SAT generation time for recursive-doubling. b) SAT generation time for balanced-trees.

Fig. 19. SAT generation time

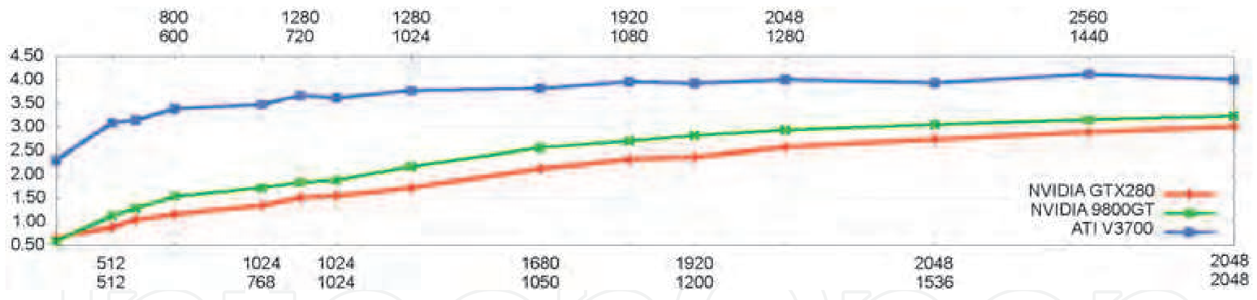
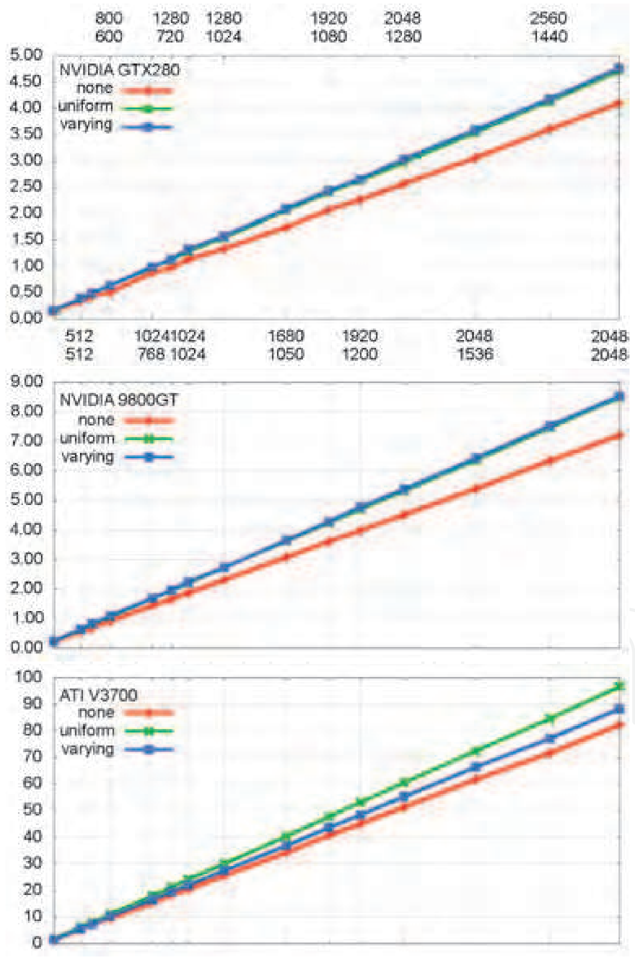
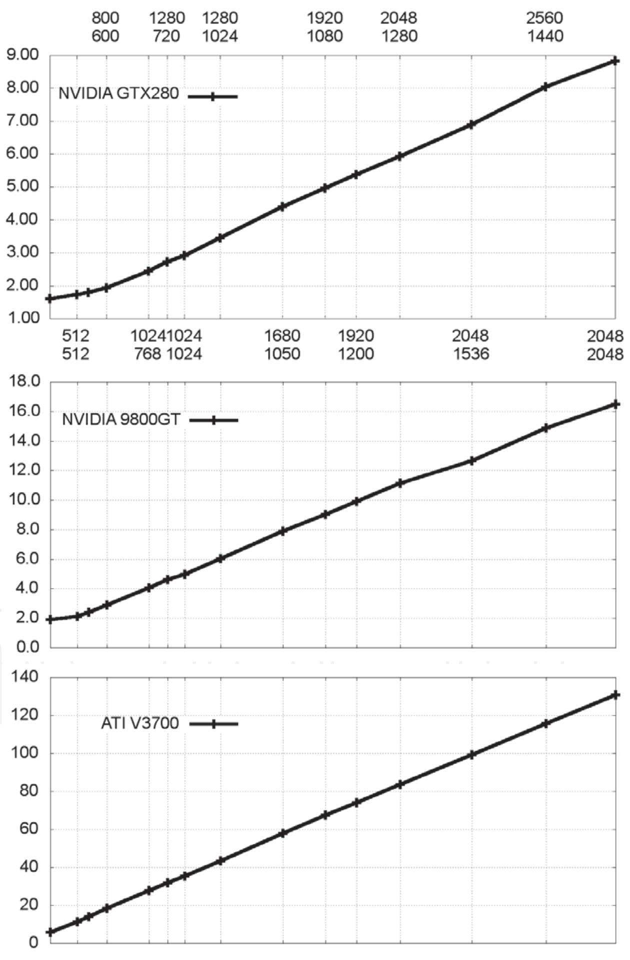


Fig. 20. Relative speed-up between balanced-trees and recursive-doubling for SAT generation based on the best (fastest *k*) times recorded for each algorithm for each image size, according to the performance results of Figures 19-a and 19-b.

holding physically accurate quantities. This may not be always the case, but HDR images should supposedly contain proportionally equivalent quantities at least, differing to real quantities only by some constant, uniform scaling factor. In the context of the proposed mesopic reproduction filters, any HDR image that infringes this constraint is considered an ill-formed image.



a) Performance of the tone mapping stage alone, with and without the mesopic filters.



b) Full frame times accounting for all the stages presented in Figure 18.

Fig. 21

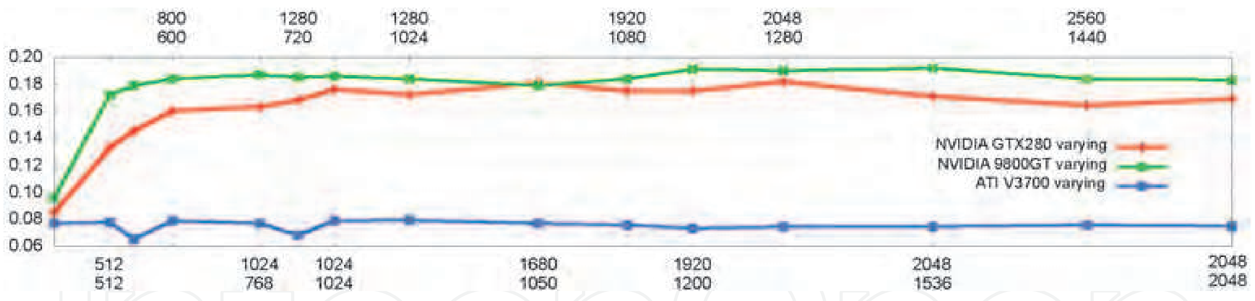


Fig. 22. Overhead of the spatially-varying mesopic filter to the tone mapping stage.

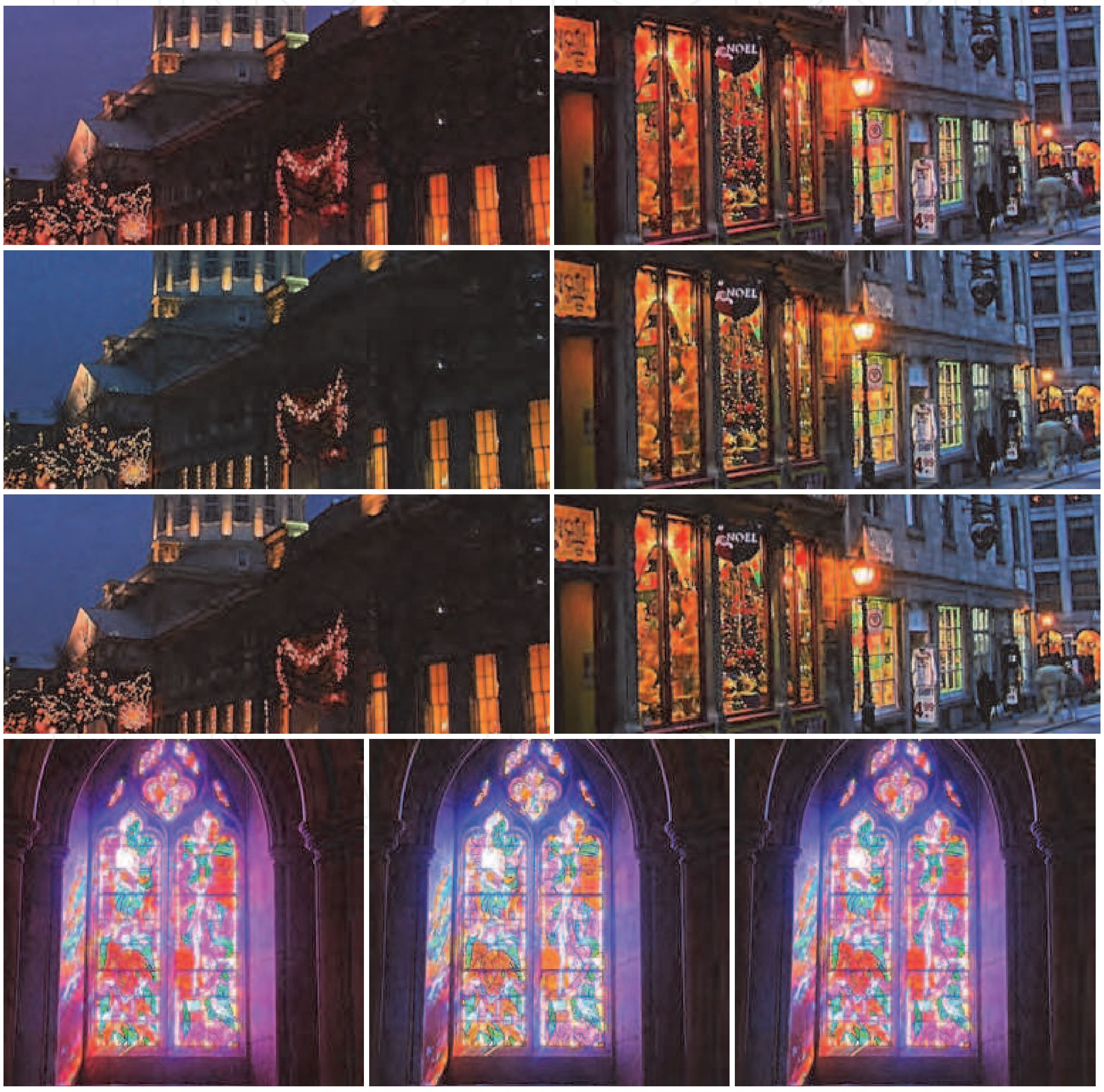


Fig. 23. Examples of the proposed mesopic filters. The first row of the top images is without the filter, the second row is the global filter and the third row is the local filter; similarly for the images at the bottom, but arranged in columns instead of in rows.

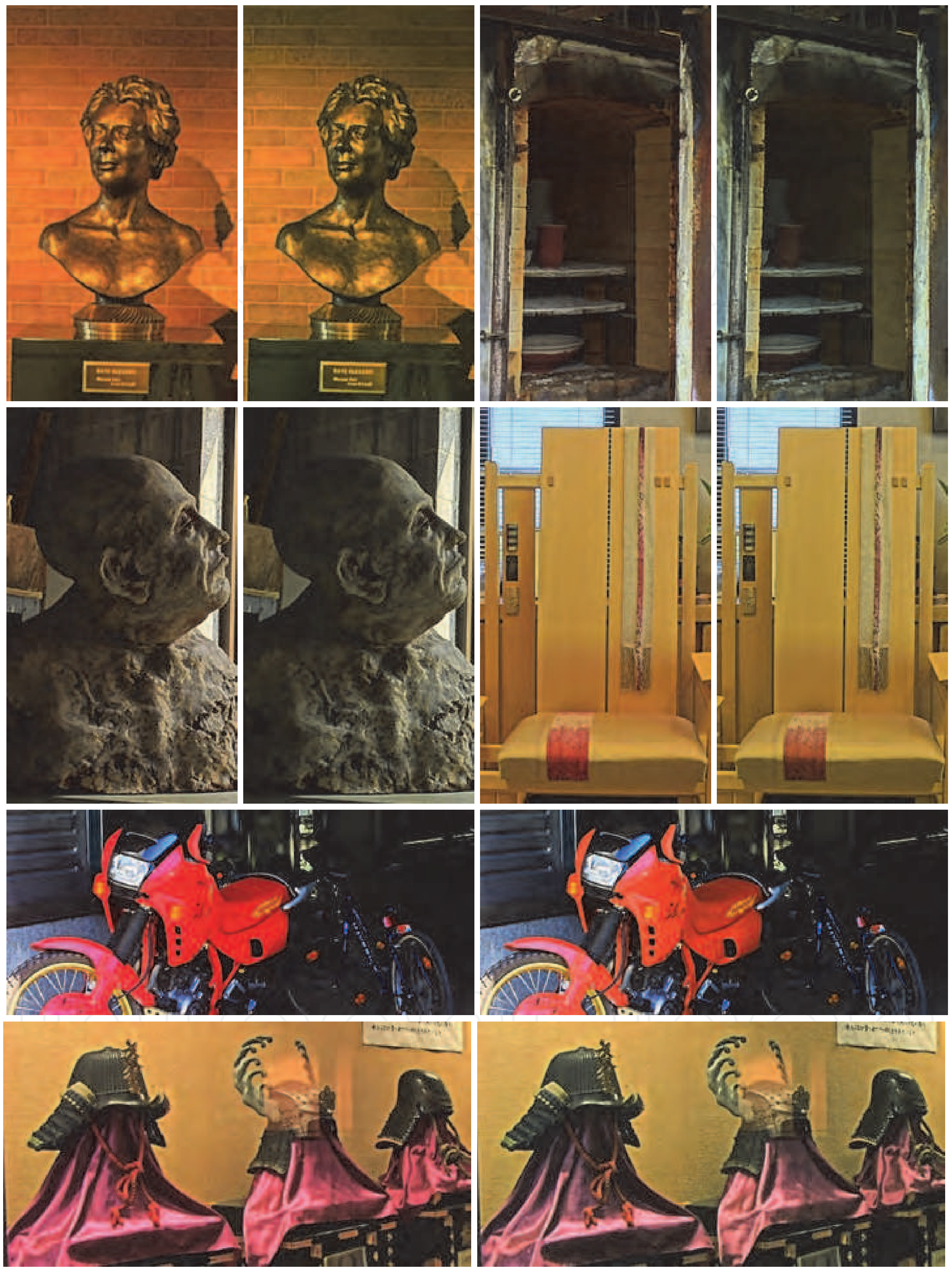


Fig. 24. Examples of the proposed mesopic filters. The leftmost image of each image set is without the filter; the rightmost is the varying filter. For these images, either the global or the local mesopic filter produces very similar results because there are no strong light intensities in the images, which makes the local averages to be somewhat close to the global average.

Finally, adapting other local operators to function along with the proposed spatially-varying mesopic filter is a possible direction for future work. However, not all local operators are guided by perceptual characteristics of the Human Visual System and thus may be less suitable for a proper mesopic vision reproduction experience.

7. Conclusion

This chapter described a faster method based on balanced-trees to accelerate the Summed-Area Table generation stage for local photographic tone reproduction. The balanced-tree approach not only has a lower computational complexity than the more widely known recursive-doubling technique, but has also memory access patterns that are more cache-friendly to common GPU architectures. Substantial improvements of about 3x in the SAT generation time were achieved on the inspected hardware profiles and image resolutions.

A balanced-tree implementation on GPU is more involved than a recursive-doubling one, but can still be accomplished with any GPU architecture that supports at least the Shader Model 2.0 feature set. Even though parallel algorithms for prefix-sum generation based on balanced-trees were readily available since long, the computer graphics community has just recently started to acknowledge and appreciate its advantages, and so far only a handful of resources exist on the subject in the graphics literature.

A novel, general and efficient approach to reproduce the Purkinje effect under mesopic vision conditions was also introduced. The method was designed to work with popular, widely-spread HDR imagery formats. The blue-shift is simulated by suppressing the responses coming from red intensities according to either global or local luminosity conditions. These red responses are smoothed based on an equivalent-lightness curve recovered through psychophysical experiments performed on real subjects. The smoothing is applied in an HVS-compatible, perceptually-linear fashion through the $CIE\ L^*a^*b^*$ opponent-color space, and this linearity conforms to recent physiological evidence.

The spatially-uniform mesopic filter is completely decoupled from luminance compression, thus allegedly suitable for any existing tone mapping operator. The more interesting spatially-varying filter exploits the foundations and perceptually-based characteristics of the local photographic operator to perform mesopic adjustments in a per-pixel basis. The filters are simple to be implemented entirely on the GPU and the overhead introduced is negligible and should not hurt the performance of existing real-time applications.

8. Acknowledgments

We would like to sincerely thank the Munsell Color Science Laboratory, Gregory Ward, Mark Fairchild (Mark Fairchild's HDR Photographic Survey), Erik Reinhard, Paul Debevec, Jack Tumblin, Yuanzhen Li, Dani Lischinski, Laurence Meylan and others for generously making their HDR imagery data-set freely available for researchers. We also thank the initiatives of GLEW (OpenGL Extension Wrangler Library), DevIL (Developer's Image Library, former OpenIL), Scilab, LibreOffice and gnuplot for the excellence of their open-source contributions, as well as to Microsoft for providing Visual Studio Professional and Windows 7 Enterprise free of charge for students through their academic alliance program, and to Fraps for the freeware version of their real-time video capture and benchmarking tool. This research was supported by Japan's Ministry of Education, Culture, Sports, Science and

Technology (Monbukagakusho/MEXT) and Japan Society for the Promotion of Science (JSPS) – KAKENHI No. 2310771.

9. References

Adams, A. (1983). *The Print*, Little, Brown and Company.

Blelloch, G. E. (1990). Prefix sums and their applications, *Technical report*, Synthesis of Parallel Algorithms.

Blommaert, F. J. & Martens, J.-B. (1990). An object-oriented model for brightness perception, *Spatial Vision* 5(1): 15–41.

Cao, D., Pokorny, J., Smith, V. C. & Zele, A. (2008). Rod contributions to color perception : linear with rod contrast, *Vision Research* 48(26): 2586–2592.

Crow, F. C. (1984). Summed-area tables for texture mapping, *Proceedings of SIGGRAPH 84* 18: 207–212.

Díaz, J., Vázquez, P.-P., Navazo, I. & Duguet, F. (2010). Real-time ambient occlusion and halos with summed area tables, *Computers & Graphics* 34(4): 337 – 350.

Dubois, P. & Rodrigue, G. (1977). An analysis of the recursive doubling algorithm, *High Speed Computer and Algorithm Organization* pp. 299–307.

Durand, F. & Dorsey, J. (2000). Interactive tone mapping, *Proceedings of the Eurographics Workshop on Rendering Techniques 2000*, Springer-Verlag, London, UK, pp. 219–230.

Ferwerda, J. A., Pattanaik, S. N., Shirley, P. & Greenberg, D. P. (1996). A model of visual adaptation for realistic image synthesis, *Proceedings of SIGGRAPH 96*, ACM, New York, NY, USA, pp. 249–258.

Goodnight, N., Wang, R., Woolley, C. & Humphreys, G. (2003). Interactive time-dependent tone mapping using programmable graphics hardware, *Proceedings of the 14th Eurographics workshop on Rendering*, EGRW '03, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, pp. 26–37.

Harris, M. (2007). *Parallel Prefix Sum (Scan) with CUDA*, NVIDIA Corporation.
URL: `http://developer.download.nvidia.com/compute/cuda/1_1/Website/projects/scan/doc/scan.pdf`

Heckbert, P. S. (1986). Filtering by repeated integration, *Proceedings of of SIGGRAPH 86*, ACM, New York, NY, USA, pp. 315–321.

Hensley, J., Scheuermann, T., Coombe, G., Singh, M. & Lastra, A. (2005). Fast summed-area table generation and its applications, *Computer Graphics Forum* 24: 547–555.

Hoffmann, G. (2000). CIE color space, *Technical report*, University of Applied Sciences, Emden.
URL: `http://www.fho-emden.de/~hoffmann/ciexyz29082000.pdf`

Hurvich, L. & Jameson, D. (1955). Some quantitative aspects of an opponent-colors theory. ii. brightness, saturation, and hue in normal and dichromatic vision, *Journal of the Optical Society of America* 45: 602–616.

Ikeda, M. & Ashizawa, S. (1991). Equivalent lightness of colored objects of equal munsell chroma and of equal munsell value at various illuminances, *Color Reserch and Application* 16: 72–80.

Khan, S. M. & Pattanaik, S. N. (2004). Modeling blue shift in moonlit scenes by rod cone interaction, *Journal of VISION* 4(8): 316a.

Kirk, A. G. & O'Brien, J. F. (2011). Perceptually based tone mapping for low-light conditions, *ACM SIGGRAPH 2011 papers*, SIGGRAPH '11, ACM, New York, NY, USA, pp. 42:1–42:10.

Krawczyk, G., Myszkowski, K. & Seidel, H.-P. (2005). Perceptual effects in real-time tone mapping, *Proceedings of the 21st Spring conference on Computer Graphics*, New York, NY, USA, pp. 195–202.

Lauritzen, A. (2007). *Summed-Area Variance Shadow Maps*, Addison-Wesley Professional, pp. 157–182.

Ledda, P., Chalmers, A., Troscianko, T. & Seetzen, H. (2005). Evaluation of tone mapping operators using a high dynamic range display, *ACM Trans. Graph.* 24: 640–648.

Ledda, P., Santos, L. P. & Chalmers, A. (2004). A local model of eye adaptation for high dynamic range images, *Proceedings of the 3rd international conference on Computer graphics, virtual reality, visualisation and interaction in Africa*, AFRIGRAPH '04, ACM, New York, NY, USA, pp. 151–160.

Mikamo, M., Slomp, M., Tamaki, T. & Kaneda, K. (2009). A tone reproduction operator accounting for mesopic vision, *ACM SIGGRAPH ASIA 2009 Posters*, New York, NY, USA, pp. 41:1–41:1.

Minnaert, M. (1954). *The Nature of Light and Colour in the Open Air*, Dover Publications.

Pattanaik, S. & Yee, H. (2002). Adaptive gain control for high dynamic range image display, *Proceedings of the 18th spring conference on Computer graphics*, SCCG '02, ACM, New York, NY, USA, pp. 83–87.

Reinhard, E. (2003). Parameter estimation for photographic tone reproduction, *Journal of Graphics Tools* 7(1): 45–52.

Reinhard, E., Heidrich, W., Debevec, P., Pattanaik, S., Ward, G. & Myszkowski, K. (2010). *High Dynamic Range Imaging: Acquisition, Display, and Image-Based Lighting (2nd Edition)*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Reinhard, E., Stark, M., Shirley, P. & Ferwerda, J. (2002). Photographic tone reproduction for digital images, *ACM Transactions on Graphics* 21: 267–276.

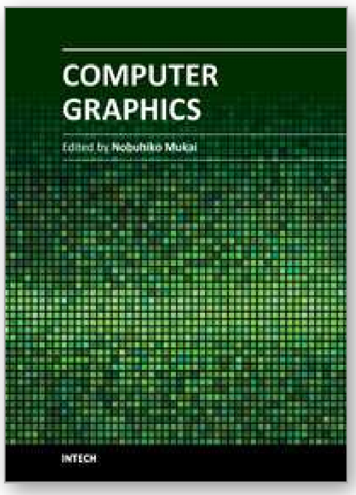
Slomp, M. & Oliveira, M. M. (2008). Real-time photographic local tone reproduction using summed-area tables, *Computer Graphics International 2008*, Istambul, Turkey, pp. 82–91.

Slomp, M., Tamaki, T. & Kaneda, K. (2010). Screen-space ambient occlusion through summed-area tables, *Proceedings of the 2010 First International Conference on Networking and Computing*, ICNC '10, IEEE Computer Society, Washington, DC, USA, pp. 1–8.

van de Hulst, H. C. (1957). *Light Scattering by Small Particles*, Wiley & Sons.

Viola, P. & Jones, M. (2004). Robust real-time object detection, *International Journal of Computer Vision* 57(2): 137–154.

Zhang, X. & Wandell, B. (1997). A spatial extension of CIELAB for digital color image reproduction, *Journal of the Society for Information Display (SID)* 5: 61–63.



Computer Graphics
Edited by Prof. Nobuhiko Mukai

ISBN 978-953-51-0455-1
Hard cover, 256 pages
Publisher InTech
Published online 30, March, 2012
Published in print edition March, 2012