

# Local Laplacian Filters: Edge-Aware Image Processing with a Laplacian Pyramid

By Sylvain Paris, Samuel W. Hasinoff, and Jan Kautz

## Abstract

The Laplacian pyramid is ubiquitous for decomposing images into multiple scales and is widely used for image analysis. However, because it is constructed with spatially invariant Gaussian kernels, the Laplacian pyramid is widely believed to be ill-suited for representing edges, as well as for edge-aware operations such as edge-preserving smoothing and tone mapping. To tackle these tasks, a wealth of alternative techniques and representations have been proposed, for example, anisotropic diffusion, neighborhood filtering, and specialized wavelet bases. While these methods have demonstrated successful results, they come at the price of additional complexity, often accompanied by higher computational cost or the need to postprocess the generated results. In this paper, we show state-of-the-art edge-aware processing using standard Laplacian pyramids. We characterize edges with a simple threshold on pixel values that allow us to differentiate large-scale edges from small-scale details. Building upon this result, we propose a set of image filters to achieve edge-preserving smoothing, detail enhancement, tone mapping, and inverse tone mapping. The advantage of our approach is its simplicity and flexibility, relying only on simple point-wise nonlinearities and small Gaussian convolutions; no optimization or postprocessing is required. As we demonstrate, our method produces consistently high-quality results, without degrading edges or introducing halos.

## 1. INTRODUCTION

Laplacian pyramids have been used to analyze images at multiple scales for a broad range of applications such as compression,<sup>6</sup> texture synthesis,<sup>18</sup> and harmonization.<sup>32</sup> However, these pyramids are commonly regarded as a poor choice for applications in which image edges play an important role, for example, edge-preserving smoothing or tone mapping. The isotropic, spatially invariant, smooth Gaussian kernels on which the pyramids are built are considered almost antithetical to edge discontinuities, which are precisely located and anisotropic by nature. Further, the decimation of the levels, that is, the successive reduction by factor 2 of the resolution, is often criticized for introducing aliasing artifacts, leading some researchers (e.g., Li et al.<sup>21</sup>) to recommend its omission. These arguments are often cited as a motivation for more sophisticated schemes such as anisotropic diffusion,<sup>1,29</sup> neighborhood filters,<sup>19,34</sup> edge-preserving optimization,<sup>4,11</sup> and edge-aware wavelets.<sup>12</sup>

While Laplacian pyramids can be implemented using simple image-resizing routines, other methods rely on more sophisticated techniques. For instance, the bilateral filter relies on a spatially varying kernel,<sup>34</sup> optimization-based methods (e.g., Fattal et al.,<sup>13</sup> Farbman et al.,<sup>11</sup> Subr et al.,<sup>31</sup> and Bhat et al.<sup>4</sup>) minimize a spatially inhomogeneous energy, and other approaches build dedicated basis functions for each new image (e.g., Szeliski,<sup>33</sup> Fattal,<sup>12</sup> and Fattal et al.<sup>15</sup>). This additional level of sophistication is also often associated with practical shortcomings. The parameters of anisotropic diffusion are difficult to set because of the iterative nature of the process, neighborhood filters tend to over-sharpen edges,<sup>5</sup> and methods based on optimization do not scale well due to the algorithmic complexity of the solvers. While some of these shortcomings can be alleviated in postprocessing, for example, bilateral filtered edges can be smoothed,<sup>3,10,19</sup> this induces additional computation and parameter setting, and a method producing good results directly is preferable. In this paper, we demonstrate that state-of-the-art edge-aware filters can be achieved with standard Laplacian pyramids. We formulate our approach as the construction of the Laplacian pyramid of the filtered output. For each output pyramid coefficient, we render a filtered version of the full-resolution image, processed to have the desired properties according to the corresponding local image value at the same scale, build a new Laplacian pyramid from the filtered image, and then copy the corresponding coefficient to the output pyramid. The advantage of this approach is that while it may be nontrivial to produce an image with the desired property everywhere, it is often easier to obtain the property locally. For instance, global detail enhancement typically requires a nonlinear image decomposition (e.g., Fattal et al.,<sup>14</sup> Farbman et al.,<sup>11</sup> and Subr et al.<sup>31</sup>), but enhancing details in the vicinity of a pixel can be done with a simple S-shaped contrast curve centered on the pixel intensity. This local transformation only achieves the desired effect in the neighborhood of a pixel, but is sufficient to estimate the fine-scale Laplacian coefficient of the output. We repeat this process for each coefficient independently and collapse the pyramid to produce the final output.

The original version of this paper was published in *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2011)* 30, 4 (Aug. 2011), 68:1–68:12.

We motivate this approach by analyzing its effect on step edges and show that edges can be differentiated from small-scale details with a simple threshold on color differences. We propose an algorithm that has a  $\mathcal{O}(N \log N)$  complexity for an image with  $N$  pixels. While our algorithm is not as fast as other techniques, it can achieve visually compelling results hard to obtain with previous work. We demonstrate our approach by implementing a series of edge-aware filters such as edge-preserving smoothing, detail enhancement, tone mapping, and inverse tone mapping. We provide numerous results, including large-amplitude image transformations. None of them exhibit halos, thereby showing that high-quality halo-free results can be indeed obtained using only the Laplacian pyramid, which was previously thought impossible.

**Contributions.** The main contribution of this work is a flexible approach to achieve edge-aware image processing through simple point-wise manipulation of Laplacian pyramids. Our approach builds upon a new understanding of how image edges are represented in Laplacian pyramids and how to manipulate them in a local fashion. Based on this, we design a set of edge-aware filters that produce high-quality halo-free results (Figure 1).

## 2. RELATED WORK

**Edge-aware image processing.** Edge-aware image manipulation has already received a great deal of attention and we refer to books and surveys for an in-depth presentation.<sup>1,20,27</sup> Recently, several methods have demonstrated satisfying results with good performance (e.g., Chen et al.,<sup>7</sup> Farbman et al.,<sup>11</sup> Fattal,<sup>12</sup> Subr et al.,<sup>31</sup> Criminisi et al.,<sup>8</sup> He et al.,<sup>17</sup> and Kass and Solomon<sup>19</sup>). Our practical contribution is to provide filters that consistently achieve results at least as good, have easy-to-set parameters, can be implemented with only basic image-resizing routines, are noniterative, and do not rely on optimization or postprocessing. In particular, unlike gradient-domain methods (e.g., Fattal et al.<sup>13</sup>), we do not need to solve the Poisson equation which may introduce artifacts with nonintegrable gradient

fields. From a conceptual standpoint, our approach is based on image pyramids and is inherently multiscale, which differentiates it from methods that are expressed as a two-scale decomposition (e.g., Chen et al.,<sup>7</sup> Subr et al.,<sup>31</sup> and He et al.<sup>17</sup>).

**Pyramid-based edge-aware filtering.** As described earlier, pyramids are not the typical representation of choice for filtering an image in an edge-preserving way, and only a few techniques along these lines have been proposed. A first approach is to directly rescale the coefficients of a Laplacian pyramid; however, this typically produces halos.<sup>21</sup> While halos may be tolerable in the context of medical imaging (e.g., Vuylsteke and Schoeters,<sup>36</sup> and Dippel et al.<sup>9</sup>), they are unacceptable in photography.

Fattal et al.<sup>13</sup> avoid halos by using a Gaussian pyramid to compute scaling factors applied to the image gradients. They reconstruct the final image by solving the Poisson equation. In comparison, our approach directly manipulates the Laplacian pyramid of the image and does not require global optimization. Fattal et al.<sup>14</sup> use a multiscale image decomposition to combine several images for detail enhancement. Their decomposition is based on repeated applications of the bilateral filter. Their approach is akin to building a Laplacian pyramid but without decimating the levels and with a spatially varying kernel instead of a Gaussian kernel. However, their study is significantly different from ours because it focuses on multi-image combination and speed. In a similar spirit, Farbman et al.<sup>11</sup> compute a multiscale edge-preserving decomposition with a least-squares scheme instead of bilateral filtering. This work also differs from ours since its main concern is the definition and application of a new optimization-based filter. In the context of tone mapping, Mantiuk et al.<sup>23</sup> model human perception with a Gaussian pyramid. The final image is the result of an optimization process, which departs from our goal of working only with pyramids.

Fattal<sup>12</sup> describes wavelet bases that are specific to each image. He takes edges explicitly into account to define the basis functions, thereby reducing the correlation between

**Figure 1.** We demonstrate edge-aware image filters based on the manipulation of Laplacian pyramids. Our approach produces high-quality results, without degrading edges or introducing halos, even at extreme settings. Our approach builds upon standard image pyramids and enables a broad range of effects via simple point-wise nonlinearities (shown in corners). For an example image (a), we show results of tone mapping using our method, creating a natural rendition (b) and a more exaggerated look that enhances details as well (c). Laplacian pyramids have previously been considered unsuitable for such tasks, but our approach shows otherwise.



(a) Input HDR image tone-mapped with a simple gamma curve (details are compressed)



(b) Our pyramid-based tone mapping, set to preserve details without increasing them



(c) Our pyramid-based tone mapping, set to strongly enhance the contrast of details

pyramid levels. From a conceptual point of view, our work and Fattal’s are complementary. Whereas he designed pyramids in which edges do not generate correlated coefficients, we seek to better understand this correlation to preserve it during filtering.

Li et al.<sup>21</sup> demonstrate a tone-mapping operator based on a generic set of spatially invariant wavelets, countering the popular belief that such wavelets are not appropriate for edge-aware processing. Their method relies on a corrective scheme to preserve the spatial and intrascale correlation between coefficients, and they also advocate computing each level of the pyramid at full resolution to prevent aliasing. However, when applied to Laplacian pyramids, strong corrections are required to avoid halos, which prevents a large increase of the local contrast. In comparison, in this work, we show that Laplacian pyramids can produce a wide range of edge-aware effects, including extreme detail amplification, without introducing halos.

Gaussian pyramids are closely related to the concept of Gaussian scale-space defined by filtering an image with a series of Gaussian kernels of increasing size. While these approaches are also concerned with the correlation between scales created by edges, they are used mostly for purposes of analysis (e.g., Witkin<sup>37</sup> and Witkin et al.<sup>38</sup>).

**Background on Gaussian and Laplacian pyramids.** Our approach is based on standard image pyramids, whose construction we summarize briefly (for more detail, see Burt and Adelson<sup>6</sup>). Given an image  $I$ , its *Gaussian pyramid* is a set of images  $\{G_\ell\}$  called *levels*, representing progressively lower resolution versions of the image, in which high-frequency details progressively disappear. In the Gaussian pyramid, the bottom-most level is the original image,  $G_0 = I$ , and  $G_{\ell+1} = \text{downsample}(G_\ell)$  is a low-pass version of  $G_\ell$  with half the width and height. The filtering and decimation process is iterated  $n$  times, typically until the level  $G_n$  has only a few pixels. The *Laplacian pyramid* is a closely related construct, whose levels  $\{L_\ell\}$  represent details at different spatial scales, decomposing the image into roughly separate frequency bands. Levels of the Laplacian pyramid are defined by the details that distinguish successive levels of the Gaussian pyramid,  $L_\ell = G_\ell - \text{upsample}(G_{\ell+1})$ , where  $\text{upsample}(\cdot)$  is an operator that doubles the image size in each dimension using a smooth kernel. The top-most level of the Laplacian pyramid, also called the *residual*, is defined as  $L_n = G_n$  and corresponds to a tiny version of the image. A Laplacian pyramid can be *collapsed* to reconstruct the original image by recursively applying  $G_\ell = L_\ell + \text{upsample}(G_{\ell+1})$  until  $G_0 = I$  is recovered.

### 3. DEALING WITH EDGES IN LAPLACIAN PYRAMIDS

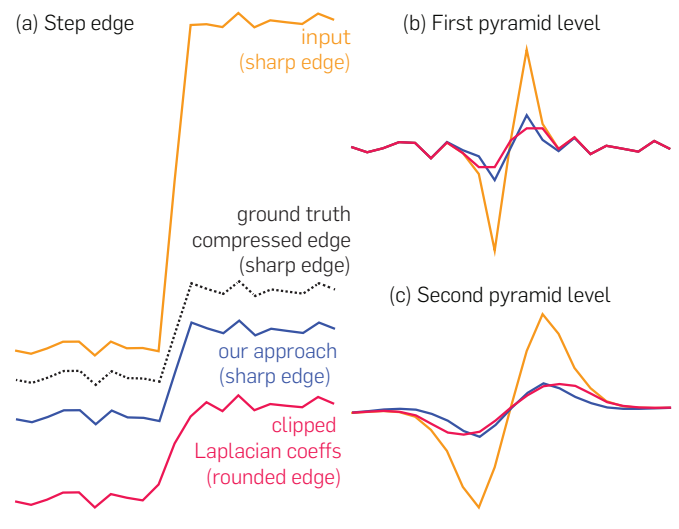
The goal of edge-aware processing is to modify an input signal  $I$  to create an output  $I'$ , such that the large discontinuities of  $I$ , that is, its edges, remain in place, and such that their profiles retain the same overall shape. For example, the amplitude of significant edges may be increased or reduced, but the edge transitions should not become smoother or sharper. The ability to process images in this edge-aware fashion is particularly important for techniques that manipulate the image in a spatially varying way, such

as image enhancement or tone mapping. Failure to account for edges in these applications leads to distracting visual artifacts such as halos, shifted edges, or reversals of gradients. In the following discussion, for the sake of illustration, we focus on the case where we seek to reduce the edge amplitude—the argument when increasing the edge amplitude is symmetric.

In this work, we characterize edges by the magnitude of the corresponding discontinuity in a color space that depends on the application; we assume that variations due to edges are larger than those produced by texture. This model is similar to many existing edge-aware filtering techniques (e.g., Aubert and Kornprobst<sup>1</sup> and Paris et al.<sup>27</sup>); we will discuss later the influence that this assumption has on our results. Because of this difference in magnitude, Laplacian coefficients representing an edge also tend to be larger than those due to texture. A naive approach to decrease the edge amplitude while preserving the texture is to truncate these large coefficients. While this creates an edge of smaller amplitude, it ignores the actual “shape” of these large coefficients and assigns the same lower value to all of them. This produces an overly smooth edge, as shown in Figure 2.

Intuitively, a better solution is to scale down the coefficients that correspond to edges, to preserve their profile, and to keep the other coefficients unchanged, so that only the edges are altered. However, it is unclear how to separate these two kinds of coefficients since edges with different profiles generate different coefficients across scales. On the other hand, according to our model, edges

**Figure 2. Range compression applied to a step edge with fine details (a). The different versions of the edge are offset vertically so that their profiles are clearly visible. Truncating the Laplacian coefficients smooths the edge (red), an issue which Li et al.<sup>21</sup> have identified as a source of artifacts in tone mapping. In comparison, our approach (blue) preserves the edge sharpness and very closely reproduces the desired result (black). Observing the shape of the first two levels (b, c) shows that clipping the coefficients significantly alters the shape of the signal (red vs. orange). The truncated coefficients form wider lobes whereas our approach produces profiles nearly identical to the input (blue vs. orange).**



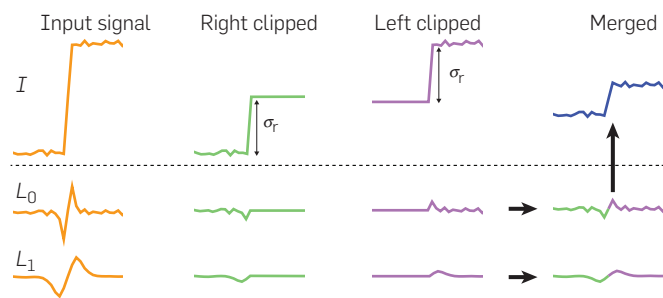


are easy to identify in image space; a threshold on color differences suffices to differentiate edges from variations due to texture. This is a key aspect of our approach: we generate new pyramid coefficients by working primarily on the *input image* itself, rather than altering the pyramid coefficients directly.

The overall design of our algorithm derives from this insight: we build an approximation of the desired output image specific to *each pyramid coefficient*. This is a major difference with the existing literature. Whereas previous techniques are formulated in terms of optimization (e.g., Farbman et al.<sup>11</sup>), PDEs (e.g., Perona and Malik<sup>29</sup>), or local averaging (e.g., Tomasi and Manduchi<sup>34</sup>), we express our filter through the computation of these local image approximations together with standard image pyramid manipulations. In practice, we use locally processed versions of the input to recompute values for each pyramid coefficient, and combine all of these new coefficient values into the final result. For each coefficient at location  $(x, y)$  and level  $\ell$ , we first determine the region in the input image on which this coefficient depends. To reduce the amplitude of edges, for example, we clamp all the pixels values in that region so that the difference to the average value does not exceed a user-provided threshold. This processed image has the desired property that edges are now limited in amplitude, to at most twice the threshold. This also has the side effect of flattening the details across the edge. As we discuss below, these details are not lost, they are actually captured by pyramid coefficients centered on the other side of the edge as illustrated in Figure 3. Then, we compute the Laplacian pyramid of this processed image to create coefficients that capture this property. In particular, this gives us the value of the coefficient  $(x, y, \ell)$  that we seek. Another way of interpreting our method is that we locally filter the image, for example, through a local contrast decrease, and then determine the corresponding coefficient in the Laplacian pyramid. We repeat this process, such that each coefficient in the pyramid is computed.

**Detail preservation.** As mentioned earlier, a reasonable concern at this point is that the clamped image has lost

**Figure 3. Simple view of our range compression approach, which is based on thresholding and local processing. For a step-like signal similar to the one in Figure 2, our method effectively builds two Laplacian pyramids, corresponding to clipping the input based on the signal value to the left and right of the step edge, then merging their coefficients as indicated by the color coding.**



details in the thresholded regions, which in turn could induce a loss in the final output. However, the loss of details does not transfer to our final result. Intuitively, the clamped details are on “the other side of the edge” and are represented by other coefficients. Applying this scheme to all pyramid coefficients accurately represents the texture on each side of the edge, while capturing the reduction in edge amplitude (Figure 3). Further, clamping affects only half of the edge and, by combining coefficients on “both sides of the edge,” our approach reconstructs an edge profile that closely resembles the input image, that is, the output profiles do not suffer from oversmoothing. Examining the pyramid coefficients reveals that our scheme fulfills our initial objective, that is, that the edge coefficients are scaled down while the other coefficients representing the texture are preserved (Figure 2).

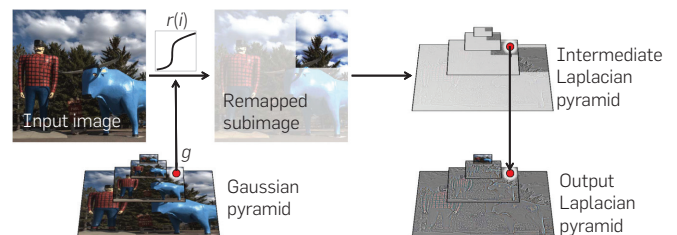
#### 4. LOCAL LAPLACIAN FILTERING

We now formalize the intuition gained in the previous section and introduce *Local Laplacian Filtering*, our new method for edge-aware image processing based on the Laplacian pyramid. A visual overview is given in Figure 4 and the pseudo-code is provided in Algorithm 1.

In Local Laplacian Filtering, an input image is processed by constructing the Laplacian pyramid  $\{L[I']\}$  of the output, one coefficient at a time. For each coefficient  $(x, y, \ell)$ , we generate an intermediate image  $\tilde{I}$  by applying a point-wise monotonic remapping function  $r_{g,\sigma}(\cdot)$  to the original full-resolution image. This remapping function, whose design we discuss later, depends on the local image value from the Gaussian pyramid  $g = G_\ell(x, y)$  and the user parameter  $\sigma$  which is used to distinguish edges from details. We compute the pyramid for the intermediate image  $\{L[\tilde{I}]\}$  and copy the corresponding coefficient to the output  $\{L[I']\}$ . After all coefficients of the output pyramid have been computed, we collapse the output pyramid to get the final result.

A direct implementation of this algorithm yields a complexity in  $\mathcal{O}(N^2)$  with  $N$  being the number of pixels in the image, since each coefficient entails the construction of another pyramid with  $\mathcal{O}(N)$  pixels. However, this cost can be

**Figure 4. Overview of the basic idea of our approach. For each pixel in the Gaussian pyramid of the input (red dot), we look up its value  $g$ . Based on  $g$ , we remap the input image using a point-wise function, build a Laplacian pyramid from this intermediate result, then copy the appropriate pixel into the output Laplacian pyramid. This process is repeated for each pixel over all scales until the output pyramid is filled, which is then collapsed to give the final result. For more efficient computation, only parts of the intermediate pyramid need to be generated.**





reduced in a straightforward way by processing only the subpyramid needed to evaluate  $L_\ell[\bar{I}](x, y)$ , illustrated in Figure 4. The base of this subpyramid lies within a  $K \times K$  subregion  $R$  of the input image  $I$ , where  $K = \mathcal{O}(2^\ell)$ ; for Laplacian pyramids built using a standard 5-tap interpolation filter, it can be shown that  $K = 3(2^{\ell+2} - 1)$ . Put together with the fact that level  $\ell$  contains  $\mathcal{O}(N/2^\ell)$  coefficients, each level requires the manipulation of  $\mathcal{O}(N)$  coefficients in total. Since there are  $\mathcal{O}(\log N)$  levels in the pyramid, the overall complexity of our algorithm is  $\mathcal{O}(N \log N)$ . Later we will see that some applications only require a fixed number of levels to be processed or limit the depth of the subpyramids to a fixed value, reducing the complexity of our algorithm further.

**Remapping function for gray-scale images.** We assume the user has provided a parameter  $\sigma$  such that intensity

**Algorithm 1**  $\mathcal{O}(N \log N)$  Version of Local Laplacian Filtering

**input:** image  $I$ , parameter  $\sigma$ , remapping function  $r$   
**output:** image  $I'$

- 1: compute input Gaussian pyramid  $\{G[I]\}$
- 2: **for all** coefficients at position  $(x, y)$  and level  $\ell$  **do**
- 3:  $g \leftarrow G_\ell(x, y)$
- 4: determine subregion  $R$  of  $I$  needed to evaluate  $L_\ell(x, y)$
- 5: create temporary buffer  $\bar{R}$  of the same size
- 6: **for all** pixels  $(u, v)$  of  $R$  **do**
- 7:   apply remapping function:  $\bar{R}(u, v) \leftarrow r(R(u, v), g, \sigma)$
- 8: **end for**
- 9: compute subpyramid  $\{L[\bar{R}]\}$
- 10: update output pyramid:  $L_\ell[I'](x, y) \leftarrow L_\ell[\bar{R}](x, y)$
- 11: **end for**
- 12: collapse output pyramid:  $I' \leftarrow \text{collapse}(\{L_\ell[I']\})$

Our algorithm considers the pyramid coefficients one by one (Step 2). Each of them is computed using the pixels from the finest resolution (Step 4) by applying the remapping function to them (Step 7) and building a Laplacian pyramid of the remapped data (Step 9). We copy the relevant coefficient into the output pyramid (Step 10) and once all the coefficients have been computed, we collapse pyramid the get the final result (Step 12).

variations smaller than  $\sigma$  should be considered fine-scale details and larger variations are edges. As a center point for this function we use  $g = G_\ell(x, y)$ , which represents the image intensity at the location and scale where we compute the output pyramid coefficient. Intuitively, pixels closer than  $\sigma$  to  $g$  should be processed as details and those farther than  $\sigma$  away should be processed as edges. We differentiate their treatment by defining two functions  $r_d$  and  $r_e$ , such that  $r(i) = r_d(i)$  if  $|i - g| \leq \sigma$  and  $r(i) = r_e(i)$  otherwise. Since we require  $r$  to be monotonically increasing,  $r_d$  and  $r_e$  must have this property as well. Furthermore, to avoid the creation of spurious discontinuities, we constrain  $r_d$  and  $r_e$  to be continuous by requiring that  $r_d(g \pm \sigma) = r_e(g \pm \sigma)$ .

The function  $r_d$  modifies the fine-scale details by altering the oscillations around the value  $g$ . In our applications we process positive and negative details symmetrically, letting us write:

$$r_d(i, g, \sigma) = g + \text{sign}(i - g) \sigma f_d(|i - g| / \sigma) \quad (1)$$

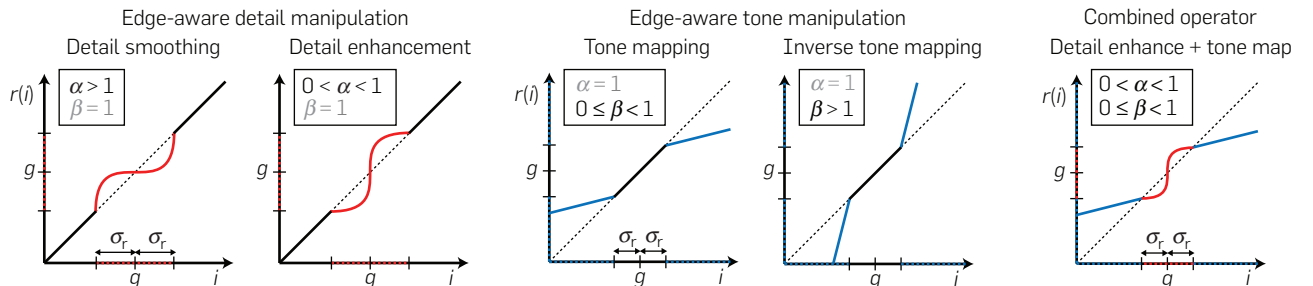
where  $f_d$  is a smooth function mapping from  $[0, 1]$  to  $[0, 1]$  that controls how details are modified. The advantage of this formulation is that it depends only on the amplitude of the detail  $|i - g|$  relative to the parameter  $\sigma$ , that is,  $|i - g| / \sigma = 1$  corresponds to a detail of maximum amplitude according to the user-defined parameter. Analogously,  $r_e$  is a function that modifies the amplitude of edges that we again formulate in a symmetric way:

$$r_e(i, g, \sigma) = g + \text{sign}(i - g) (f_e(|i - g| - \sigma) + \sigma) \quad (2)$$

where  $f_e$  is a smooth nonnegative function defined over  $[0, \infty)$ . In this formulation,  $r_e$  depends only on the amplitude above the user threshold  $\sigma$ , that is,  $|i - g| - \sigma$ . The function  $f_e$  controls how the edge amplitude is modified since an edge of amplitude  $\sigma + f_e(a)$  becomes an edge with amplitude  $\sigma + f_e(a)$ . For our previous 1D range compression example, clipping edges corresponds to  $f_e = 0$ , which limits the amplitude of all edges to  $\sigma$ . Useful specific choices for  $r_d$  and  $r_e$  are described in the next section and are illustrated in Figure 5.

The advantage of the functional forms defined in Equations (1) and (2) is that they ensure that  $r$  is continuous and increasing, and the design of a specific filter boils down to defining the two point-wise functions  $f_d$  and  $f_e$  that each

**Figure 5. Family of point-wise functions for edge-aware manipulation described in Sections 5.2 and 5.3. The parameters  $\alpha$  and  $\beta$  let us control how detail and tone are processed respectively. To compute a given Laplacian coefficient in the output, we filter the original image point-wise using a nonlinear function  $r(i)$  of the form shown. This remapping function is parametrized by the Gaussian pyramid coefficient  $g$ , describing the local image content, and a threshold  $\sigma$  used to distinguish fine details (red) from larger edges (blue).**



have clear roles:  $f_d$  controls the amplification or attenuation of details while  $f_e$  controls the amplification or attenuation of edges.

**Extension to color images.** To handle color, it is possible to treat only the luminance channel and reintroduce chrominance after image processing (Section 5.3). However, our approach extends naturally to color images as well, letting us deal directly with 3D vectors representing, for example, the RGB or CIE-Lab channels. Algorithm 1 still applies, and we need only to update  $r_d$  and  $r_e$ , using bold typeface to indicate vectors:

$$r_d(\mathbf{i}, \mathbf{g}, \sigma) = \mathbf{g} + \text{unit}(\mathbf{i} - \mathbf{g})\sigma f_d(\|\mathbf{i} - \mathbf{g}\|/\sigma) \quad (3a)$$

$$r_e(\mathbf{i}, \mathbf{g}, \sigma) = \mathbf{g} + \text{unit}(\mathbf{i} - \mathbf{g})[f_e(\|\mathbf{i} - \mathbf{g}\| - \sigma) + \sigma] \quad (3b)$$

with  $\text{unit}(\mathbf{v}) = \mathbf{v}/\|\mathbf{v}\|$  if  $\mathbf{v} \neq \mathbf{0}$  and  $\mathbf{0}$  otherwise. These equations define details as colors within a ball of radius  $\sigma$  centered at  $\mathbf{g}$  and edges as the colors outside it. They also do not change the roles of  $f_d$  and  $f_e$ , letting the same 1D functions that modify detail and edges in the gray-scale case be applied to generate similar effects in color. For images whose color channels are all equal, these formulas reduce to the gray-scale formulas of Equations (1) and (2).

## 5. APPLICATIONS AND RESULTS

We now demonstrate how to realize practical image processing applications using our approach and discuss implementation details. First we address edge-preserving smoothing and detail enhancement, followed by tone mapping and related tools. We validate our method with images used previously in the literature<sup>10–13, 27</sup> and demonstrate that our method produces artifact-free results.

### 5.1. Implementation

We use the pyramids defined by Burt and Adelson,<sup>6</sup> based on  $5 \times 5$  kernels. On a 2.26 GHz Intel Xeon CPU, we process a one-megapixel image in about a minute using a single thread. This can be halved by limiting the depth of the intermediate pyramid to at most five levels, by applying the remapping to level  $\max(0, \ell - 3)$  rather than always starting at the full-resolution image. This amounts to applying the remapping to a downsampled version of the image when processing coarse pyramid levels. The resulting images

are visually indistinguishable from the full-pyramid process with a PSNR on the order of 30 to 40 dB. While this performance is slower than previous work, our algorithm is highly data parallel and can easily exploit a multicore architecture. Using OpenMP, we obtain an  $8\times$  speed-up on an 8-core machine, bringing the running time down to 4 seconds.

### 5.2. Detail manipulation

To modify the details of an image, we define an S-shaped point-wise function as is classically used for the local manipulation of contrast. For this purpose, we use a power curve  $f_d(\Delta) = \Delta^\alpha$ , where  $\alpha > 0$  is a user-defined parameter. Values larger than 1 smooth the details out, while values smaller than 1 increase their contrast (Figures 5 and 6). To restrict our attention to the details of an image, we set the edge-modifying function to the identity  $f_e(a) = a$ .

In the context of detail manipulation, the parameter  $\sigma$  controls how at what magnitude signal variations should be considered edges and therefore be preserved. Large values allow the filter to alter larger portions of the signal and yield larger visual changes (Figure 7). In its basic form, detail manipulation is applied at all scales, but one can also control which scales are affected by limiting processing to a subset of the pyramid levels (Figure 6c, d, e). While this control is discrete, the changes are gradual, and one can interpolate between the results from two subsets of levels if continuous control is desired. Our results from Figures 6 and 7 are comparable to results of Farbmán et al.<sup>11</sup>; however, we do not require the complex machinery of a multiresolution preconditioned conjugate gradient solver. Note that our particular extension to color images allows us to boost the color contrast as well (Figures 6, 7, and 8).

**Reducing noise amplification.** As in other techniques for texture enhancement, increasing the contrast of the details may make noise and artifacts from lossy image compression more visible. We mitigate this issue by limiting the smallest  $\Delta$  amplified. In our implementation, when  $\alpha < 1$ , we compute  $f_d(\Delta) = \tau\Delta^\alpha + (1 - \tau)\Delta$ , where  $\tau$  is a smooth step function equal to 0 if  $\Delta$  is less than 1% of the maximum intensity, 1 if it is more than 2%, with a smooth transition in between. All the results in this paper and supplemental material are computed with this function.

**Figure 6. Smoothing and enhancement of detail, while preserving edges ( $\sigma = 0.3$ ). Processing only a subset of the levels controls the frequency of the details that are manipulated (c, d, e). The images have been cropped to make the flower bigger and its details more visible.**



### 5.3. Tone manipulation

Our approach can also be used for reducing the intensity range of a high-dynamic-range (HDR) image, according to the standard tone mapping strategy of compressing the large-scale variations while preserving (or enhancing) the details.<sup>35</sup> In our framework, we manipulate large-scale variations by defining a point-wise function modifying the edge amplitude,  $f_c(a) = \beta a$ , where  $\beta \geq 0$  is a user-defined parameter (Figure 5).

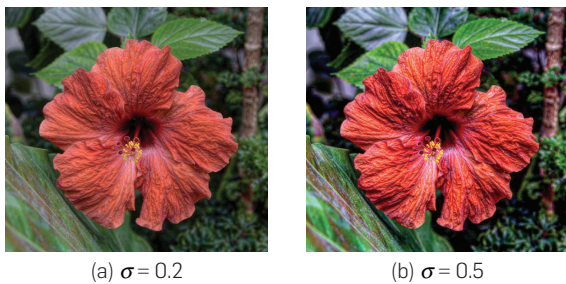
In our implementation of tone manipulation, we process the image intensity channel only and keep the color unchanged.<sup>10</sup> We compute an intensity image  $I_i = \frac{1}{61}(20I_r + 40I_g + I_b)$  and color ratios  $(\rho_r, \rho_g, \rho_b) = \frac{1}{I_i}(I_r, I_g, I_b)$ , where  $I_r, I_g,$  and  $I_b$  are the RGB channels. We apply our filter on the log intensities  $\log(I_i)$ ,<sup>35</sup> using the natural logarithm. For tone mapping, we set our filter with  $\alpha \leq 1$  so that details are preserved or enhanced, and  $\beta < 1$  so that edges are compressed. This produces new values  $\log(I_i')$ , which we must then map to the displayable range of  $[0, 1]$ . We remap the result  $\log(I_i')$  by first offsetting its values to make its maximum 0, then scaling them so that they cover a user-defined range.<sup>10, 21</sup> In our implementation, we estimate a robust maximum and minimum with the 99.5th and 0.5th percentiles, and we set the scale

factor so that the output dynamic range is 100:1 for the linear intensities. Finally, we multiply the intensity by the color ratios  $(\rho_r, \rho_g, \rho_b)$  to obtain the output RGB channels, then gamma correct with an exponent of 1/2.2 for display. We found that fixing the output dynamic range not only makes it easy to achieve a consistent look but also constrains the system. As a result, the  $\sigma$  and  $\beta$  parameters have similar effects, both controlling the balance between local and global contrast in the rendered image (Figure 9). From a practical standpoint, we advise keeping  $\sigma$  fixed and varying the slope  $\beta$  between 0, where the local contrast is responsible for most of the dynamic range, and 1, where the global contrast dominates. Unless otherwise specified, we use  $\sigma = \log(2.5)$ , which gave consistently good results in our experiments. Since we work in the log domain, this value corresponds to a ratio between pixel intensities. It does not depend on the dynamic range of the scene, and assumes only that the input HDR image measures radiance up to scale.

Our tone mapping operator builds upon standard elements from previous work that could be substituted for others. For instance, one could instead use a sigmoid to remap the intensities to the display range<sup>30</sup> or use a different color management method (e.g., Mantiuk et al.<sup>24</sup>). Also, we did not apply any additional “beautifying curve” or increased saturation as is commonly done in photo editing software. Our approach produces a clean output image that can be post-processed in this way if desired.

Range compression is a good test case to demonstrate the abilities of our pyramid-based filters because of the large modification involved. For high compression, even subtle inaccuracies can become visible, especially at high-contrast edges. In our experiments, we did not observe aliasing or oversharpening artifacts even on cases where other methods suffer from them (Figures 10 and 11). We also stress-tested our operator by producing

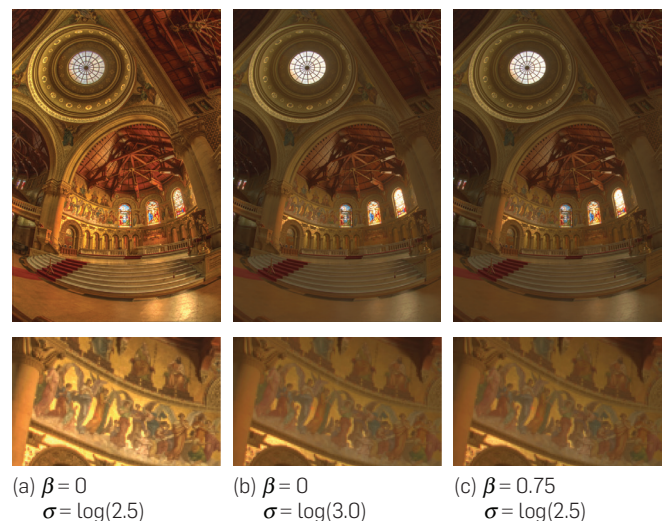
**Figure 7. Effect of the  $\sigma$  parameter for detail enhancement ( $\alpha = 0.25$ ). Same input as Figure 6.**



**Figure 8. Filtering only the luminance (b) preserves the original colors in (a), while filtering the RGB channels (c) also modifies the color contrast ( $\alpha = 0.25, \beta = 1, \sigma = 0.4$ ).**



**Figure 9.  $\beta$  and  $\sigma$  have similar effects on tone mapping results, they control the balance between global and local contrast.  $\alpha$  is set to 1 in all three images.**



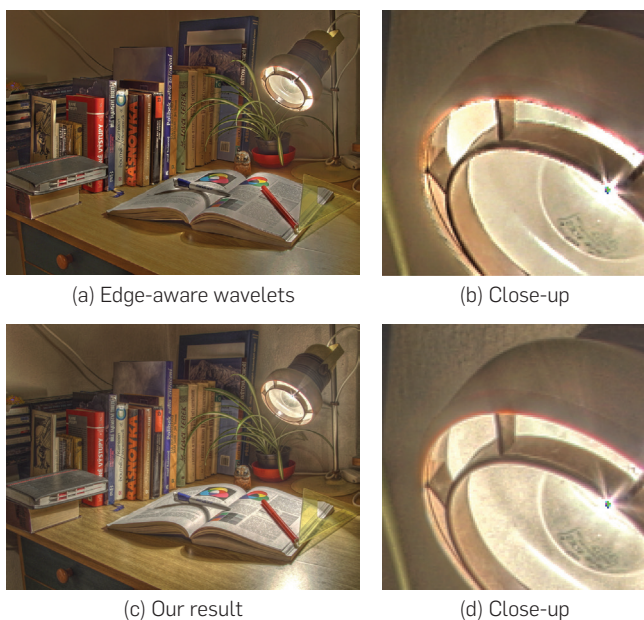


results with a low global contrast ( $\beta = 0$ ) and high local details ( $\alpha = 0.25$ ). In general, the results produced by our method did not exhibit any particular problems (Figure 12). We compare exaggerated renditions of our method with Farbman et al.<sup>11</sup> and Li et al.<sup>21</sup> Our method produces consistent results without halos, whereas the other methods either create halos or fail to exaggerate detail (Figure 13).

One typical difficulty we encountered is that sometimes the sky interacts with other elements to form high-frequency textures that undesirably get amplified by our detail-enhancing filter (Figures 8b and 14). Such “misinterpretation” is common to all low-level filters without semantic understanding of the scene, and typically requires user feedback to correct.<sup>22</sup>

We also experimented with inverse tone mapping, using slope values  $\beta$  larger than 1 to increase the dynamic range of a normal image. Since we operate on log intensities, roughly speaking, the linear dynamic range gets exponentiated by  $\beta$ . Applying our tone-mapping operator on these range-expanded results gives images close to the originals, typically with a PSNR between 25 and 30 dB for  $\beta = 2.5$ . This shows that our inverse tone mapping preserves the image content well. While a full-scale study on an HDR monitor is beyond the scope of this paper, we believe that our simple approach can complement other relevant techniques (e.g., Masia et al.<sup>25</sup>). Sample HDR results are provided in supplemental material.

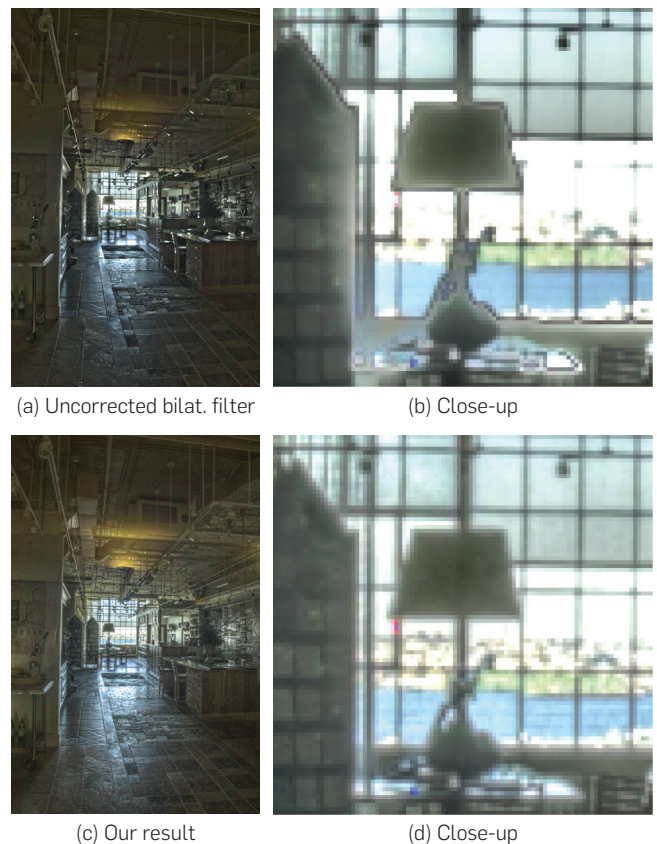
**Figure 10.** The extreme contrast near the light bulb is particularly challenging. Images (a) and (b) are reproduced from Fattal.<sup>12</sup> The edge-aware wavelets suffer from aliasing and generate an irregular edge (b). In comparison, our approach (d) produces a clean edge. We set our method to approximately achieve the same level of details ( $\sigma = \log(3.5)$ ,  $\alpha = 0.5$ ,  $\beta = 0$ ).



### 5.4. Discussion

While our method can fail in the presence of excessive noise or when extreme parameter settings are used (e.g., the *lenna* picture in supplemental material has a high level of noise), we found that our filters are very robust and behave well over a broad range of settings. Figure 15 shows a variety of parameters values applied to the same image and the results are consistently satisfying, high-quality, and halo-free; many more such examples are provided in supplemental material. While the goal of edge-aware processing can be ill-defined, the results that we obtain show that our approach allows us to realize many edge-aware effects with intuitive parameters and a simple implementation. The current shortcoming of our approach is its running time. We can mitigate this issue, thanks to the multiscale nature of our algorithm, allowing us to generate quick previews that are faithful to the full-resolution results (Figure 16). Furthermore, the algorithm is highly parallelizable and should lend itself to a fast GPU implementation. Beyond these practical aspects, our main contribution is a better characterization of the multiscale properties of images.

**Figure 11.** The bilateral filter sometimes oversharpens edges, which can lead to artifacts (b). We used code provided by Paris and Durand<sup>26</sup> and multiplied the detail layer by 2.5 to generate these results. Although such artifacts can be fixed in postprocessing, this introduces more complexity to the system and requires new parameters. Our approach produces clean edges directly (d). We set our method to achieve approximately the same visual result ( $\sigma = \log(2.5)$ ,  $\alpha = 0.5$ ,  $\beta = 0$ ).





**Figure 12.** We stressed our approach by applying a strong range compression coupled with a large detail increase ( $\alpha = 0.25$ ,  $\beta = 0$ ,  $\sigma = \log(2.5)$ ). The results are dominated by local contrast and are reminiscent of the popular, exaggerated “HDR look” but without the unsightly halos associated with it. In terms of image quality, our results remain artifact-free in most cases. We explore further parameter variations in the supplemental material.



**Figure 13.** We compare exaggerated, tone-mapped renditions of an HDR image. The wavelet-based method by Li et al.<sup>21</sup> is best suited for neutral renditions and generates halos when one increases the level of detail (a). The multiscale method by Farbman et al.<sup>11</sup> performs better and produces satisfying results for intermediate levels of detail (b), but halos and edge artifacts sometimes appear for a larger increase, as in this image for instance; see the edge of the white square on the blue book cover and the edge of the open book (c). In comparison, our approach achieves highly detailed renditions without artifacts (d). These results as well as many others may be better seen in the supplemental material.

(a) Li et al.<sup>21</sup> (detailed rendition using parameters suggested by the authors)

(b) Farbman et al.<sup>11</sup> (detailed rendition using parameters suggested by the authors)

(c) Farbman et al.<sup>11</sup> (exaggerated rendition using parameters suggested by the authors)

(d) Our result with exaggerated details ( $\alpha = 0.25$ ,  $\beta = 0$ )





Many problems related to photo editing are grounded in these properties of images and we believe that a better understanding can have benefits beyond the applications demonstrated in this paper.

## 6. CONCLUSION

**Link to recent work.** We first presented this work at the ACM SIGGRAPH conference in 2011. The main difference

**Figure 14.** Our approach is purely signal-based and its ignorance of scene semantics can lead to artifacts. For a large increase in local contrast (b), at a level similar to Figure 12, the sky gets locally darker behind clouds, because it forms a blue-white texture amplified by our filter. Our result for this example is good elsewhere, and this issue does not appear with a more classical rendition (a).



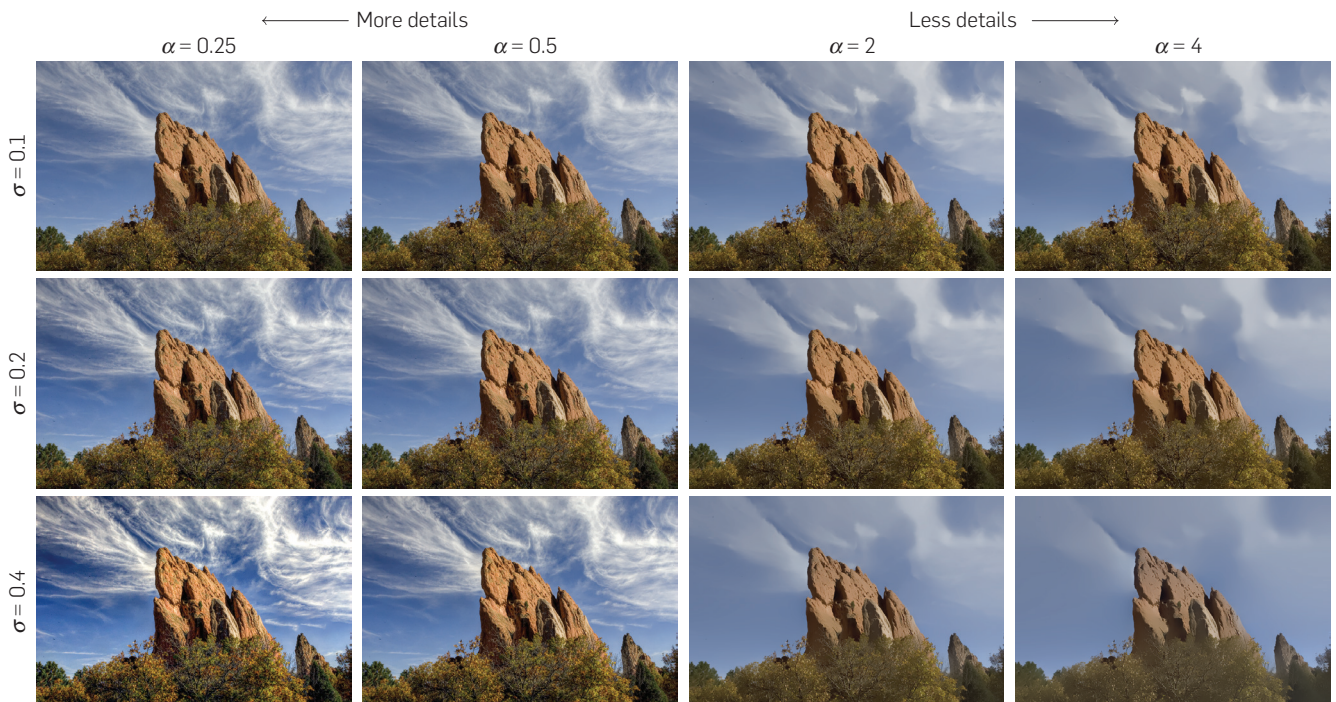
(a) No detail increase ( $\alpha = 1$ )      (b) Detail increased ( $\alpha = 0.25$ )

with our original article is Section 3 that now focuses on qualitative properties of edges. A formal discussion of these properties can be found in Paris et al.<sup>28</sup> Since then, we also extended this work with a fast algorithm that makes Local Laplacian Filters practical, an analysis that shows their relationship to the Bilateral Filter, an application to the transfer of gradient histograms applied to photographic style transfer, and additional comparisons with existing techniques such as the Guided Filter.<sup>17</sup> These results are described in Aubry et al.<sup>2</sup>

Although Local Laplacian Filters can reduce image details, Xu et al.<sup>39, 40</sup> have shown that they do not fully remove them and have proposed filters that completely suppress details for applications such as cartoon rendering and mosaic texture removal. By addressing the extreme detail removal problem, this work is complementary to Local Laplacian Filters that perform well at extreme detail increase. Hadwiger et al.<sup>16</sup> have introduced a dedicated data structure to process very large images efficiently and have demonstrated its application to Local Laplacian Filtering.

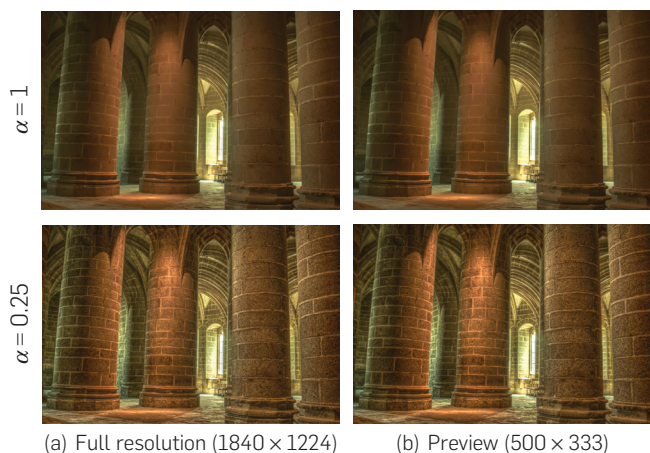
**Closing note.** We have presented a new technique for edge-aware image processing based solely on the Laplacian pyramid. It is conceptually simple, allows for a wide range of edge-aware filters, and consistently produces artifact-free images. We demonstrate high-quality results over a large variety of images and parameter settings, confirming the method's robustness. Our results open new perspectives on multiscale image analysis and editing since Laplacian pyramids were previously considered as ill-suited for manipulating edges. Given the wide use of pyramids and the need

**Figure 15.** Our filter to enhance and reduce details covers a large space of possible outputs without creating halos.






**Figure 16. Our approach generates faithful previews when applied to a low-resolution version of an image ( $\beta = 0$ ,  $\sigma = \log(2.5)$ ).**



for edge-aware processing, we believe our new insights can have a broad impact in the domain of image editing and its related applications.

### Acknowledgments

We thank Ted Adelson, Bill Freeman, and Frédo Durand for inspiring discussions and encouragement; Alan Erickson for the *Orion* image; and the anonymous reviewers for their constructive comments. This work was supported in part by an NSERC Postdoctoral Fellowship, the Quanta T-Party, NGA NEGI-1582-04-0004, MURI Grant N00014-06-1-0734, and gifts from Microsoft, Google, and Adobe. We thank Farbman et al. and Li et al. for their help with comparisons. 

### References

- Aubert, G. and Kornprobst, P. *Mathematical Problems in Image Processing: Partial Differential Equations and the Calculus of Variations*. Vol. 147 of Applied Mathematical Sciences. Springer, 2002.
- Aubry, M., Paris, S., Hasinoff, S.W., Kautz, J., and Durand, F. *Fast and Robust Pyramid-based Image Processing*. Tech. Rep. MIT-CSAIL-TR-2011-049. MIT, 2011.
- Bae, S., Paris, S., and Durand, F. Two-scale tone management for photographic look. *ACM Trans. Graph. (Proc. SIGGRAPH)* 25, 3 (2006), 637–645.
- Bhat, P., Zitnick, C.L., Cohen, M., and Curless, B. Gradientshop: A gradient-domain optimization framework for image and video filtering. *ACM Trans. Graph.* 29 (2010), 2.
- Buades, A., Coll, B., and Morel, J.-M. The staircasing effect in neighborhood filters and its solution. *IEEE Trans. Image Process.* 15 (2006), 6.
- Burt, P.J. and Adelson, E.H. The Laplacian pyramid as a compact image code. *IEEE Trans. Commun.* 31 (1983), 4.
- Chen, J., Paris, S., and Durand, F. Real-time edge-aware image processing with the bilateral grid. *ACM Trans. Graph. (Proc. SIGGRAPH)* 26 (2007), 3.
- Criminisi, A., Sharp, T., Rother, C., and Perez, P. Geodesic image and video editing. *ACM Trans. Graph.* 29 (2010), 5.
- Dippel, S., Stahl, M., Wiemker, R., and Blaffert, T. Multiscale contrast enhancement for radiographies: Laplacian pyramid versus fast wavelet transform. *IEEE Trans. Med. Imaging* 21 (2002), 4.
- Durand, F. and Dorsey, J. Fast bilateral filtering for the display of high-dynamic-range images. *ACM Trans. Graph. (Proc. SIGGRAPH)* 21 (2002), 3.
- Farbman, Z., Fattal, R., Lischinski, D., and Szeliski, R. Edge-preserving decompositions for multi-scale tone and detail manipulation. *ACM Trans. Graph. (Proc. SIGGRAPH)* 27 (2008), 3.
- Fattal, R. Edge-avoiding wavelets and their applications. *ACM Trans. Graph. (Proc. SIGGRAPH)* 28 (2009), 3.
- Fattal, R., Lischinski, D., and Werman, M. Gradient domain high dynamic range compression. *ACM Trans. Graph. (Proc. SIGGRAPH)* 21 (2002), 3.

- Fattal, R., Agrawala, M., and Rusinkiewicz, S. Multiscale shape and detail enhancement from multi-light image collections. *ACM Trans. Graph. (Proc. SIGGRAPH)* 26 (2007), 3.
- Fattal, R., Carroll, R., and Agrawala, M. Edge-based image coarsening. *ACM Trans. Graph.* 29 (2009), 1.
- Hadwiger, M., Sicat, R., Beyer, J., Krüger, J., and Möller, T. Sparse PDF maps for non-linear multiresolution image operations. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 31 (2012), 5.
- He, K., Sun, J., and Tang, X. Guided image filtering. In *Proceedings of European Conference on Computer Vision (Proc. ECCV)* (2010).
- Heeger, D.J. and Bergen, J.R. Pyramid-based texture analysis/synthesis. In *Proceedings of the ACM SIGGRAPH Conference (Proc. SIGGRAPH)* (1995).
- Kass, M. and Solomon, J. Smoothed local histogram filters. *ACM Trans. Graph. (Proc. SIGGRAPH)* 29 (2010), 3.
- Kimmel, R. *Numerical Geometry of Images: Theory, Algorithms, and Applications*. Springer, 2003.
- Li, Y., Sharan, L., and Adelson, E.H. Compressing and companding high dynamic range images with subband architectures. *ACM Trans. Graph. (Proc. SIGGRAPH)* 24 (2005), 3.
- Lischinski, D., Farbman, Z., Uyttendaele, M., and Szeliski, R. Interactive local adjustment of tonal values. *ACM Trans. Graph. (Proc. SIGGRAPH)* 25 (2006), 3.
- Mantiuk, R., Myszkowski, K., and Seidel, H.-P. A perceptual framework for contrast processing of high dynamic range images. *ACM Trans. Appl. Percept.* 3 (2006), 3.
- Mantiuk, R., Mantiuk, R., Tomaszewska, A., and Heidrich, W. Color correction for tone mapping. *Comput. Graph. Forum (Proc. Eurographics)* 28 (2009), 2.
- Masia, B., Agustín, S., Fleming, R.W., Sorkine, O., and Gutierrez, D. Evaluation of reverse tone mapping through varying exposure conditions. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 28 (2009), 5.
- Paris, S. and Durand, F. Tone-mapping code. <http://people.csail.mit.edu/sparis/code/src/tonemapping.zip>.
- Paris, S., Kornprobst, P., Tumblin, J., and Durand, F. Bilateral filtering: Theory and applications. *Found. Trends Comput. Graph. Vision* 4, 1 (2009), 1–74.
- Paris, S., Hasinoff, S.W., and Kautz, J. Local Laplacian Filters: Edge-aware image processing with a Laplacian pyramid. *ACM Trans. Graph. (Proc. SIGGRAPH)* 30 (2011), 4.
- Perona, P. and Malik, J. Scale-space and edge detection using anisotropic diffusion. *IEEE Trans. Pattern Anal. Mach. Intell.* 12 (1990), 7.
- Reinhard, E., Stark, M., Shirley, P., and Ferwerda, J. Photographic tone reproduction for digital images. *ACM Trans. Graph. (Proc. SIGGRAPH)* 21 (2002), 3.
- Subr, K., Soler, C., and Durand, F. Edge-preserving multiscale image decomposition based on local extrema. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 28 (2009), 5.
- Sunkavalli, K., Johnson, M.K., Matusik, W., and Pfister, H. Multiscale image harmonization. *ACM Trans. Graph. (Proc. SIGGRAPH)* 29 (2010), 3.
- Szeliski, R. Locally adapted hierarchical basis preconditioning. *ACM Trans. Graph. (Proc. SIGGRAPH)* 25 (2006), 3.
- Tomasi, C. and Manduchi, R. Bilateral filtering for gray and color images. In *Proceedings of the IEEE International Conference on Computer Vision (Bombay, India, 1998)*.
- Tumblin, J. and Turk, G. Low curvature image simplifiers (LCIS): A boundary hierarchy for detail-preserving contrast reduction. In *Proc. SIGGRAPH* (1999).
- Vuylsteke, P. and Schoeters, E.P. Multiscale image contrast amplification (MUSICA). In *Proc. SPIE*, Volume 2167 (1994).
- Witkin, A. Scale-space filtering. In *Proceedings of the International Joint Conference on Artificial Intelligence*, Volume 2 (Karlsruhe, Federal Republic of Germany (a.k.a. West Germany), 1983).
- Witkin, A., Terzopoulos, D., and Kass, M. Signal matching through scale space. *Int. J. Comput. Vision* 1 (1987), 2.
- Xu, L., Lu, C., Xu, Y., and Jia, J. Image smoothing via LO gradient minimization. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 30 (2011), 5.
- Xu, L., Yan, Q., Xia, Y., and Jia, J. Structure extraction from texture via relative total variation. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 31 (2012), 5.

Sylvain Paris (sparis@adobe.com), Adobe Research.

Jan Kautz (j.kautz@ucl.ac.uk), University College London.

Samuel W. Hasinoff (hasinoff@google.com), Google Inc.

Images credits: Martin Čadik, Paul Debevec, Frédéric Drago, Frédo Durand, Mark Fairchild, Dani Lischinski, Byong Mok Oh, Erik Reinhard, and Gregory J. Ward.



Watch the authors discuss this work in this exclusive Communications video.