08 Frequency Domain and Gradient Domain Tone Reproduction

In addition to local and global operators, there are two other classes of operators that work in a fundamentally different way. First, it may be possible under favorable conditions to separate illuminance from surface reflectance. By compressing only the illuminance component, an image may be successfully reduced in dynamic range. Second, we may exploit the fact that an image area with a high dynamic range tends to exhibit large gradients between neighboring pixels. This leads to a solution whereby the image is differentiated. Then the gradients are manipulated before the result is integrated into a compressed image.

Frequency-dependent operators are interesting from a historical perspective as well as for the observations about image structure they afford. These algorithms may therefore help us better understand the challenges we face when preparing HDR images for display. The following also explores gradient domain operators, in that they are algorithmically related to frequency domain operators.

8.1 FREQUENCY DOMAIN OPERATORS

Tone reproduction and dynamic range reduction are generally thought of as fairly recent developments. The problem was introduced to the field of lighting design in 1984 [186], and to the computer graphics community in 1993 [130,131]. However, HDR images and the problem of dynamic-range reduction are as old as the field of photography, for which the printing process may be seen as a tone-mapping technique. The problem also surfaced again with the invention of digital images. The first digital images were scanned with a bit depth of 12 bits, but could only be displayed with a bit depth of 8 bits. As a result, the first digital images had to be tone mapped prior to display. To our knowledge, the first digital tone-reproduction operator was published in 1968 by Oppenheim and colleagues [91].

Although this operator appears to be largely forgotten, the work itself contains several key ideas (including homomorphic filtering) that have found their way into numerous other tone-reproduction operators. In addition, the resulting tone-reproduction operator produces visually appealing output for a variety of images and perhaps deserves more attention than it currently receives.

Oppenheim's operator is a frequency-dependent compressor, in which low frequencies are attenuated more than higher frequencies. This approach was recently also taken by a technique called *bilateral filtering* [23] (Section 8.1.2). This term refers to an edge-preserving smoothing technique that forms the basis for various image-processing tasks, including tone reproduction. The bilateral filtering technique is used to separate an image into a base layer and a detail layer. The base layer tends to be low frequency and HDR, whereas the detail layer is high frequency and LDR. The tone-reproduction operator then proceeds by compressing the base layer before recombining it with the detail layer. At the same time, the output of the bilateral filter may be seen as providing a local adaptation value for each pixel, and therefore classification of this algorithm as a local operator would have been equally valid.

A similar separation into base and detail layers may be achieved with the trilateral filter [10], which is an extension of the bilateral filter. The difference between this and the bilateral filter lies in the technique used to separate the image into two layers.

All three techniques, however, apply a compression scheme that is frequency dependent, and thus they are grouped in this chapter. In the following subsections, each of these techniques is presented in more detail.

8.1.1 OPPENHEIM FREQUENCY-BASED OPERATOR

Under simplified assumptions, such as a scene consisting of diffuse objects only and no directly visible light sources, image formation may be thought of as the product of illuminance and reflectance. As indicated in Section 7.1.3, the illuminance component is then HDR, whereas the reflectance component is not. It would therefore be advantageous if we could separate the two components, and perform dynamic-range compression on the illuminance component only.

This approach implicitly assumes that the surfaces in a scene are diffuse. This is to an approximation true for many objects, but the method ignores high-frequency HDR phenomena such as specular reflections, caustics, and directly visible light sources. We would therefore not recommend this approach for images depicting these types of lighting.

For the remaining class of images, separation of reflectance and illuminance is to some degree possible by observing that illumination varies slowly over the image, whereas reflection is sometimes static and sometimes dynamic [91]. This is because objects tend to have well-defined edges and vary in size and texture. As such, partially independent processing of illuminance and reflectance is possible in the frequency domain.

Oppenheim et al. therefore suggest applying a whitening filter to the density representation of an image, which attenuates low frequencies while preserving higher frequencies. This is based on the observation that density representations of images tend to show a sharp peak in the low frequencies, with a plateau for medium and high frequencies.

As an aside, whitening is the process in which the amplitude of the Fourier representation is altered such that all frequencies carry an equal amount of energy. This is generally achieved by amplifying higher frequencies. The opposite approach, in which higher frequencies are attenuated, has the effect of blurring the image. These two effects are demonstrated in Figure 8.1.

Frequency-sensitive attenuation of an image thus starts by taking the logarithm of each pixel to compute densities. Then the FFT is computed on the density representation so that low frequencies may be attenuated more than high frequencies. The inverse Fourier transform is then applied to return to a density representation. In turn, the density image is exponentiated to yield a displayable image. For a Fourier-transformed density image, we experimented with the following attenuation function.

$$s(f) = (1 - c) + c\frac{kf}{1 + kf}$$

Both amplitude and phase spectra are multiplied by this scaling, which depends on frequency f and takes two user parameters c and k. The user parameter c controls

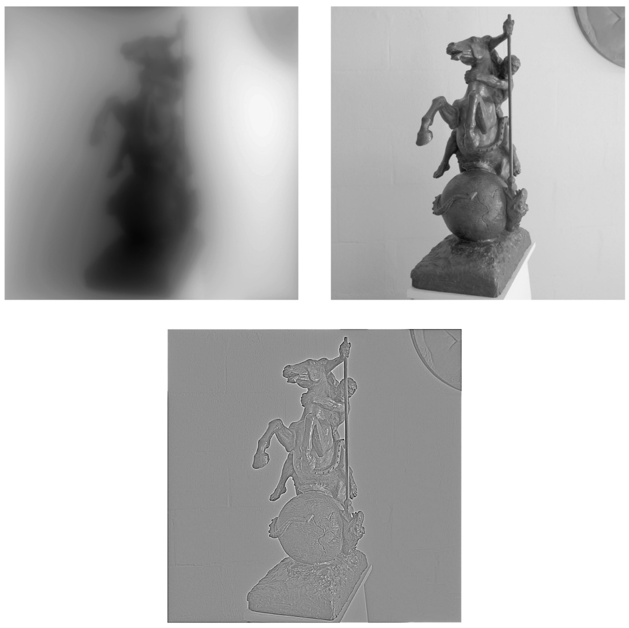

FIGURE 8.1 *The image in the middle was blurred by attenuating higher frequencies (top left) and whitened by amplifying higher frequencies (bottom). (Image courtesy of the Albin Polasek Museum, Winter Park, Florida.)*

the maximum amount of attenuation applied to the zero-frequency DC (direct current) component, whereas the k user parameter determines how rapidly the slope reaches the plateau of 1.

A reasonable default value for c is 0.5, as recommended by Oppenheim et al. [91], which generally lies between 0 and 1. The scaling functions $s(f)$ spanned

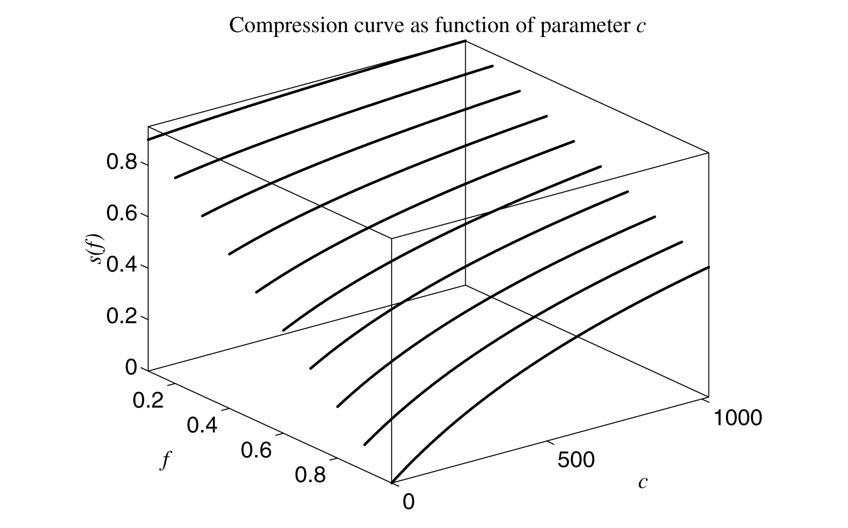

FIGURE 8.2 *To demonstrate Oppenheim's operator, the effect of the parameter choice of c on the scaling function $s(f)$ is shown.*

by different choices of c are plotted in Figure 8.2. An HDR image compressed with different values for c is shown in Figure 8.3.

The k parameter could be initialized to 0.01, with a sensible range for this parameter being [0.001,0.02]. Its impact on the shape of the scaling function $s(f)$ is shown in Figure 8.4. Images compressed with different values of k are presented in Figure 8.5. For smaller values of k, the plateau at which no attenuation is applied occurs for higher frequencies and thus the image is compressed more. The higher the value of k the sooner the plateau is reached, and less dynamic-range reduction is achieved.

In summary, Oppenheim et al. were the first to address the dynamic-range reduction problem. They proposed to attenuate low frequencies in the density (log)

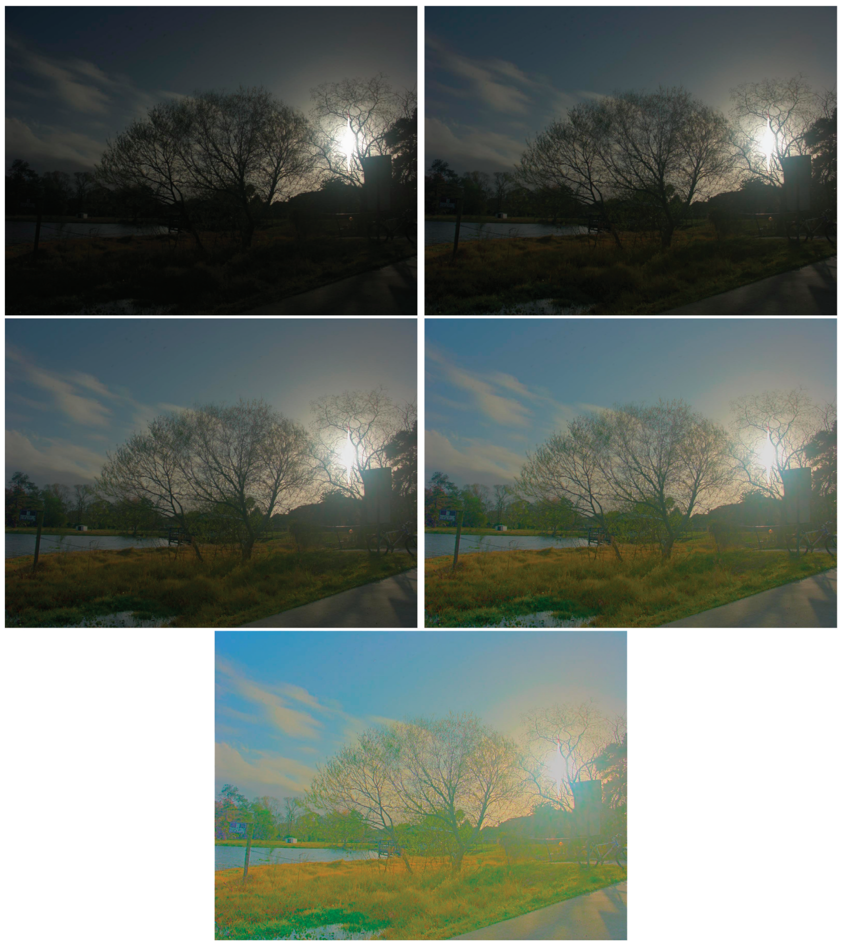

FIGURE 8.3 The effect of different values of c in Oppenheim's operator. In reading order, c is given values of 0.1, 0.3, 0.5, 0.7, and 0.9.

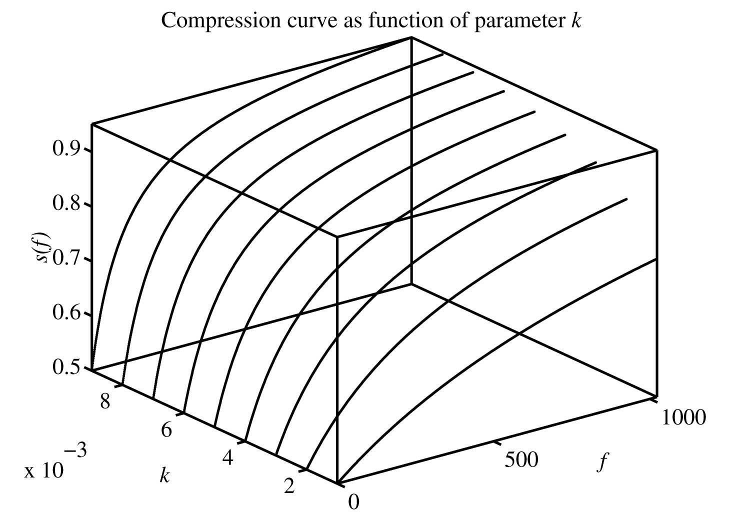

FIGURE 8.4 *Effect of the parameter choice of k on the scaling function $s(f)$ in Oppenheim's operator.*

domain, while preserving higher frequencies. This type of processing is called homomorphic filtering, which affords partially independent processing of illuminance and reflectance. They observed that reflectance is typically high frequency and LDR, whereas illuminance produces slow gradients within an arbitrary HDR. For images depicting sharp shadow boundaries, participating media, specular highlights, or directly visible light sources, this separation may not always be performed cleanly and the method may therefore not always yield satisfactory results.

Aspects of this algorithm — including homomorphic filtering (Section 7.1.3), separation of the image into illuminance and reflectance, and the concept of tone reproduction — were first introduced in Oppenheim's work. With a suitable choice

FIGURE 8.5 *The effect of different values of k in Oppenheim's operator. In reading order, k is increased from 0.002 to 0.004, 0.008, and 0.016.*

of the parameters c and k (introduced by us), this algorithm produces reasonable output, despite the theoretical restrictions mentioned previously.

8.1.2 DURAND BILATERAL FILTERING

The idea that an image may be separated into a high-frequency component that contains only LDR information and a low-frequency component with an HDR is explicitly exploited by Oppenheim's operator by attenuating low frequencies in the Fourier domain. Separation of an image into separate components whereby only one of the components needs to be compressed may also be achieved by applying an edge-preserving smoothing operator.

Durand and Dorsey introduced the bilateral filter to the computer graphics community and showed how it may be used to help solve the tone-reproduction problem [23]. Bilateral filtering is an edge-preserving smoothing operator that effectively blurs an image but keeps sharp edges intact. An example is shown in Figure 8.6, in which the smoothed image is shown on the right. Edges in this image are preserved (compare with the unprocessed image on the left), whereas interior regions have reduced detail. This section introduces a tone-reproduction operator that uses bilateral filtering and goes by the same name.

Blurring an image is usually achieved by convolving the image with a Gaussian filter kernel. The bilateral filter extends this idea by reducing the weight of the Gaussian kernel if the density difference is too large (see Equation 7.7). A second Gaussian is applied to density differences. Following Oppenheim, this method operates on a density image, rather than on linear values.

The result of this computation, as seen in Figure 8.6, is to some extent analogous to the illuminance component as discussed by Oppenheim et al. [91]. From the input image and this illuminance image, the reflectance image may be reconstructed by dividing the input and illuminance image. The smoothed image is known as the

reinhard v.2005/03/22 Prn:14/06/2005; 15:02 F:reinhard08.tex; VTEX/JOL p. 10

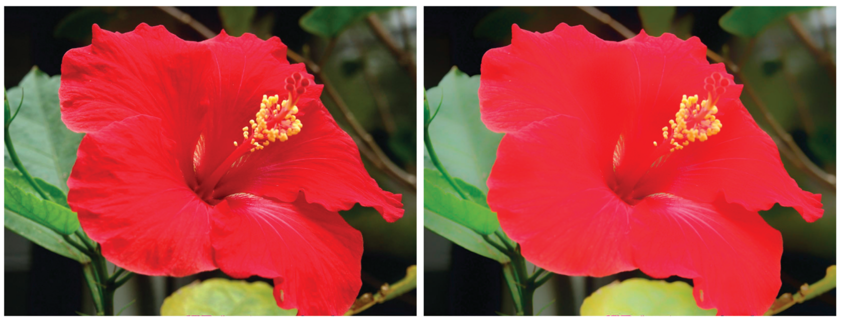

FIGURE 8.6 *The image on the left was smoothed with a bilateral filter, resulting in the image on the right.*

base layer, whereas the result of this division is called the detail layer. Note that the base and detail layers do not necessarily split the image into an illuminance and reflectance component. This method does not make the implicit assumption that the scene depicted is predominantly diffuse.

Examples of an HDR input image, an HDR base layer, and an LDR detail layer are shown in Figure 8.7. In this figure, the bilateral filter is applied to the luminance channel only. To reconstruct the base layer in color, we replaced the luminance channel of the image (in Yxy color space) with this output, exponentiated the result to yield a linear image, and converted to RGB. The detail layer was reconstructed in a similar manner.

After the bilateral filter is applied to construct base and detail layers in the logarithmic domain, the dynamic range may be reduced by scaling the base layer to a user-specified contrast. The two layers are then recombined and the result is exponentiated and converted to RGB to produce the final displayable result.

The amount of compression applied to the base layer is user specified, but Durand and Dorsey note that a target dynamic range of about 5 log units suffices for

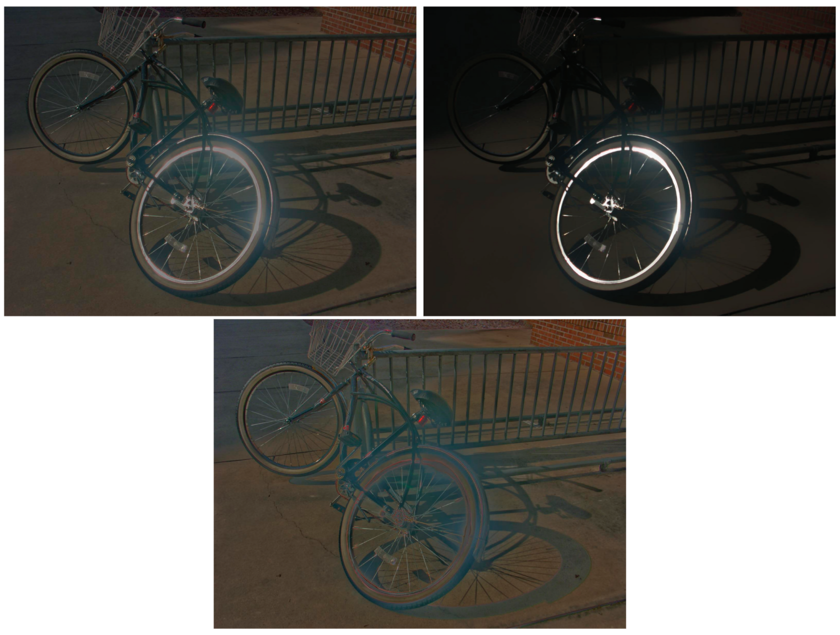

FIGURE 8.7 *HDR image tone mapped with bilateral filtering (left). The corresponding base layer and detail layers are shown in the right-hand and bottom images.*

many images.[1] For images that show light sources directly, this value may be adjusted. The effect of this parameter is shown in Figure 8.8, in which the contrast of the base layer was varied between 2 log units and 7 log units.

Bilateral filtering may be implemented directly in image space, but the convolution with a spatial Gaussian modulated by a Gaussian in density differences is rel-

1 We use the natural logarithm in this case.

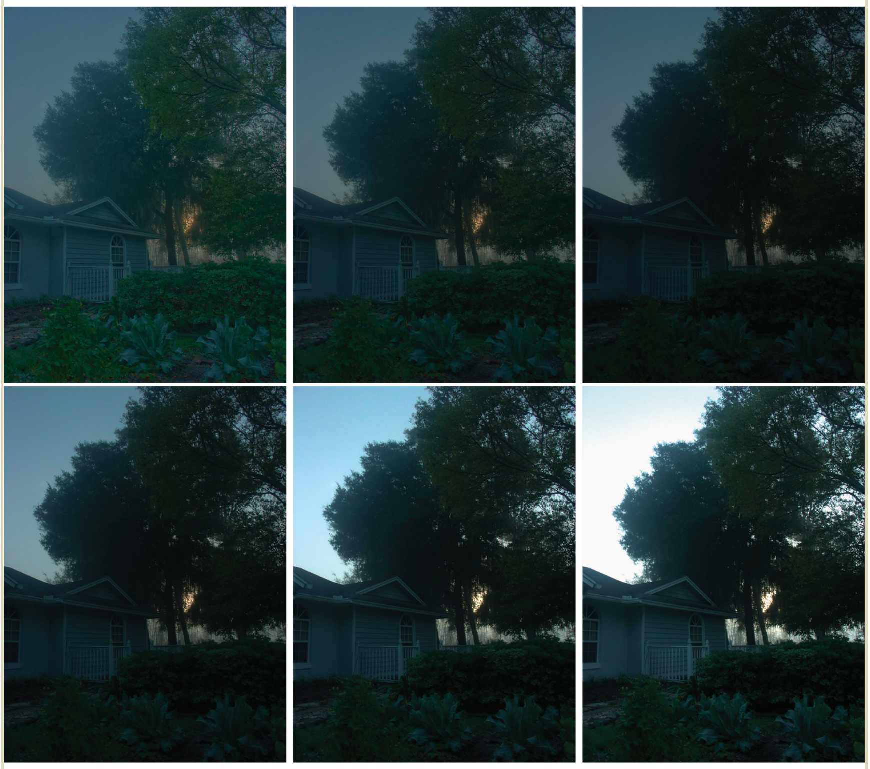

FIGURE 8.8 *Bilateral filtering results with varying amounts of compression applied to the base layer. The dynamic range of the base layer was set between 2 log units (top left-hand image) and 7 log units (bottom right-hand image).*

atively expensive to compute. In addition, the second Gaussian makes this method unsuitable for execution in the Fourier domain. Durand and Dorsey show how these disadvantages may be overcome by splitting the density differences into a number of segments [23]. The results are then recombined, yielding an approximate solution that in practice is indistinguishable from accurate spatial processing. The computation is given by the following.

$$D_j^{\text{smooth}}(x, y) = \frac{1}{k_j(x, y)} \sum_u \sum_v b_j(x, y, u, v)\ D(x-u, y-v)$$

$$k_j(x, y) = \sum_u \sum_v b_j(x, y, u, v)$$

$$b_j(x, y, u, v) = f\left(\sqrt{(x-u)^2 + (y-v)^2}\right) g\left(D(x-u, y-v) - D_j\right)$$

Here, the values D_j form a quantized set of possible values for pixel (x, y). The final output for this pixel is a linear combination of the output of the two smoothed values D_j^{smooth} and D_{j+1}^{smooth}. These two values are chosen such that D_j and D_{j+1} are the closest two values to the input density D of pixel (x, y).

For each segment j, the previous equation may be executed in the Fourier domain, thus gaining speedup. The number of segments depends on the dynamic range of the input image, as well as the choice of standard deviation for the Gaussian $g()$, which operates on density differences. A suitable choice for this standard deviation is about 0.4. The computation time of the bilateral filter depends on the number of segments. There is therefore a trade-off between computation time and visual quality, which may be chosen by specifying this standard deviation.

We have experimented with different values and show the results in Figure 8.9. For this particular image, the choice of standard deviation has a relatively small effect on its visual appearance. However, this parameter directly influences the number of segments generated, and thus affects the computation time. For this image, the largest standard deviation we chose was 8 log units, resulting in the creation of two segments. For values close to the default of 0.4, the number of segments is much higher due to the image's high dynamic range. This image was split into 19 segments for a standard deviation of 0.5, and into 38 segments for a standard deviation of 0.25. The computation times recorded for these images are graphed in Figure 8.10.

This computation time is substantially higher than those reported by Durand and Dorsey [23], most likely because the dynamic range of this image is higher

FIGURE 8.9 *Bilateral filtering showing results for different choices of standard deviation of the Gaussian filter operating on density differences, starting at 0.25 for the top left-hand image and doubling for each subsequent image. The bottom right-hand image was therefore created with a standard deviation of 8 log units.*

than many of their examples. In this chapter, we use a standard deviation of 0.4 as recommended in the original paper, but note that discrepancies in reported computation times may be due to the choice of images.

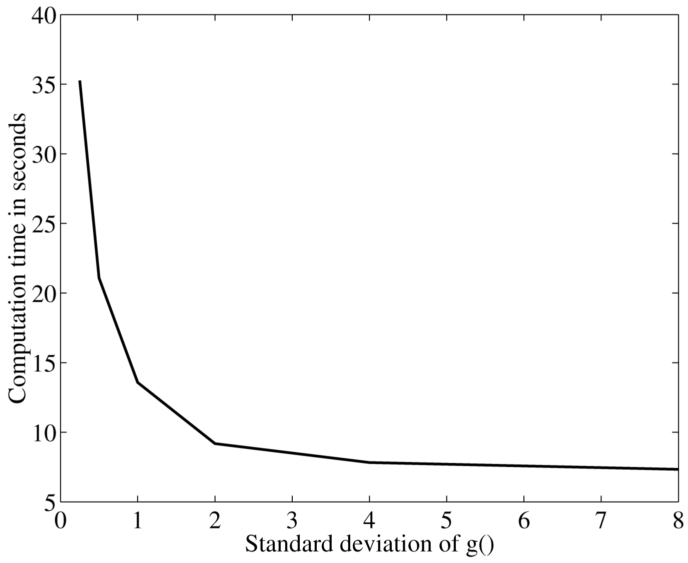

FIGURE 8.10 *Computation time of Durand and Dorsey's bilateral filter as a function of the standard deviation of the Gaussian filter.*

Further, Durand and Dorsey observed that bilateral filtering aims at low-pass filtering, and thus for most of the computations the full resolution of the input image is not required. It is therefore possible to sample the image using nearest-neighbor downsampling, perform bilateral filtering, and then upsample the results to the full resolution. This significantly reduces the computational cost of the algorithm for downsampling factors of up to 10 to 25. Higher factors will not yield a further reduction in computation time, because upsampling and linear interpolation will then start to dominate the computation. The visual difference between no downsampling and downsampling within this range is negligible. We therefore downsampled all results in this section with a factor of 16.

In summary, bilateral filtering is a worthwhile technique that achieves a hands-off approach to tone reproduction. The method is able to smooth an image without blurring across sharp edges. This makes the method robust against outliers and other anomalies. The method splits a density image into an HDR and an LDR layer. The HDR layer is then compressed and recombined with the other layer. The result is exponentiated to form an LDR image. Various techniques are available to speed up the process.

8.1.3 CHOUDHURY TRILATERAL FILTERING

Although the bilateral filter has attractive features for edge-preserving filtering, Choudhury and Tumblin note that this filter also has certain drawbacks. In particular, the filter smooths across sharp changes in the gradients of the image, and the filter poorly smooths high-gradient and high-curvature regions [10].

The trilateral filter aims to overcome these limitations by extending the bilateral filter. In fact, two modified versions of the bilateral filter are applied in succession. The algorithm starts by computing a density image from a luminance image, whereupon image gradients are computed. These gradients are then smoothed and used as an indicator of the amount by which the bilateral filter should be tilted to adapt to the local region. The smoothing itself is achieved through bilateral filtering. Figure 8.11 shows images of the various steps involved in the algorithm.

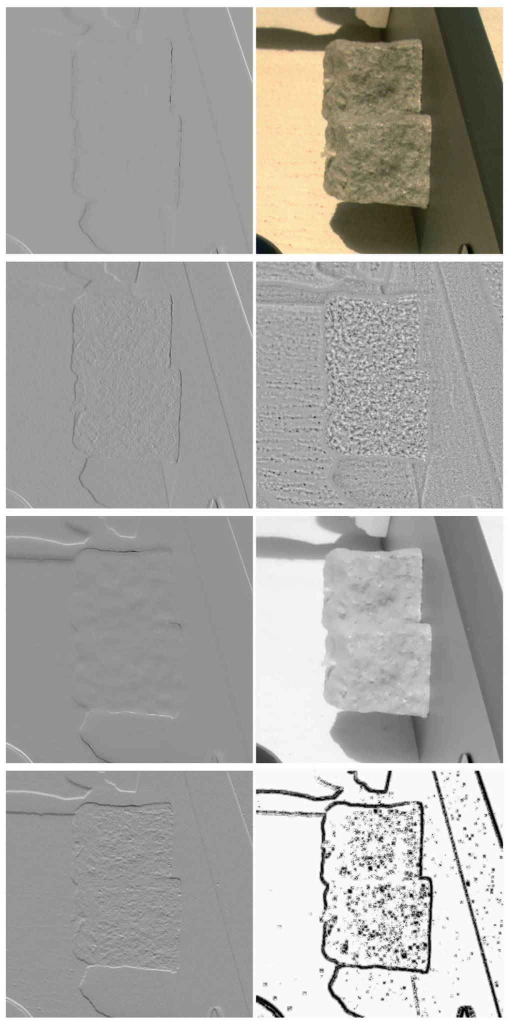

FIGURE 8.11 Image gradients and smoothed image gradients (top row) for x and y directions. The tilting angle A_θ is shown as the first image of the second row, followed by the output of the trilateral filter, which may be viewed as the base layer. The input density image and the base layer are then subtracted to produce the detail layer (third image on the second row). Compression of the base layer and recombination with the detail layer yields the final tone-mapped result (last image, second row).

With the filter kernel given by $b(x, y, u, v)$, the bilaterally smoothed tilting vector A may be computed for each pixel as follows.

$$\begin{aligned}
A(x, y) &= \frac{1}{w(x, y)} \sum_u \sum_v b(x, y, u, v) \nabla D_{\text{in}}(x-u, y-v) \\
w(x, y) &= \sum_u \sum_v b(x, y, u, v) \\
b(x, y, u, v) &= f\left(\sqrt{(x-u)^2 + (y-v)^2}\right) \\
&\quad \times g\left(\|\nabla D_{\text{in}}(x-u, y-v) - \nabla D_{\text{in}}(x, y)\|\right)
\end{aligned}$$

The filter output is normalized by the weight factor w. The gradients of the input are computed using forward differences, as follows.

$$\nabla D_{\text{in}}(m, n) \approx \left(D_{\text{in}}(m+1, n) - D_{\text{in}}(m, n), D_{\text{in}}(m, n+1) - D_{\text{in}}(m, n)\right)$$

If we were to apply a bilateral filter to the input image after tilting the filter by $A(x, y)$, its Gaussian constituents $f()$ and $g()$ would no longer be orthogonal. Therefore, rather than computing a spatial weight $s()$ for neighboring densities $D(x-u, y-v)$ by measuring the spatial distance between (x, y) and $(x-u, y-v)$, this distance is now measured through a plane of density values with orientation $P(x-u, y-v)$. This orientation is a scalar value that may be computed as follows.

$$P(x-u, y-v) = D_{\text{in}}(x, y) + A(x, y) \cdot (u, v)^T$$

Before computing trilateral output values, $P(x-u, y-v)$ is subtracted from the input density values to compute a local detail signal $D_\Delta(x-u, y-v)$, as follows.

$$D_\Delta(x-u, y-v) = D_{\text{in}}(x-u, y-v) - P(x-u, y-v)$$

The output of the trilateral filter $D^{\text{smooth}}(x, y)$ is then obtained as follows.

$$D^{\text{smooth}}(x, y) = D_{\text{in}}(x, y) + \frac{1}{w_\Delta(x, y)} \sum_u \sum_v b(x, y, u, v) D_\Delta(x-u, y-v)$$

$$w_\Delta(x,y) = \sum_u \sum_v n(x,y,u,v)$$

$$b(x,y,u,v) = f\left(\sqrt{(x-u)^2 + (y-v)^2}\right) g\big(D_\Delta(x-u,y-v)\big)\delta_A(x-u,y-v)$$

By tilting the trilateral filter it is possible to smooth more accurately in high-gradient regions, but this comes at the cost of a potential for extending the filter window beyond local boundaries into regions of dissimilar gradients. This may cause undesirable blurring across sharp ridges and corners where the bilaterally smoothed gradient A changes abruptly.

This problem is solved by the binary function δ_A introduced in the previous equation. This function exploits a feature of the functional shape of the smoothed gradient field A to limit the contribution of pixel $(x-u, y-v)$ if it lies across a sharp edge. A sharp edge is present if there is a large jump in the magnitude of A between (x,y) and $(x-u, y-v)$. Thus, δ_A is the Kronecker delta function, which is 1 if the gradient step is below a specified threshold R, and 0 if the jump in gradient magnitude is too large. This is represented as follows.

$$\delta_A(x-u,y-v) = \begin{cases} 1 & \text{if } \|A(x-u,y-v) - A(x,y)\| < R \\ 0 & \text{otherwise} \end{cases}$$

A computationally efficient way of approximating the search for gradients in a local neighborhood for a pixel (x,y) is to precompute a stack of minimum and maximum gradients at different spatial resolutions. We refer to the original paper on trilateral filtering for additional information [10].

Although the method has seven internal parameters, only one needs to be specified by the user. All other parameters are derived from this single user parameter. The user parameter $\sigma_{c,\Theta}$ is the neighborhood size of the bilateral gradient-smoothing filter, specified in pixels. The influence of this parameter on the various stages of processing is shown in Figure 8.12.

For small kernel sizes, too much detail ends up in the base layer, which is subsequently compressed. The consequence is that these details are absent from the final tone-mapped image. For larger values of $\sigma_{c,\Theta}$, the details are separated more sensibly from the HDR component and thus detail is preserved in the tone-mapped images. This is shown in the rightmost column in Figure 8.12, where $\sigma_{c,\Theta}$ is set

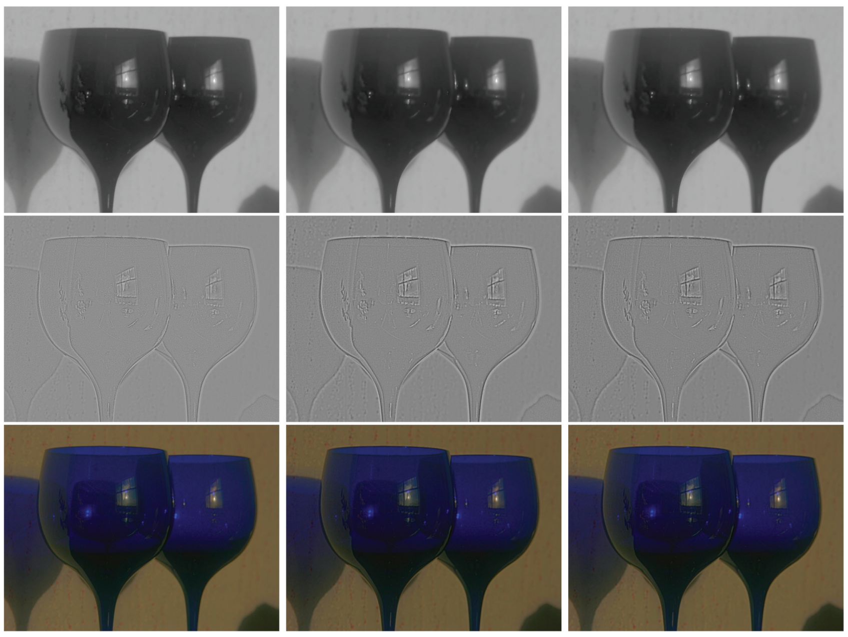

FIGURE 8.12 *For three different values of* σ *(3, 13, and 21 pixels), we show the base layer (top), the detail layer (middle), and the tone-mapped result (bottom).*

to 21. This value is recommended for practical use. Larger values have an adverse effect on the computation time without creating better images.

For the purpose of comparison, Figure 8.13 shows an image tone mapped with both bilateral and trilateral filters. With comparable parameter choices, the overall impression of the two images is similar, although several differences between the two images exist. In particular, the trilateral filter affords a better visualization of

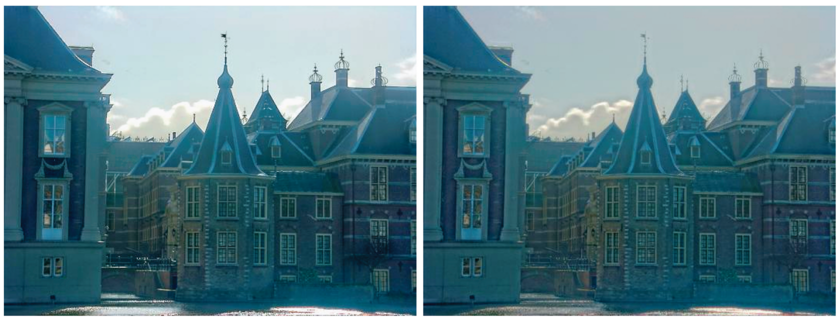

FIGURE 8.13 *Bilateral (left) and trilateral filters (right) applied to the same image with comparable parameter settings.*

the clouds. On the other hand, the tree in the lower right-hand corner is better preserved by the bilateral filter.

In summary, the trilateral filter is a further development over the bilateral filter. The filter smooths the image while preserving edges. Good results are achieved by tilting the filter kernel dependent on the local gradient information in an image. Like Oppenheim's method and Durand and Dorsey's bilateral filtering approach, Choudhury and Tumblin's trilateral filter is used to separate a density image into an LDR high-frequency image, and an HDR low-frequency image. The latter is compressed and recombined with the former to produce a tone-mapped density image. This result is then exponentiated to compute a displayable image.

8.2 GRADIENT DOMAIN OPERATORS

High-frequency components in an image cause rapid changes from one pixel to the next. On the other hand, low-frequency features cause the differences between neighboring pixels to be relatively small. It is therefore possible to partially distinguish between illuminance and reflectance in a different way by considering the gradients in the image. Under the assumption of diffusely reflecting scenes, this separation may be reasonably successful, as shown by Horn's lightness computation (discussed in the following section).

Although such separation depends on thresholding, tone reproduction does not necessarily require separation of illuminance and reflectance. In addition, HDR imagery frequently depicts scenes that deviate significantly from the assumption of diffuse reflection. Fattal et al. have shown that image gradients may be attenuated rather than thresholded, leading to a capable tone-reproduction operator (discussed in Section 8.2.2).

8.2.1 HORN LIGHTNESS COMPUTATION

The first to explore the idea of separating reflectance from illuminance on the basis of the gradient magnitude was Berthold Horn [53]. His work outlines a computational model of human lightness perception, that is a perceptual quantity that correlates with surface reflectance. Like Stockham and colleagues [91,123], this work assumes that each pixel of an image is formed as the product of illumination and surface reflectance, as follows.

$$L_\mathrm{v}(x, y) = E_\mathrm{v}(x, y) r(x, y)$$

Here, $L_\mathrm{v}(x, y)$ is the pixel's luminance and $E_\mathrm{v}(x, y)$ and $r(x, y)$ are illuminance and reflectance components, respectively. In the log domain, a density image would represent the same information, as follows.

$$\begin{aligned} D(x, y) &= \log(L_\mathrm{v}(x, y)) \\ &= \log(E_\mathrm{v}(x, y)) + \log(r(x, y)) \end{aligned}$$

Taking the derivative of $D(x, y)$ gives us the gradient, which is a 2-vector of partial derivatives in the horizontal and vertical directions. The gradient field of an image may be approximated using forward differences, as follows.

$$\begin{aligned} \nabla G(x, y) &= \big(G_\mathrm{x}(x, y), G_\mathrm{y}(x, y)\big) \\ &= \big(D(x+1, y) - D(x, y), D(x, y+1) - D(x, y)\big) \end{aligned}$$

Note that differences in the log domain correspond to ratios in linear space. By computing a gradient field of a density image we are effectively computing contrast ratios.

Edges in an image will produce sharp pulses in the gradient field, whereas the spatial variation of illumination will produce only small gradient values. To separate reflectance from illuminance, it is now possible to threshold the gradient and discard any small gradients, as follows.

$$\nabla G(x, y) = 0 \qquad \text{iff} \quad \sqrt{G_\mathrm{x}(x, y)^2 + G_\mathrm{y}(x, y)^2} < t$$

Integration of the remaining gradients yields an image that represents lightness. Integration of a discrete image is straightforward in one dimension, but amounts to solving a partial differential equation in two dimensions. The particular form of this equation is as follows.

$$\nabla^2 D(x, y) = \operatorname{div} G(x, y)$$

This is Poisson's equation with ∇^2 the Laplacian operator and $\operatorname{div} G(x, y)$ the divergence of $G(x, y)$. The Laplacian and divergence may be approximated in two dimensions using a differencing scheme, as follows.

$$\begin{aligned}
\nabla^2 D(x, y) &\approx D(x+1, y) + D(x-1, y) + D(x, y+1) \\
&\quad + D(x, y-1) - 4D(x, y) \\
\operatorname{div} G(x, y) &\approx G_\mathrm{x}(x, y) - G_\mathrm{x}(x-1, y) + G_\mathrm{y}(x, y) - G_\mathrm{y}(x, y-1)
\end{aligned}$$

The Poisson equation cannot be solved analytically, but must be approximated numerically. The method of choice is the full multigrid method, for which off-the-shelf routines are available [101]. Finally, the resulting density image $D(x, y)$ is exponentiated to produce the final image $L_\mathrm{d}(x, y)$.

The success of separating illuminance from reflectance in this manner depends on the choice of threshold value t. Setting the threshold too low will cause the resulting image to contain both the reflectance component and some residual illuminance. If the threshold is chosen too high, the integrated result will only partially represent reflectance.

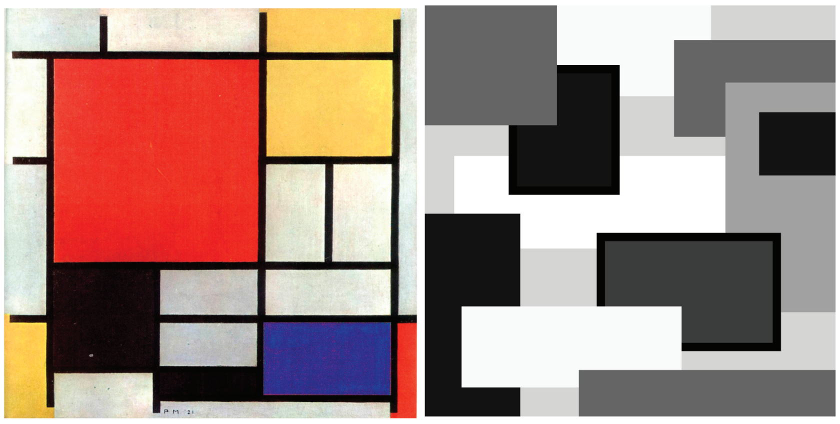

FIGURE 8.14 *Left: a reproduction of one of Piet Mondrian's paintings. On the right, a mini-world of Mondrian (from [88]).*

It is also important to note that this method assumes that no light sources are directly visible in the image. Horn presented his work in the context of Land's retinex theory, which was tested with mini-worlds of Mondrian [69].[2] In such worlds, scenes are flat areas divided into subregions of uniform matte color (see Figure 8.14). The lighting of such worlds creates smooth shading variations within each panel, but sharp gradient jumps between regions. Thus, the observation that reflectance causes sharp spikes in the gradient whereas illuminance is smoothly varying holds for this type of idealized scene.

For practical scenes that are generally more complicated, this assumption may not hold. In particular, if there are light sources directly visible in the image one

2 Mini-worlds of Mondriaan are inspired by the neo-plasticist painting style pioneered by famous Dutch artist Piet Mondriaan. Over time, the spelling has become anglicized so that mini-worlds of Mondriaan are now more commonly known as mini-worlds of Mondrian.

may expect the illuminance component to also exhibit large gradients, causing this approach to fail to successfully separate illuminance and reflectance. Depth discontinuities, specular highlights, and the presence of fluorescent materials may also cause the separation of reflectance from illuminance to be incomplete. Horn concludes that the method may be reasonable for the computation of lightness, but not for computing reflectance when applied to general images.

In Figure 8.15 the output of this approach is shown for different threshold values t. Small gradients are indeed removed, whereas dependent on the choice of threshold value reflectance edges are reasonably well respected. The images produced by this technique bear a resemblance to those created by bilateral filtering. In particular, compare the results of Figure 8.15 with Figure 8.6. Both techniques blur images without blurring across sharp edges. We therefore speculate that Horn's lightness computations may be viewed as an early example of an edge-preserving smoothing operator.

For the purpose of demonstration, Figure 8.15 shows an LDR image because it is close in nature to a mini-world of Mondrian. HDR images do not tend to adhere to the restrictions imposed by the mini-worlds of Mondriaan. The direct application of the previous thresholding technique is therefore not practical. On the other hand, large gradients in HDR images are correlated with illuminance variations. We therefore applied the same thresholding technique to an HDR image, although now we remove gradients that are larger than the threshold t, as follows.

$$\nabla G(x, y) = 0 \qquad \text{iff} \quad \sqrt{G_{\mathrm{x}}(x, y)^2 + G_{\mathrm{y}}(x, y)^2} > t$$

Results of this new thresholding scheme are shown in Figure 8.16. It is clear that this thresholding scheme is a fairly crude method of bringing an HDR image within a displayable range. It indicates that compressing the gradient field in some fashion may be a viable approach, although perhaps not using simple thresholding.

Although Horn was largely interested in computational models of human lightness perception, we have shown that a small modification could make the technique suitable for HDR compression. Thresholding may be too crude for practical purposes, and the appropriate selection of a suitable threshold would be a matter of trial and error. On the other hand, modifying the gradient field of an image and then integrating the result does present an opportunity for effective dynamic

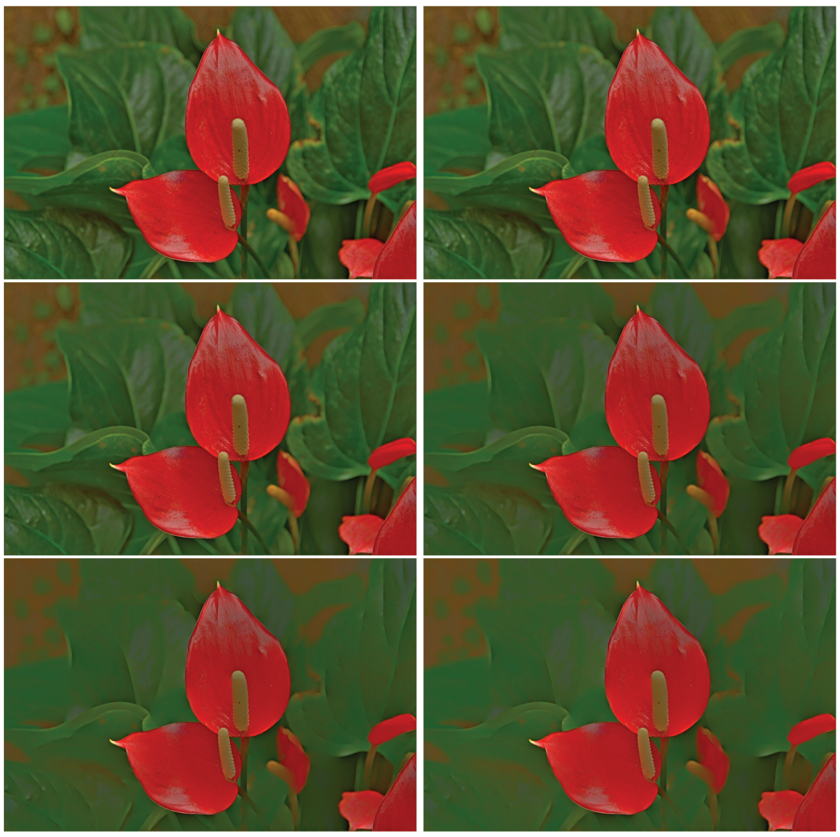

FIGURE 8.15 Horn's lightness computation for threshold values of $t = 0.0$ (top left) through $t = 0.1$ in increments of 0.02. The original photograph is shown in the top left.

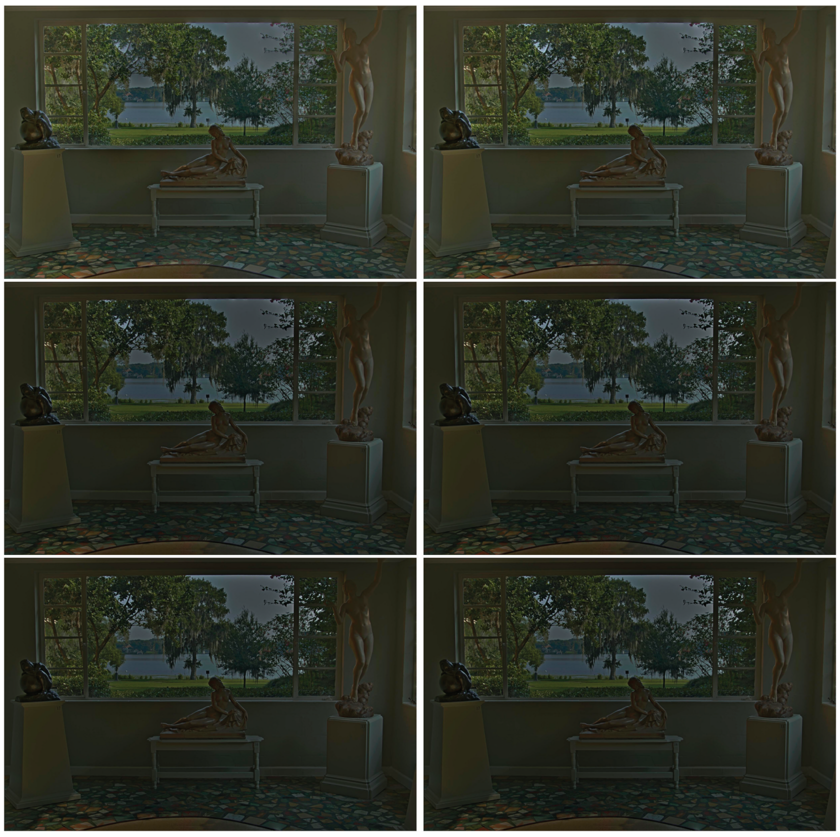

FIGURE 8.16 *New thresholding scheme applied to Horn's lightness computation using threshold values of $t = 0.25$ (top left) through $t = 1.50$ in increments of 0.25. (Image courtesy of the Albin Polasek Museum, Winter Park, Florida.)*

range reduction. This approach is taken by Fattal's gradient domain compression algorithm, discussed in the following section.

8.2.2 FATTAL GRADIENT DOMAIN COMPRESSION

Fattal et al. presented an alternative compression algorithm that achieves HDR reduction by applying a compressive function to the gradient field [32]. Following Horn, they compute the gradient field of a density image, manipulate these gradients, and then integrate by solving a Poisson equation.

However, rather than thresholding the gradient field their compressive function is more sophisticated. Fattal et al. observe that any drastic change in luminance across an HDR image gives rise to luminance gradients with a large magnitude. On the other hand, fine details (such as texture) correspond to much smaller gradients. The proposed solution should therefore identify gradients at various spatial scales and attenuate their magnitudes. By making the approach progressive (i.e., larger gradients are attenuated more than smaller gradients), fine details may be preserved while compressing large luminance gradients. After computing a density image $D(x,y) = \log(L(x,y))$, the method proceeds by computing the gradient field $\nabla G(x,y)$, as follows.

$$\nabla G(x,y) = \big(D(x+1,y) - D(x,y), D(x,y+1) - D(x,y)\big)$$

This gradient field is then attenuated by multiplying each gradient with a compressive function $\Phi(x,y)$, resulting in a compressed gradient field $\nabla G'(x,y)$, as follows.

$$\nabla G'(x,y) = \nabla G(x,y)\Phi(x,y)$$

As in Horn's approach, a compressed density image $D'(x,y)$ is constructed by solving the Poisson equation, as follows.

$$\nabla^2 D'(x,y) = \operatorname{div} G'(x,y)$$

The rationale for solving this partial differential equation is that we seek a density image $D'(x,y)$ with a gradient that approximates $G'(x,y)$ as closely as possible. In the least squares sense, this conforms to minimizing the following integral.

$$\iint \left\|\nabla D'(x,y) - G(x,y)\right\|^2 dx\,dy$$
$$= \iint \left(\frac{\delta D'(x,y)}{\delta x} - G_{\mathrm{x}}(x,y)\right)^2 + \left(\frac{\delta D'(x,y)}{\delta y} - G_{\mathrm{y}}(x,y)\right)^2 dx\,dy$$

According to the variational principle, $D'(x,y)$ must satisfy the Euler–Lagrange equation [101], yielding

$$2\left(\frac{\delta^2 D'(x,y)}{\delta x^2} - \frac{\delta G_{\mathrm{x}}(x,y)}{\delta x}\right) + 2\left(\frac{\delta^2 D'(x,y)}{\delta y^2} - \frac{\delta G_{\mathrm{y}}(x,y)}{\delta y}\right) = 0$$

Rearranging terms produces this Poisson equation, which may be solved using the full multigrid method [101]. Exponentiating the compressed density image then produces the tone-mapped image $L_{\mathrm{d}}(x,y)$, as follows.

$$L_{\mathrm{d}}(x,y) = \exp\bigl(D'(x,y)\bigr)$$

To a large extent the choice of attenuation function will determine the visual quality of the result. In the previous section, a very simple example is shown by setting large gradients to zero. This produces compressed images, but at the cost of visual quality. Fattal et al. follow a different approach and only attenuate large gradients.

Their attenuation function is based on the observation that edges exist at multiple scales [148]. To detect significant ratios, a multiresolution edge-detection scheme is employed. Rather than attenuate a significant gradient at the resolution where it is detected, the attenuation is propagated to the full resolution gradient field before being applied. This scheme avoids haloing artifacts.

First, a Gaussian pyramid $D_0, D_1 \ldots D_d$ is constructed from the density image. The number of levels d is chosen such that at this coarsest level the resolution of the image is at least 32 by 32. At each level s, a gradient field $\nabla G_s(x,y)$ is computed using central differences, as follows.

$$\nabla G_s(x,y) = \left(\frac{D_s(x+1,y) - D_s(x-1,y)}{2^{s+1}}, \frac{D_s(x,y+1) - D_s(x,y-1)}{2^{s+1}}\right)$$

At each level and for each pixel, a scale factor may be computed based on the magnitude of the gradient, as follows.

$$\phi_s(x, y) = \frac{\alpha}{\|\nabla G_s(x, y)\|} \left(\frac{\|\nabla G_s(x, y)\|}{\alpha} \right)^{\beta}$$

This scale factor features two user-defined parameters α and β. Gradients larger than α are attenuated provided that $\beta < 1$, whereas smaller gradients are not attenuated and in fact may even be somewhat amplified. A reasonable value for α is 0.1 times the average gradient magnitude. Fattal et al. suggest setting user parameter β between 0.8 and 0.9, although we have found that larger values (up to about 0.96) are sometimes required to produce a reasonable image. The attenuation function $\Phi(x, y)$ can now be constructed by considering the coarsest level first and then propagating partial values in top-down fashion, as follows.

$$\begin{aligned} \Phi_d(x, y) &= \phi_d(x, y) \\ \Phi_s(x, y) &= U(\Phi_{s+1}(x, y))\phi_s(x, y) \\ \Phi(x, y) &= \Phi_0(x, y) \end{aligned}$$

Here, $\Phi_s(x, y)$ is the partially accumulated scale factor at level s, and $U()$ is an upsampling operator with linear interpolation. For one image, the two parameters α and β were varied to create the tableau of images shown in Figure 8.17. For smaller values of β, more details are visible in the tone-mapped image. A similar effect occurs for decreasing values of α. Both parameters afford a trade-off between the amount of compression applied to the image and the amount of detail visible. In our opinion, choosing values that are too small for either α or β produces images that contain too much detail to appear natural.

FIGURE 8.17 *Fattal's gradient domain compression. The user parameters α and β were varied: from left to right α is given values of 0.10, 0.25, and 0.40. From top to bottom, β is 0.85, 0.89, and 0.95.*

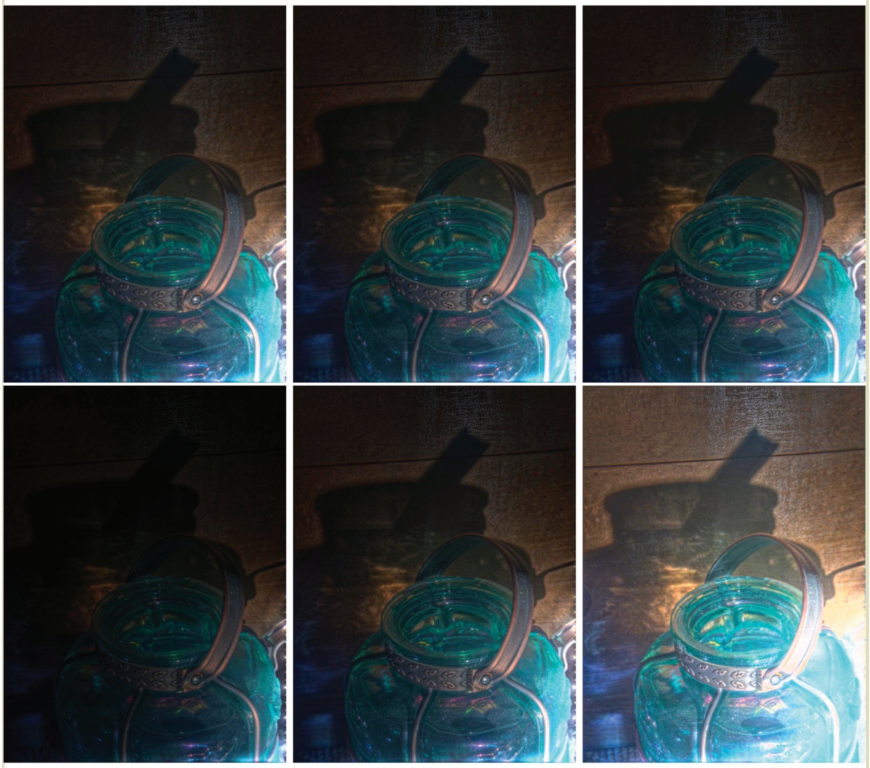

FIGURE 8.18 *Clamping in Fattal's gradient domain compression. Top row: clamping 0.1, 1, and 10% of the dark pixels. Bottom row: clamping 0.1, 1, and 10% of the light pixels.*

This approach may benefit somewhat from clamping, a technique whereby a percentage of the smallest and largest pixel intensities is removed and the remaining range of intensities is scaled to fit the display range. Figure 8.18 shows the effect of varying the percentage of dark pixels that are clamped (top row) and separately varying the percentage of light pixels that are clamped (bottom row). The effect of clamping dark pixels is fairly subtle, but dramatic effects may be achieved by clamping a percentage of light pixels. In general, if a tone-mapped image appears too gray it may be helpful to apply some clamping at the light end. This removes outliers that would cause the average luminance to drop too much after normalization.

In summary, Fattal's gradient domain compression technique attenuates gradients, but does so in a gentler manner than simply thresholding. The two user parameters provide a trade-off between the amount of compression and the amount of detail available in the image. Too much compression has the visual effect of exaggerated small details. The technique is similar in spirit to Horn's lightness computations and is the only recent example of a tone-reproduction operator working on gradient fields.

8.3 PERFORMANCE

For many applications the speed of operation is important. For most tone-reproduction operators, performance is simply a function of the size of the image. In this section we report results obtained on an Apple iBook G3 running at 800 MHz using images with 1,600 by 1,200 pixels.

We show the timing required to execute each tone-mapping operator, but exclude the time it takes to read the image from disk or write the result to file. We also routinely normalize the result of the tone-reproduction operators and apply gamma correction. None of these operations is included in the timing results.

All timing results are summarized in Table 8.1. This table should be interpreted with the following caveats. The timing given for Miller's operator is a rough average of the timings shown in Figure 8.19. The timing for the bilateral filter is given for a downsampling factor of 16. The computation time of Chiu's spatially variant operator is representative of the algorithm explained in this chapter, but not for the full algorithm as described by Chiu et al. (in that we have omitted the iterative smoothing stage).

Operator	**Time (in seconds)**
GLOBAL OPERATORS	
Miller's operator	≈ 15.0
Tumblin–Rushmeier's operator	3.2
Ward's scale factor	0.96
Ferwerda's operator	1.0*
Ferschin's exponential mapping	3.0
Logarithmic mapping	3.4
Drago's logarithmic mapping	2.8
Reinhard's global photographic operator	3.7
Reinhard and Devlin's photoreceptor model	9.7
Ward's histogram adjustment	3.4
Schlick's uniform rational quantization	3.4
LOCAL OPERATORS	
Chiu's spatially variant operator	10.0
Rahman and Jobson's multiscale retinex	120.0
Johnson and Fairchild's iCAM	66.0
Ashikhmin's operator	120.0
Reinhard's local photographic operator	80.0
GRADIENT DOMAIN OPERATORS	
Horn's lightness computation	45.0
Fattal's gradient domain compression	45.5
FREQUENCY-BASED OPERATORS	
Oppenheim's operator	12.4
Durand's bilateral filtering	23.5

TABLE 8.1 *Computation time for all operators using 1,600 by 1,200 images on an Apple iBook with 512 MB RAM and a G3 processor running at 800 MHz. Note: * Ferwerda's operator does not include the algorithm to lower visual acuity in scotopic lighting conditions.*

8.3.1 LOCAL AND GLOBAL OPERATORS

In general, global operators are the fastest to execute because normally only two or three passes over the image are required. In each pass, only very simple computations are performed. For applications that require real-time operation, global operators would be the first choice.

Local operators rely on the computation of local averages for each pixel. Such local averages are often computed by convolving the image with a filter kernel. For filter kernels larger than about 3 by 3 pixels, it is faster to Fourier transform both the image and the filter kernel and perform a pairwise multiplication in the Fourier domain. The convolved image is then obtained by applying an inverse Fourier transform on the result. Whether the convolution is computed directly or by means of the Fourier transform, local operators tend to be much slower than their global counterparts.

The performance of global operators is usually dependent only on the size of the image. The exception is Miller's operator, which is also weakly dependent on the maximum display luminance. The running time as a function of maximum display luminance is plotted in Figure 8.19. In all cases, the execution time remains below about 17 seconds.

The performance of the iCAM model depends on whether the adaptation level is computed from the luminance channel only, or for all three channels independently. The former takes 44 seconds, whereas the latter takes 66 seconds.

Our implementation of the multiscale observer model requires a substantial amount of memory to store the full image pyramid. We were not able to reliably measure the execution time of this operator for the default image size of 1,600 by 1,200 because our iBook did not have sufficient memory (512 MB) to complete the computation without significant swapping.

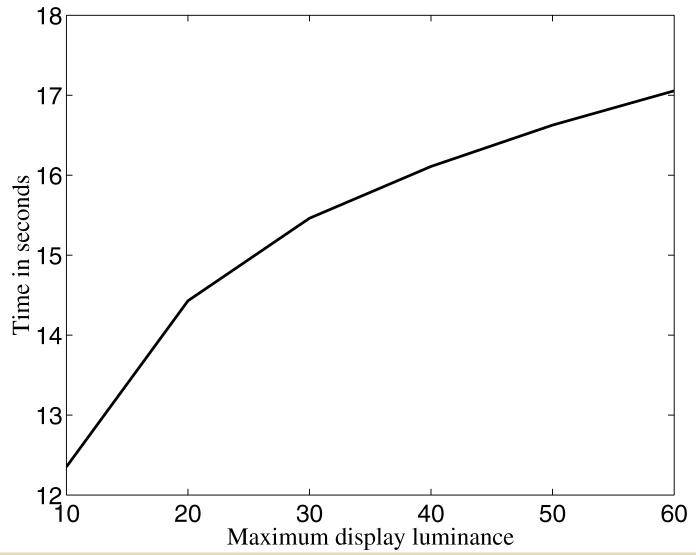

FIGURE 8.19 *Running time of Miller's operator as function of the maximum display luminance user parameter.*

The performance of the local version of Reinhard's photographic operator is about 80 seconds. For two reasons, this constitutes a performance gain with respect to other operators that also build a Gaussian pyramid. First, this operator compresses a luminance channel, as opposed to three color channels in the multiscale observer model. In comparison with Ashikhmin's operator, the total number of levels in the Gaussian pyramid is smaller.

8.3.2 GRADIENT AND FREQUENCY DOMAIN OPERATORS

Gradient domain operators require an integration step that is both approximate and costly, though less so than techniques that build image pyramids. The numerical integration method of choice is the full multigrid method, which dominates the computation time of this class of operators. For example, Horn's lightness computation takes about 45 seconds. We found the gradient domain operator to be very similar in performance to Horn's operator (about 45.5 seconds).

Frequency domain operators rely on FFTs to obtain a frequency-space representation. We used the public domain FFTW library [39], and for these operators the performance of the FFT transform dominates the computation time. Note that the speed of executing an FFT depends strongly on the size of the image. Any image size that has a large number of factors will be substantially faster than image sizes that have a smaller number of factors. Although the running time depends on image size, this dependency is not linear. Our results are obtained with 1,600-by-1,200 images. These numbers may be factored into $2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 5 \times 5$ and $2 \times 2 \times 2 \times 2 \times 3 \times 5 \times 5$, and therefore have eight and seven factors, respectively. This yields a relatively fast computation of FFTs. On the other hand, if the image were smaller by 1 pixel in each dimension (1,599 by 1,199 pixels) the factors would be $3 \times 13 \times 41$ and 11×109. This would have a negative impact on the computation time. In general, images that are powers of 2 will be the fastest and images the size of prime numbers will be the slowest to compute. In terms of performance, it is beneficial to pad images to a power-of-2 size if a Fourier transform needs to be computed.

An alternative filtering technique is to apply a fast but approximate filter, such as that described by Burt and Adelson [8]. For filter kernels with a Gaussian shape, this may speed up the computations, but with a loss of accuracy. We have found that this approximation is useful only for larger filter kernels. For very small filter kernels, this approximation may not be accurate enough.

Oppenheim's frequency-based operator takes about 12.4 seconds. Bilateral filtering takes about 23 seconds when a downsampling factor of 16 is selected. This is somewhat slower than Oppenheim's operator, although not dramatically so. Smaller downsampling factors will cause the computation time to increase significantly. If no downsampling is used, the computation time of the same image increases to 685 seconds. The progression of computation times as a function of downsampling factors is depicted in Figure 8.20.

The computational complexity of the trilateral filter is of necessity higher than for the bilateral filter, in that its main computational cost is the double application of the bilateral filter. If the same optimizations are employed as outlined for the

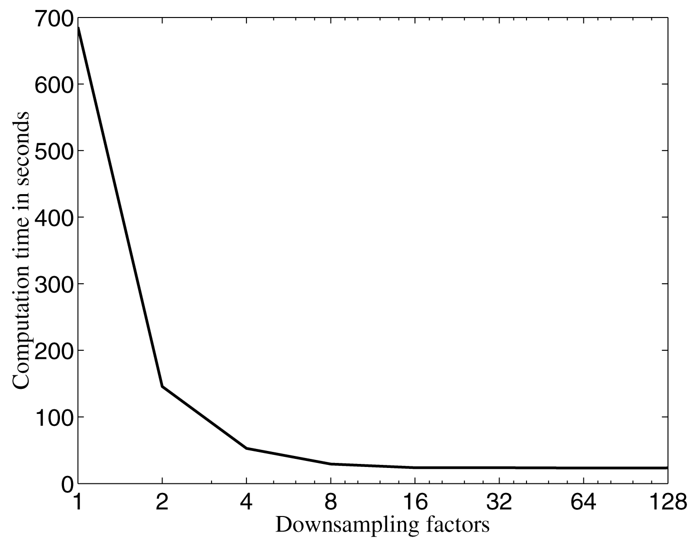

FIGURE 8.20 *Computation time (in seconds) for the bilateral filter as a function of downsampling factors.*

bilateral filter in Section 8.1.2, we expect the running times to be about double that of the bilateral filter. However, our implementation does not incorporate these optimizations, and we therefore recorded computation times that are substantially higher.

8.4 DISCUSSION

Tone-reproduction operators achieve dynamic-range reduction based on a small set of distinct observations. We have chosen to classify operators into four classes, two of which are loosely based on image formation and two others operating in the spatial domain.

The underlying assumption of the first two classes is that images are formed by light being reflected from surfaces. In particular, the light intensities recorded in an image are assumed to be the product of light being reflected from a surface and the surface's ability to reflect. This led to Oppenheim's frequency-dependent attenuation. Subdivision of an image into base and detail layers may be seen as a frequency-dependent operation. The bilateral and trilateral filters, however, lift the restriction that images are assumed largely diffuse. These filters operate well for images depicting scenes containing directly visible light sources, specular reflections, and so on.

A parallel development has occurred in the class of gradient domain operators. Horn's lightness computation is aimed at disentangling illumination from reflectance by thresholding gradients. It necessarily assumes that scenes are diffuse. This restriction is lifted by Fattal et al., who attenuate large gradients but keep small gradients intact.

Various tone-reproduction operators use partial computational models of the human visual system to achieve dynamic-range reduction. A returning theme within this class of operators is the notion of adaptation luminance. Global operators often derive an adaptation level from the (log) average luminance of the image, whereas local operators compute a separate adaptation level for each pixel. Local adaptation levels are effectively weighted local averages of pixel neighborhoods. If the size of these neighborhoods is grown to include the entire image, these local operators default to global operators. It should therefore be possible to replace the global adaptation level of a global operator with a locally derived set of adaptation levels. The validity of this observation is demonstrated in Figure 8.21, in which the semisaturation constant of Reinhard and Devlin's photoreceptor model is fed with various luminance adaptation computations.

There are various ways of computing local adaptation luminances. The use of a stack of Gaussian-blurred images may be closest to the actual working of the human visual system, and with a carefully designed scale selection mechanism (such as shown by Reinhard et al.'s photographic operator, as well as Ashikhmin's operator), local adaptation levels may be computed that minimize haloing artifacts. Alternatively, edge-preserving smoothing filters may be used to derive local adapta-

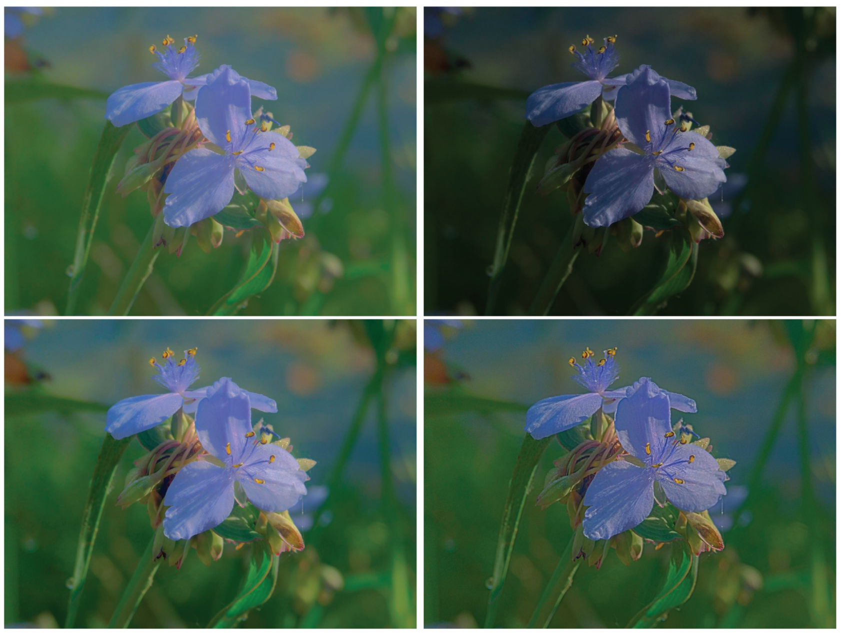

FIGURE 8.21 *Global photoreceptor model (top left) and global photoreceptor model with the light adaptation parameter set to 0 (top right), followed by local versions in which adapting luminances are computed with Durand's bilateral filter (bottom left) and Pattanaik's gain control operator (bottom right) [23,96].*

tion luminances. Examples of such filters are the bilateral and trilateral filters, as well as the low-curvature image simplifier [132] and the mean shift algorithm [13].

The human visual system adapts over a period of time to novel lighting conditions. This is evident when entering a dark tunnel from bright daylight. It takes a short period of time before all details inside the tunnel become visible. Such time-dependent behavior may also be included in tone-reproduction operators [22,35, 43,95,112].

Finally, we would like to stress the fact that each of these operators has its own strengths and weaknesses. Computational complexity, presence or absence of artifacts, and ability to deal with extreme HDR images should all be considered. We believe that there is no single operator that will be the best choice for all tasks, or even for all images.

For instance, in photography the purpose of tone reproduction may be to produce an image that appears as beautiful as possible. It may not be necessary to show every last detail in the captured image to achieve this goal. In addition, for this type of application a fully automatic operator may be less desirable than one that provides intuitive user parameters that allow the final result to be steered in the direction the photographer has in mind.

On the other hand, the task may be to visualize data, for instance if the HDR data is the result of a scientific simulation. In such cases it may be more important to visualize all important details than to produce an appealing image. It may be undesirable to have user parameters in this case.

For video and film, tone reproduction should produce consistent results between consecutive frames. In addition, tone reproduction could conceivably be used creatively to steer the mood of the scene, and thus help convey a story.

Thus, appropriate tone-reproduction operators should be matched to the task at hand. The current state of affairs is that we do not know how to match an operator to a given task. Selection of tone-reproduction operators is usually a matter of taste, as well as public availability of source code. We hope to alleviate the latter problem by having made the source code of all of our implementations available on the companion DVD-ROM.