Smoothed Local Histogram Filters

Michael Kass
Pixar Animation Studios

Justin Solomon
Pixar Animation Studios and Stanford University

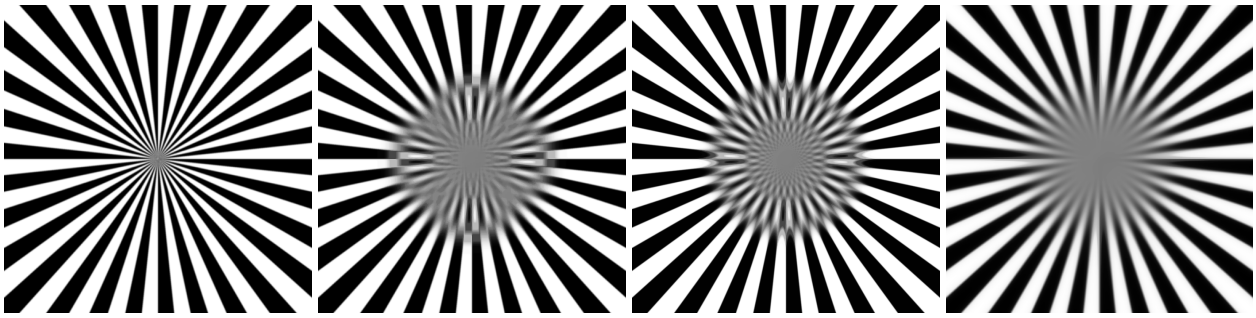


Figure 1: From left to right: Pinwheel image, Photoshop Median Filter, Isotropic Equal Weight Median, Our Median Filter.

Abstract

Local image histograms contain a great deal of information useful for applications in computer graphics, computer vision and computational photography. Making use of that information has been challenging because of the expense of computing histogram properties over large neighborhoods. Efficient algorithms exist for some specific computations like the bilateral filter, but not others. Here, we present an efficient and practical method for computing accurate derivatives and integrals of locally-weighted histograms over large neighborhoods. The method allows us to compute the location, height, width and integral of all local histogram modes at interactive rates. Among other things, it enables the first constant-time isotropic median filter, robust isotropic image morphology operators, an efficient "dominant mode" filter and a non-iterative alternative to the mean shift. In addition, we present a method to combat the over-sharpening that is typical of histogram-based edge-preserving smoothing. This post-processing step should make histogram-based filters not only fast and efficient, but also suitable for a variety of new applications.

CR Categories: I.3.3 [Computer Graphics]—Picture/Image Generation; I.4.3 [Image Processing and Computer Vision]—enhancement, filtering

Keywords: Mode filter, Bilateral filter, Histogram

1 Introduction

A variety of popular image filters can be expressed as functions of the local histogram. The median filter, for example, is the 50% point of the histogram. The gray-scale mathematical morphology operations of dilation and erosion are the 100% point (max) and 0% point (min) of the histogram. The bilateral filter can be expressed as a simple ratio of linear functions of the histogram. Mean-shift, histogram equalization and a collection of other valuable filters also have simple expressions in terms of local histograms.

All histogram-based computations face a computational challenge when dealing with large neighborhoods. In a naïve implementation, every pixel in a neighborhood has to be examined and sorted in order to compute a single output value. For sizeable neighborhoods, this is an expensive proposition.

A collection of acceleration techniques have been developed for specific filters, but all of them have important limitations either on the types of filters to which they apply or the shape and weighting of the local histogram neighborhood. Here we introduce a general method for accelerating a wide range of local histogram-based computations using center-weighted isotropic histogram neighborhoods. The method allows us to compute values of the smoothed local histogram, its derivatives and its integrals with computational expense independent of the neighborhood size. In addition to allowing simple, efficient and improved implementations of well-known filters, it also enables some interesting new histogram-based computations.

An important application of histogram-based filters is in edge-preserving smoothing. Often, such filters are used to decompose images into the sum of a base layer and a detail layer for further processing. For this purpose, an important limitation is that histogram-based filters often increase the sharpness of the edges they preserve. When trying to decompose an image into a series of layers for purposes like tone mapping or contrast enhancement, the over-sharpening of edges leads to halos, gradient reversals or other artifacts in the output. Here we show how to mitigate this effect and thereby increase the range of applicability for the histogram-based filters our method accelerates.

2 Previous Work

Probably the first histogram-related image filter to receive attention in image processing was the median filter, a filter that at each point outputs the median intensity of a local neighborhood of pixels. For $n \times n$ regions, the straightforward algorithm takes $O(n^2 log(n))$ operations per pixel if a full sort is done, or $O(n^2)$ if the sort is binned. Huang [1975] showed that this filter can be accelerated for

ACM Reference Format
Kass, M., Solomon, J. 2010. Smoothed Local Histogram Filters. *ACM Trans. Graph. 29*, 4, Article 100 (July 2010), 10 pages. DOI = 10.1145/1778765.1778837 http://doi.acm.org/10.1145/1778765.1778837.

rectangular neighborhoods. Given a histogram for an $n \times n$ region centered on one pixel, the histogram for an $n \times n$ region centered on an adjacent pixel can be calculated incrementally in just $O(n)$ operations by taking advantage of the overlap. Weiss [2006] later improved this computation to $O(\log(n))$, and Porikli [2005] and Perreault and Hebert [2007] showed how to do the computation in constant time.

Unfortunately, none of these fast median algorithms is isotropic, and none of them provides weights that drop off smoothly from the center of the neighborhood. Figure 1 illustrates why this is important. The rectangular equally-weighted neighborhood used by Adobe Photoshop's median filter causes severe artifacts on this test input. Perreault and Hebert [2007] recognize this problem and give other examples of the artifacts that arise from rectangular neighborhoods. They suggest improving their algorithm with a modification that allows it to produce equally-weighted octagonal neighborhoods. Yet Figure 1(c) shows that this is insufficient. Even going all the way to a circular equally-weighted neighborhood leaves unacceptable artifacts. The smooth center-weighted neighborhood computation we present here avoids the problem.

Van Herk [1992] developed a constant-time algorithm for computing the minimum or maximum of rectangular or octagonal regions. This method of implementing the image morphology operators of dilation and erosion is attractive for its speed, but the algorithm suffers from the same limitations as the rectangular or square algorithms for the median filter. The result will not be isotropic, and the uniform weighting means that the effect of a single noise pixel will extend to the edge of a neighborhood and then fall off abruptly. In our approach, the minimum and maximum can be replaced with low and high percentiles of the center-weighted histogram, allowing an isotropic, robust and smooth result.

Felsberg et al. [2006] and Paris and Durand [2006] both present acceleration methods for particular histogram-based image smoothing filters using look-up tables followed by fast linear low-pass filters. Felsberg et al. use B-spline look-up tables for a robust filter they call *channel smoothing*. Paris and Durand use Gaussian look-up tables to compute the bilateral filter. Their method was further refined by Chen et al. [2007] and accelerated by Porikli [2008]. Our technique generalizes all these approaches to enable acceleration of a wider collection of histogram-based filters including the median filter and morphology operators as well as some new filters we introduce here. In addition to low-pass look-up tables, we also use look-up tables that are integrals and derivatives of low-pass filters. As a result, we can quickly and accurately compute integrals and derivatives of the local histogram, identify all its peaks and valleys, and evaluate the size of the corresponding populations.

Comaniciu and Meer [2002] argue for the importance of finding modes in the local histogram. They present an iterative algorithm called the *mean shift* based on the work of Fukunaga and Hostetler [1975], which converges at each point to the histogram mode closest to the input pixel value. Paris and Durand [2007] have shown how to use fast Gaussian filtering to accelerate this iteration. With our approach, we can go further and remove the need for iteration. Since we have information about the derivative of the smoothed histogram, we directly locate the nearest mode by evaluating the point at which the derivative of the smoothed histogram vanishes.

In the presence of noise, the mode closest to a given pixel may not be the best choice. Felsberg et al. [2006], in formulating their channel smoothing filter, advocate using the mode with the highest histogram density to achieve robustness. This approach has a severe weakness. Under ordinary circumstances, the mode corresponding to the desired signal will have a much larger histogram population than the mode corresponding to the noise. The relative histogram amplitudes, however, depend on the respective variances. Low variance noise can have a higher histogram amplitude than the signal, even when the corresponding population is much smaller. This is illustrated in Figure 5(d) where channel smoothing actually makes the noise worse, while our dominant mode filter (Figure 5(f)) makes the noise disappear.

Some of the methods discussed above generalize well to histograms of multi-dimensional quantities such as RGB color vectors. For example, the mean shift and the integral histogram method of Porikli [2005] have straightforward generalizations to arbitrary numbers of dimensions. The method of Adams et al. [2009] is particularly noteworthy for high numbers of dimensions. Here, however, we will restrict ourselves to problems involving histograms of one-dimensional quantities. There are a variety of ways to use one-dimensional filters to process color images. Paris et al. [2007] discuss the issues and possible approaches. For illustration, in all the examples of this paper we process color images by converting RGB to HSV space, filtering the V channel while leaving H and S unchanged, and then converting back to RGB. Depending on the application, other color spaces, such as CIELAB space may be superior.

3 Histogram Properties

The traditional histogram sorts data into a collection of bins, indicating the frequency with which the data falls into each bin. While this is convenient for exploratory data analysis, the discrete quantization of data into bins is artificial. From a signal processing point of view, we are better off examining a smoothed histogram. If $I_{\mathbf{p}}$ is the image intensity at a point $\mathbf{p}$, then the smoothed histogram of a neighborhood around the point can be written

$$f_{\mathbf{p}}(s) = \frac{1}{n} \sum_{i=1}^{n} K(I_{\mathbf{q}_i} - s) \tag{1}$$

where K is a smoothing kernel that sums to one, n is the number of points in the neighborhood, and $\mathbf{q}_i$ ranges over the pixels in the neighborhood of $\mathbf{p}$. Note that if K is a unit-area box function and $f(s)$ is sampled appropriately, this reduces to the traditional binned histogram.

Equation 1 has another important interpretation. If we consider the pixel intensity values to be samples from a random variable, then f is a popular estimate of the underlying probability density function. This method of estimating the probability density is most generally known as kernel density estimation, while in the pattern recognition literature, it has come to be known as the Parzen window technique [Duda and Hart 1973; Parzen 1962].

There are a variety of possible smoothing kernels one could consider for K. In many cases, we are concerned with extrema in f, so we prefer to use a kernel that can never introduce new extrema in the course of smoothing. The unique kernel with this property is the Gaussian [Babaud et al. 1986] which we will use for all our examples.

Whether we consider f to be a smoothed histogram or a probability density estimate, it is valuable to have the influence of nearby pixels fall off gradually with distance. For this, we introduce an all-positive weighting function W with unit sum. Then the smoothed, locally-weighted histogram (or density estimate) can be written

$$\hat{f}_{\mathbf{p}}(s) = \sum_i K(I_{\mathbf{q}_i} - s) W(\mathbf{p} - \mathbf{q}_i) \tag{2}$$

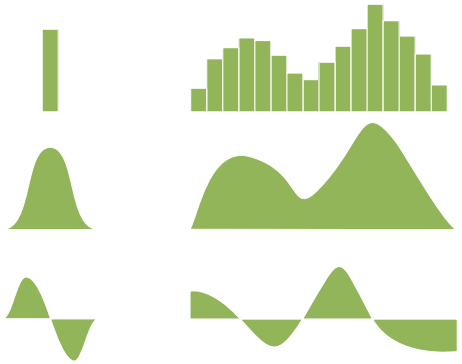


Figure 2: Left: Look-up table. Right: Image mapped through look-up table with different shifts. First row: Raw Histogram. Second Row: Smoothed Histogram. Third Row: Derivative of Smoothed Histogram.

or as the 2D spatial convolution

$$\hat{f}_{\mathbf{p}}(s) = K(I_{\mathbf{p}} - s) * W \tag{3}$$

When W and K are both Gaussians, this is equivalent to an expression used by Paris and Durand [2006] to normalize the bilateral filter. When W is a Gaussian and K is a quadratic B-spline, this is equivalent to an expression used by Felsberg et al. [2006] for channel smoothing. Both choices for K are reasonable. As Paris and Durand observed, K determines the frequency content of $\hat{f}_{\mathbf{p}}(s)$ as a function of s and W determines its spatial frequency content. Both are low-pass filters, so it is a simple matter to compute the necessary sampling rates.

An advantage of using a Gaussian for W is that Gaussian convolutions can be performed in constant time per input pixel, independent of the neighborhood size. At least two different practical constant-time methods are available. Burt [1983] provides a hierarchical method which achieves constant-time filtering by downsampling coarse levels of the pyramid. Deriche [1993] provides a constant-time method using infinite impulse response (IIR) recursive filters. We use the Deriche approach for our CPU implementation and vendor-supplied routines for fast Gaussian convolutions in our GPU implementation. Other choices for W are possible at additional computational cost: a convolution with an arbitrary kernel W can be computed at a cost of $O(\log(n))$ operations per output pixel using a 2D FFT. All the examples in this paper were computed with a Gaussian for W.

The smoothed local histogram $\hat{f}_{\mathbf{p}}(s)$ tells us a great deal about the image neighborhood near $\mathbf{p}$. If it has a single peak or mode, then it is likely that the pixels in the neighborhood are members of the same population. On the other hand, if the histogram has multiple modes, then the neighborhood probably contains pixels from two or more distinct populations. For stylization, noise reduction, segmentation and other purposes, we would like to be able to identify and characterize the modes. In particular, we would like to identify the number of modes, their values, their widths and the percentage of the population contained within each mode.

At a histogram mode, $\partial \hat{f}(s)/\partial s$ will vanish. Since only K depends on s, we can write the derivative of the histogram as

$$D_{\mathbf{p}}(s) = \frac{\partial \hat{f}_{\mathbf{p}}(s)}{\partial s} = -K'(I_{\mathbf{p}} - s) * W. \tag{4}$$

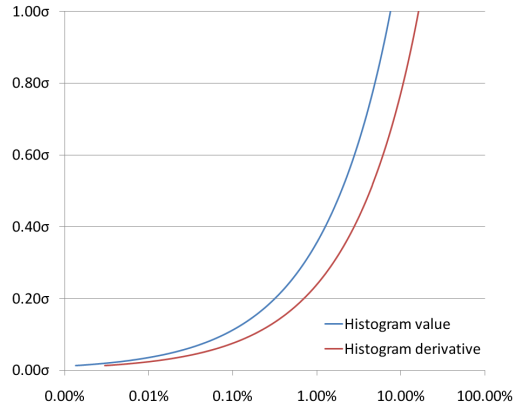


Figure 3: Error Graph

Since K is a low-pass kernel, its derivative K' will also be band limited. As a result, we can sample the derivative of the histogram $D_{\mathbf{p}}(s)$ at or above the Nyquist sampling rate without loss of information. Let $s_i, 1 < i < m$, be a set of samples of s over the relevant range at the Nyquist rate of K' or above. Then all the histogram modes can be identified from the functions

$$D_i(\mathbf{p}) = -K'(I_{\mathbf{p}} - s_i) * W. \tag{5}$$

Computing the functions $D_i(\mathbf{p})$ is straightforward and well suited to modern GPU hardware. For each i, we first compute a look-up table L_i that maps any intensity value $I_{\mathbf{p}}$ to $K'(s_i - I_{\mathbf{p}})$. We then map the input image I through the look-up table and convolve the result with the spatial kernel W to get the function D_i.

Strictly speaking, to guarantee accurate results at the Nyquist rate, sinc interpolation is necessary. However, by increasing the sampling rate sufficiently, we can ensure that linear interpolation in s is as accurate as desired. With sufficient sampling, we have a very simple algorithm to calculate the modes of $\hat{f}_{\mathbf{p}}$. At each point $\mathbf{p}$ we simply look for zero crossings in $D_i(\mathbf{p})$ as we increase i. If we find a negative-going zero crossing between $D_i(\mathbf{p})$ and $D_{i+1}(\mathbf{p})$, then there is a mode located approximately at

$$s = s_i + \frac{D_i(\mathbf{p})}{D_i(\mathbf{p}) - D_{i+1}(\mathbf{p})}(s_{i+1} - s_i). \tag{6}$$

Figure 2 illustrates the algorithm graphically. On the left side of the figure are a set of image look-up tables. The look-up tables are used with a variety of different offsets or horizontal shifts s_i. Mapping the image through the look-up table at a given offset and smoothing over the local neighborhood gives a sample of the function on the right.
The upper left shows a box-function look-up table. Mapping an image through this look-up table and then smoothing over a neighborhood gives a sample from the traditional histogram. The histogram is drawn as a discontinuous function because traditional histogram bins are non-overlapping.

Of course, if we consider the offset of the look-up table to be continuous rather than discrete, the result of mapping it over the image and smoothing over a neighborhood becomes continuous. This is illustrated in the second row of Figure 2. Spatially smoothing the output of the look-up table gives a sample of the smoothed, locally weighted histogram shown on the right of the second row, which corresponds to $\hat{f}_{\mathbf{p}}(s)$ in Equation 3.

To find extrema in $\hat{f}_{\mathbf{p}}(s)$, we use a look-up table shaped like the derivative of the kernel K as shown on the bottom left of Figure 2.

(a) Original Tractor Image

(b) Closest-mode Filtered

(c) Dominant-mode Filtered

Figure 4: Closest Mode vs. Dominant Mode

Mapping an image through this look-up table and smoothing gives a sample of the derivative of the smoothed histogram shown on the bottom right. Interpolating zero crossings as we change the offset s_i of our look-up tables allows us to identify all the extrema in the smoothed local histogram.

3.1 Sampling Rates

The amount of smoothing of the histogram and the amount of error that can be tolerated determine the number of look-up table samples required for all of our filters. Figure 3 shows the relationship for the case where K is a Gaussian of standard deviation σ and the error criterion is the root-mean-square difference between the ideal filter and the effective filter resulting from linear interpolation. Two curves are shown. The blue curve shows the rms error vs. sampling rate vs. for estimating the smoothed histogram $f_{\mathbf{p}}(s)$. The red curve shows the same trade-off for approximating $\partial f_{\mathbf{p}}(s)/\partial s$.

For example, if 5% rms interpolation error is tolerable in $\partial f_{\mathbf{p}}(s)/\partial s$, the centers of the look-up tables D_i should be no farther apart than $.54\sigma$. The error can be reduced to 1% if the centers are no farther apart than $.24\sigma$. Estimates of $f_{\mathbf{p}}(s)$ can use a lower sampling rate, as shown in the figure.

4 Filters

4.1 Closest-Mode Filter

Identifying all the modes in the smoothed local histogram allows us to compute the mode closest to the intensity of each input pixel. If we define the closest mode to be the mode one would reach by steepest ascent in the smoothed local histogram, then this is exactly equivalent to what Barash and Comaniciu [2004] call the "restricted mean shift." All we need to do is interpolate among the functions D_i to estimate $D(I_{\mathbf{p}})$, the derivative of the smoothed local histogram at $I_{\mathbf{p}}$. If the derivative is positive, we output the first mode greater than the pixel value. Otherwise we output the first mode less than the pixel value. Figure 4(b) shows the results. Since we compute the image without iteration and without any use of the mean-shift vector, we propose calling it the "closest-mode filter."

To compute Figure 4 the original image was first transformed into HSV space, then closest-mode filtered in the V channel and finally transformed back to RGB. The choice of HSV space for the filtering is to avoid slight misalignments that can arise in the positioning of edges in the different color channels, much as discussed by Paris and Durand [2006] with bilateral filtering.

Where the input image crosses a mode boundary, the output of the closest mode filter is discontinuous. When computed with a single sample per pixel, this can result in significant aliasing. To attenuate aliasing artifacts, Figure 4(b) has been antialiased with 3×3 oversampling. We compute the functions D_i once per pixel, but supersample the original image with bilinear interpolation. The closest modes for each of the image super-samples are averaged together to yield the value of the output pixel. Since most of the computation time is in doing the convolutions to evaluate the functions D_i, 3×3 antialiasing has a small impact on the total running time.

4.2 Median Filter

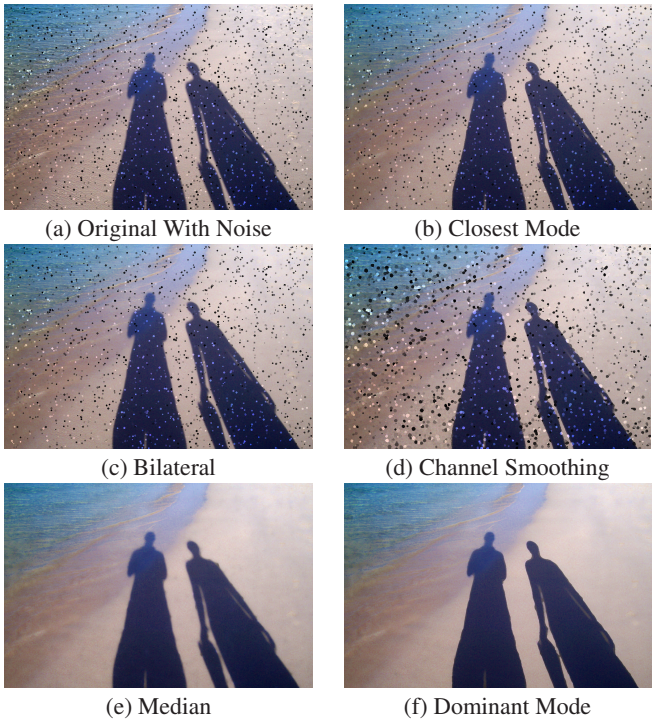
Edge-preserving filters like the closest-mode filter and the bilateral can be very valuable for noise reduction (e.g. [Dominguez et al. 2003; Paris et al. 2007]). In the presence of extreme noise, however, neither of these filters is as robust as a median filter. The reason is that both filters rely too much on the central pixel in each neighborhood. The closest-mode filter uses the value of the central pixel to choose a mode. If the central pixel is an outlier, the filter may choose an outlier mode. Similarly, the bilateral filter uses the central pixel to choose the weights of nearby pixels in its local average. If the central pixel is an outlier, the weights may be very inappropriate.

The histogram computations we have developed for mode finding are appropriate for computing the median filter as well. More generally, the technique we use for computing the functions D_i applies equally well to integrals or further derivatives of the smoothed local histogram $\hat{f}_{\mathbf{p}}(s)$. Let

$$C(s) = \int_{-\infty}^{s} K(u)du \tag{7}$$

If K is a Gaussian, then C will be a corresponding error function (erf). We can estimate the integrals of $\hat{f}_{\mathbf{p}}(s)$

$$R_{\mathbf{p}}(s) = \int_{-\infty}^{s} \hat{f}_{\mathbf{p}}(s)ds = C(I_{\mathbf{p}} - s) * W. \tag{8}$$



(a) Original With Noise (b) Closest Mode (c) Bilateral (d) Channel Smoothing (e) Median (f) Dominant Mode

Figure 5: Image corrupted with noise and then processed with histogram-based filters.

Figure 6: Top: Original. Bottom: After closing and opening.

using the functions

$$R_i(\mathbf{p}) = C(I_\mathbf{p} - s_i) * W \tag{9}$$

which sample R at a collection of intensities s_i. As with D_i, we can compute R_i by using a look-up table followed by a spatial convolution. We then interpolate among the R_i to find the value of s for which $R = 1/2$. This value of s is the median.

One can use more sophisticated interpolation schemes, but with adequate sampling, even linear interpolation is sufficient. C is much lower-pass than D, so the required number of samples s_i is substantially less than for D.

Figure 5 shows our median algorithm being used for noise reduction. An original image corrupted with noise (a) is then processed with different filters. The closest-mode filter (b) and the bilateral filter (c) fail to remove the noise. The median filter (e) removes the noise without introducing artifacts. As before, we have applied the filtering only to the V component of the HSV representations.

Our method of computing the median takes constant time per output sample, independent of the neighborhood size. Unlike any previous fast methods, it works with Gaussian-weighted neighborhoods to produce a center-weighted isotropic result. Figure 1 shows the impact of Gaussian weighting on an antialiased pinwheel image. The Photoshop result clearly shows anisotropic artifacts which are not present in our result. The anisotropy can be removed by using a circular equal-weight neighborhood as shown in the figure, but the artifacts remain. By using an isotropic weighting function that smoothly decays from the center, our filter avoids the artifacts.

The median computation requires only the look-up tables $I \rightarrow C(I_\mathbf{p} - s_i)$ which are lower-pass than K, so they can be computed at even lower sampling rates for the same accuracy. The mean-square error in this case depends on the integration interval, so it is difficult to summarize in a single figure of merit. In our experience, 15 samples of s_i suffice to provide good-quality median filter of 8-bit images using a Gaussian function K with standard deviation equal to the sample separation.

4.3 Morphology

Morphological operators [Haralick et al. 1987] are a class of popular image filters used for a wide range of image-processing applications. The fundamental morphological operators are *dilation*, defined as the maximum intensity within a region and *erosion* defined as the minimum intensity within a region. In the morphology literature, the region is traditionally referred to as the *structuring element*. When all samples in the neighborhood are equally weighted, dilation, erosion and median can be seen as examples of a wider class of filters known as *rank-order filters*. These filters output the ith largest of n samples in a neighborhood. With unequal weighting and smoothed histograms, the proper generalization is to *percentile filters*: filters that output a given percentile of the smoothed histogram.

When the percentile is 0% or 100% as in the traditional dilation and erosion operators, unequal neighborhood weighting does not affect the result. For any percentile in-between, however, the weighting does have an impact. If we modify the traditional erode and dilate operators to use the 5% and 95% points, we will have a very similar result to the traditional operators, but with added robustness against noise. High amplitude noise which affects less than 5% of the neighborhood will have a very small effect on the result. More severe noise will have an effect which falls off smoothly with distance instead of sharply with the traditional operators.

Bousseau et al. [2007] propose using image morphology operators to make photographs look like water color paintings. In particular, they suggest using *closing* followed by *opening* where closing is defined as erosion followed by dilation and *opening* is defined as dilation followed by closing. Figure 6 shows the results of using

our robust isotropic dilation and erosion operators at the 5% and 95% points. The neighborhood is isotropic and center weighted, and the running time of the algorithm is independent of the size of the neighborhood.

4.4 Dominant-Mode Filter

The median filter is robust in the presence of noise, but its ability to track the center of the input population is limited by its fixed use of the 50% point of the distribution. In principle, a more accurate result is possible by using a robust criterion to choose among the local modes. While many robust selection criteria are possible, a simple and effective method is to choose the mode corresponding to the largest population of samples. We call this the dominant-mode filter.

Computing the dominant mode is straightforward. We use equation 6 to compute not only the negative-going zero crossings in D which represent modes, but also the positive-going zero crossings in D which represent anti-modes. For each mode, we compute the integral of the distribution from its left anti-mode to its right anti-mode using the functions R_i in equation 9. The mode with the largest integral is the dominant mode.

Note that the dominant mode can be used in situations where the domain is circular, as for example the hue of an HSV image, or the 2D orientation of an image feature. In these cases, the median is not meaningful, and something like the dominant mode is required to achieve the kind of noise immunity that the median has on non-circular domains. Note that some care must be used when computing integrals on a circular domain; a branch cut is required. Integrals that span the branch cut must be computed in two pieces.

Figure 5(f) shows the dominant-mode filter applied to the image in Figure 5(a). Like the median filter, it removes the noise very effectively. A close examination will also reveal that it produces a sharper output on some edges than the median. This is a common feature of mode filters. At mode boundaries, the dominant mode filter will switch discontinuously from one mode to another, creating an absolutely sharp edge where a somewhat blurred edge may have existed previously. For a smoother result, modes can be blended together when they account for nearly equal fractions of the population.

Felsberg et al. [2006] use very similar motivation to develop a filter they call *channel smoothing*. Without an efficient algorithm for integrating the population associated with each mode, they rely on an estimate of the smoothed histogram height to pick among modes. This criterion, however, is a dangerous one because the height of a histogram mode depends critically on its variance. A low-variance noise mode with a small population may have a greater histogram height than a high-variance signal mode corresponding to a much larger population. This is shown in Figure 5(f) where channel smoothing accentuates the low-variance noise.

Figure 4(c) shows the dominant-mode filter applied to the tractor image of Figure 4(a). Compared to the closest-mode filtered image in Figure 4(b), high spatial frequency details have been removed. For example, the word "Ford" on the tractor is readable in 4(b), but becomes a single smooth region in 4(c). The white sprinklers in the field are present in 4(b), but disappear from 4(c). Farbman et al. [2008] note "While the [bilateral] is quite effective at smoothing small changes in intensity while preserving strong edges, its ability to achieve progressive coarsening is rather limited." The same can be said of the closest-mode filter. By contrast, increasing the neighborhood size of the dominant-mode filter does spatially coarsen the result.

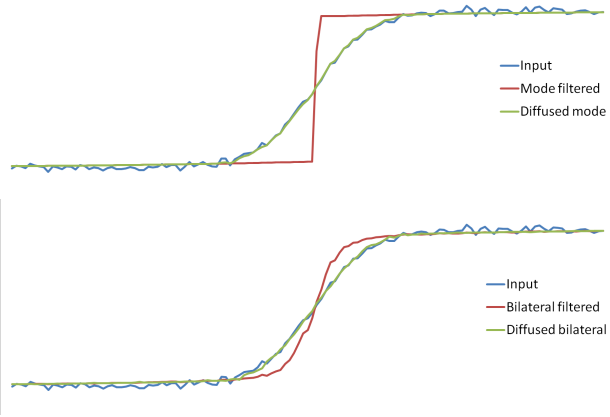


Figure 7: Original synthetic input signal (blue) is filtered (red) resulting in over-sharpening. Our diffusion algorithm yields a smooth signal (green) that no longer over-sharpens. Top: Using closest-mode filter. Bottom: Using the bilateral filter.

5 Separating Base from Texture

In a variety of computational photography applications, images are decomposed into a piecewise-smooth base layer and one or more detail layers [Farbman et al. 2008]. The idea is to capture the large-scale variations of image intensity in the base layer while putting fine-scale image texture into the detail layers. Depending on the application, the base layer and the detail layer can then be processed in different ways and recombined.

The base layer is typically created with an edge-preserving smoothing filter such as the bilateral filter [Durand and Dorsey 2002], anisotropic diffusion [Tumblin and Turk 1999] or a least-squares procedure [Farbman et al. 2008]. The detail layer or layers then characterize what is left after the base layer is removed, either by subtraction or sometimes division. In applications like HDR tone mapping, the detail may be boosted, while in image stylization, the detail may be attenuated.

A key problem with all of these methods is preventing high-contrast large-scale edges from "leaking" into the detail layers, which can cause halos or other artifacts in further processing. Farbman et al. [2008] show in some detail how edge smoothing or blurring in the base-layer filter can produce a large signal in the detail image.

The closest-mode filter (and hence the mean-shift) is very good at tracking the center of the underlying distributions of image textures. Peaks in the local histogram are strong and robust indicators of the centers of the distributions, and they rarely blur high-contrast features. These features should make the filter very attractive. From the point of view of separating a base layer from a detail layer, however, local histogram modes have a severe weakness that has up to now prevented their use: they radically over-sharpen edges.

5.1 Selective Diffusion

Local image histograms alone say nothing at all about the spatial layout of their data samples [Koenderink and Doorn 1999]. They contain no indication of a gradual spatial shift from one mode to another. Thus, in order to track a blurred edge accurately, more information must be extracted from the original images. We propose extracting this information by supplementing edge-preserving histogram-based filters with a *diffusion* step. Our basic observation is that wherever blurring our edge-preserving filter causes it to get

Figure 9: Left: Original. Middle: Diffused closest-mode filter. Right: Detail enhaced by 2.5×.

closer to the original, the blurred version is preferable as a base layer.

Let S be the output of an edge-preserving smoothing filter. Our goal is to construct a modified output image D which is diffused from S anywhere that diffusion causes it to agree more closely with the original input image I. We will do this iteratively, considering a variety of different Gaussian blurring kernels $G(\sigma_i)$ in turn. In our experience, sampling the blurs by ratios of $\sqrt{2}$ works well. Let $D_0 = S$ be the original output of the filter. Then we will construct D_i from D_{i-1} by selectively blending between D_{i-1} and a blurred version $B_i = D_{i-1} * G(\sigma_i)$

An important observation is that we only want to update a pixel with a blurred version if an entire region around that pixel of size σ_i is improved by the blurring. Accordingly we construct error metrics to measure the local L^2 deviation of the unblurred and blurred versions from the original image:

$$\begin{aligned} E_u(\mathbf{p}) &= (D_i - I)^2 * G(\eta\sigma_i) \\ E_b(\mathbf{p}) &= (B_i - I)^2 * G(\eta\sigma_i) \end{aligned} \tag{10}$$

where η controls the region size. We have found $\eta = .2$ works well. Let $R(\mathbf{p}) = E_b(\mathbf{p})/E_u(\mathbf{p})$ be the ratio of the error of the blurred version to the unblurred version. Where R is larger then one, we prefer the unblurred version. Where R is smaller, we blend towards the blurred one. The exact blending is probably unimportant. The particular formula we use is

$$D_i = \begin{cases} B_i & R < .5 \\ 2\left(R - \frac{1}{2}\right)(D_{i-1} - B_i) + B_i & R \in [.5, 1) \\ D_{i-1} & R \geq 1 \end{cases} \tag{11}$$

Figure 7 shows the result for a synthetic blurred step edge with added noise. The closest-mode filter (top) turns the soft edges into very sharp transition. After diffusion, however, it accurately tracks the input edge while smoothing out the high-frequency noise. The bilateral filter (bottom) also sharpens the edge, but not nearly to the same degree. Here again, the diffusion method fixes the over-sharpening.

Figure 8 illustrates the diffusion process for a close crop of the Cormorant photo shown in Figure 9. Figure 8(a) shows the result of closest-mode filtering the original. The edges are very sharp, as is typical for the closest-mode filter. In Figure 8(b) the output of the closest-mode filter has gone through diffusion, greatly softening the edges. Figure 8(c) shows what happens if the closest-mode filter is used as a base image for contrast enhancement. The difference between the original and the base has been scaled by 3.0 and added to the base. The result shows ugly gradient reversal artifacts around the sharp edges, as described by Farbman et al. [2008]. When a diffused version of the closest-mode filter is used as the base image, however, the artifacts go away. The results for the full image are shown in Figure 9.

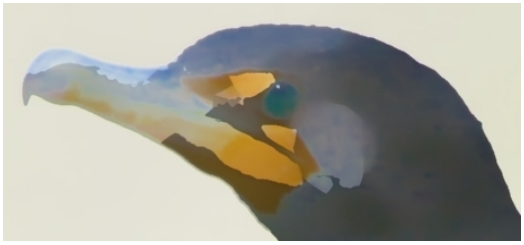
In Figure 10, the same contrast enhancement has been applied using the bilateral filter to separate base from detail. The worst over-sharpening artifacts of the bilateral happen under slightly different circumstances from the closest-mode filter, so we have chosen a different photograph to illustrate the problem. Figure 10 shows the results with the raw bilateral. The halos and gradient reversals are very evident. After diffusion the artifacts are gone.

There are a variety of algorithms that are not specifically histogram-based and have been proposed for separating a base from a detail image. One notable recent example is the method of Subr et al. [2009] which processes local extrema into upper and lower envelopes that are averaged to create the base layer. Like histogram-based methods, this approach can over-sharpen edges. Figure 10 shows the method of Subr et at. used for contrast enhancement. Once again, gradient reversals are evident, particularly at the top of the dog's silhouette. Selective diffusion of the base image removes the gradient reversals.

5.2 Multi-layer

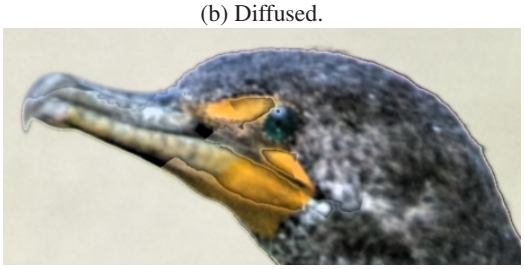
When our diffusion method is used with the closest-mode filter for a base layer, the signal remaining in the detail layer retains a great deal of structure. This suggests that the decomposition can be used recursively. Let $\mathcal{B}_0$ be the base image created by closest-mode filtering an original image I followed by selective diffusion. Then $\mathcal{D}_1 = I - \mathcal{B}_1$ is the first-level detail image. We can apply the same type of decomposition to $\mathcal{D}_1$ forming a new base image $\mathcal{B}_2$ and a detail layer $\mathcal{D}_2$. After n steps, we have a decomposition of the image into a set of layers plus a residual.

$$I = \left(\sum_{i=1}^{n} \mathcal{B}_i\right) + \mathcal{D}_n \tag{12}$$



(a) Closest-mode filtered Cormorant head.

(b) Diffused.



(c) Detail increased by 2.5× causing artifacts.

(d) Detail increased by 2.5× with diffusion.

Figure 8: Close up of closest-mode diffusion.

Instead of changing the spatial neighborhood size as we move through the layers of this decomposition [Burt and Adelson 1983], [Fattal et al. 2008], [Farbman et al. 2008], we find it valuable to change the histogram smoothing kernel K, reducing it at each iteration. This way, the decomposition is in terms of levels of contrast. The highest contrast features appear at the first level, and lower-contrast features appear at succeeding levels. Figure 11 shows an example. For display, we have increased the contrast of $\mathcal{B}_2$, $\mathcal{B}_3$ and $\mathcal{D}_3$. The contrast-enhanced image shown in Figure 11(b) is the sum $\mathcal{B}_1 + 2\mathcal{B}_2 + 2\mathcal{B}_3 + 3\mathcal{D}_3$. The separate layers are shown in Figure 12.

6 Performance

The running time of all the filters we have described are proportional to the inverse of the histogram smoothing and independent of the neighborhood sizes. The great majority of the work is in the spatial filtering, so the best estimate of the running time on any given hardware is the rate at which it can do hierarchical or recursive Gaussian convolutions. Detailed pseudo-code for all the algorithms described here can be found in the accompanying supplemental materials.

(a) Original

(b) Detail increased by 3× from bilateral.

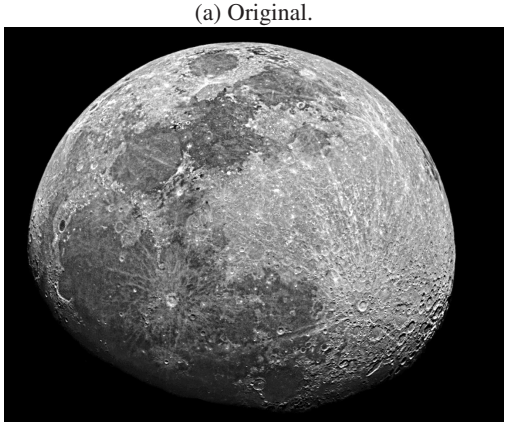
(c) With diffusion.

(d) Detail increased by 3× from Subr et al. base.

(e) With diffusion.

Figure 10: Selective Diffusion with Bilateral and Subr et al.

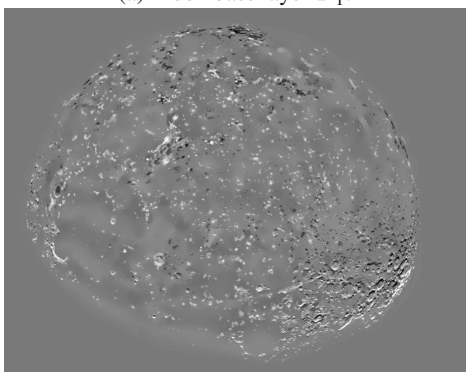
(a) Original.



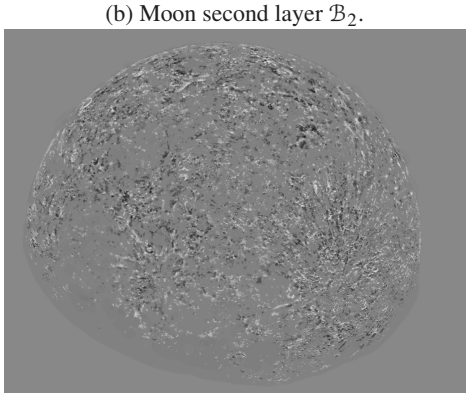
(b) After multi-layer contrast boost.

Figure 11: Multi-layer enhancement of the Moon.

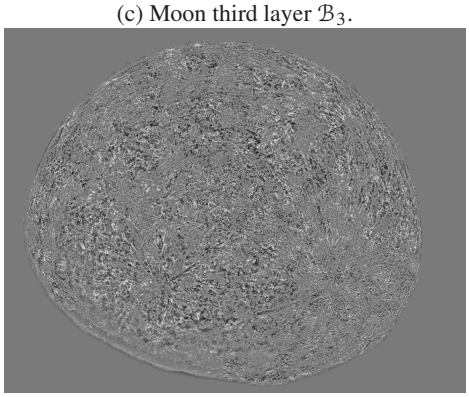
(a) Moon base layer $\mathcal{B}_1$.



(b) Moon second layer $\mathcal{B}_2$.



(c) Moon third layer $\mathcal{B}_3$.



(d) Moon residual layer $\mathcal{D}_3$.

Figure 12: Layers in the decomposition of the Moon.

Our implementations are not heavily optimized and achieve the following speeds. The median and morphology filters run at the rate of approximately 6 megapixels per second in CUDA on an nVidia Quadro FX 770M GPU using 15 samples of s_i and doing all the computations in floating point. The closest mode computations typically require more samples because the derivative look-up table is higher frequency. With 20 samples, we achieve 4 megapixels per second throughput. For the diffusion described in Section 5.1, our GPU implementation achieves 1.2 megapixels per second.

Our single-threaded CPU implementation on an Intel 2.83 GHz Xeon E5440 can compute the median filter or morphology operators at the rate of 1.2 megapixels per second with 15 intensity samples. The closest-mode filter goes at the rate of .54 megapixels per second. The dominant-mode filter is slower because it needs both integrals and derivatives of the histogram. Our CPU implementation computes the result at .36 megapixels per second. Selective diffusion goes at the rate of .67 megapixels per second. We believe there is a great deal of room to improve these numbers with multithreading and a more careful implementation.

7 Conclusion

The ability to compute histogram integrals and derivatives accurately and efficiently opens up a wide range of possibilities for interesting non-linear image computations. Without doubt, the examples in this paper just scratch the surface.

We have provided the first efficient algorithm to compute an isotropic center-weighted median filter and have shown that a small modification enables efficient and robust dilation and erosion operators. In addition to their other benefits, our filters allow continuous change in the filter width. In a traditional square-window median filter, the smallest available filters are 3×3 and 5×5. For some

applications, 3×3 may have too little noise immunity, and 5×5 may smooth the signal too much. Our filters provide a continuum of choices in between.

We have also introduced a new edge-preserving smoothing filter, the dominant mode filter, which allows the robustness of the median filter to apply to circular domains like orientations. In addition, it provides a histogram-based method of smoothing spatial frequencies with larger neighborhood sizes.

Our methods make it possible to compute the closest-mode filter directly without the iterative mean-shift computation and allows fast antialiasing of the result. In addition, we provide a way to overcome the most important limitation of the closest-mode filter for digital photography applications. Our diffusion operator controls over-sharpening, and can be used with any edge-preserving filter. Finally, we note that the separation into base and detail can be repeated for different contrast levels, giving a very useful but different sort of multi-level decomposition than ordinarily used for digital photography or other image processing purposes.

8 Acknowledgements

The photos in Figures 6 and 11 were taken by D'Arcy Norman and Thomas Shahan and licensed under the creative commons.

References

ADAMS, A., GELFAND, N., DOLSON, J., AND LEVOY, M. 2009. Gaussian kd-trees for fast high-dimensional filtering. *ACM Trans. on Graphics 28*, 3, 1–12.

BABAUD, J., WITKIN, A., BAUDIN, M., AND DUDA, R. 1986. Uniqueness of the Gaussian kernel for scale-space filtering. *IEEE Trans. Pattern Anal. Mach. Intell. 8*, 1, 26–33.

BARASH, D., AND COMANICIU, D. 2004. A common framework for nonlinear diffusion, adaptive smoothing, bilateral filtering and mean shift. *Image and Vision Computing 22*, 73–81.

BOUSSEAU, A., NEYRET, F., THOLLOT, J., AND SALESIN, D. 2007. Video watercolorization using bidirectional texture advection. *ACM Trans. on Graphics 26*, 3, 104.

BURT, P. J., AND ADELSON, E. H. 1983. The Laplacian pyramid as a compact image code. *IEEE Trans. on Communications COM-31,4*, 532–540.

BURT, P. 1983. Fast filter transforms for image processing. *Computer Graphics and Image Processing 16*, 532–540.

CHEN, J., PARIS, S., AND DURAND, F. 2007. Real-time edge-aware image processing with the bilateral grid. *ACM Trans. on Graphics 23*, 3 (July), 103.

COMANICIU, D., AND MEER, P. 2002. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. on Pattern Analysis and Machine Intelligence 24*, 5, 603–619.

DERICHE, R. 1993. Recursively implementing the Gaussian and its derivatives. Tech. Rep. 1893, INRIA, Unit de Recherche Sophia-Antipolis.

DOMINGUEZ, G., BISCHOF, H., AND BEICHEL, R. 2003. Fast 3d mean shift filter for CT images. In *Image Analysis, Proc. SCIA '2003*, Springer, 59–97.

DUDA, R., AND HART, P. 1973. *Pattern Classification and Scene Analysis*. Wiley.

DURAND, F., AND DORSEY, J. 2002. Fast bilateral filtering for the display of high-dynamic-range images. In *Proc. SIGGRAPH '02*, 257–266.

FARBMAN, Z., FATTAL, R., LISCHINSKI, D., AND SZELISKI, R. 2008. Edge-preserving decompositions for multi-scale tone and detail manipulation. *ACM Trans. on Graphics 27*, 3.

FATTAL, R., AGRAWALA, M., AND RUSINKIEWICZ, S. 2008. Multiscale shape and detail enhancement from multi-light image collections. *ACM Trans. on Graphics 26*, 3.

FELSBERG, M., FORSSÉEN, P.-E., AND SCHARR, H. 2006. Channel smoothing: Efficient robust smoothing of low-level signal features. *IEEE Trans. on Pattern Analysis and Machine Intelligence 28*, 2, 209–222.

FUKUNAGA, K., AND HOSTETLER, L. D. 1975. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Trans. on Information Theory 21*, 32–40.

HARALICK, R., STERNBERG, S., AND ZHUANG, X. 1987. Image analysis using mathematical morphology. *IEEE Trans. Pattern Analysis and Machine Intelligence 9*, 4 (July), 532–550.

HUANG, T. 1975. *Two-Dimensional Signal Processing II: Transforms and Median Filters*. Springer Verlag, Berlin.

KOENDERINK, J. J., AND DOORN, A. J. V. 1999. The structure of locally orderless images. *International Journal of Computer Vision 31*, 2-3, 159–168.

PARIS, S., AND DURAND, F. 2006. A fast approximation of the bilateral filter using a signal processing approach. In *Proc. European Conference on Computer Vision (ECCV '06)*, 568–580.

PARIS, S., AND DURAND, F. 2007. A topological approach to hierarchical segmentation using mean shift. In *Proc. IEEE conference on Computer Vision and Pattern Recognition (CVPR'07)*.

PARIS, S., KORNPROBST, P., TUMBLIN, J., AND DURAND, F. 2007. A gentle introduction to bilateral filtering and its applications. In *SIGGRAPH '07: ACM SIGGRAPH 2007 courses*.

PARZEN, E. 1962. On estimation of a probability density function and mode. *Ann. Math. Stat. 33*, 1065–1076.

PERREAULT, S., AND HEBERT, P. 2007. Median filtering in constant time. *IEEE Trans. on Image Processing 16*, 9 (Sep.), 2389–2394.

PORIKLI, F. 2005. Integral histogram: A fast way to extract histograms in cartesian spaces. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '05)*, 829–836.

PORIKLI, F. 2008. Constant time o(1) bilateral filtering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '08)*, 1–8.

SUBR, K., SOLER, C., AND DURAND, F. 2009. Edge-preserving multiscale image decomposition based on local extrema. In *SIGGRAPH Asia '09: ACM SIGGRAPH Asia 2009 papers*, 1–9.

TUMBLIN, J., AND TURK, G. 1999. LCIS: A boundary hierarchy for detail-preserving contrast reduction. In *Proc. SIGGRAPH '99*, 83–90.

VAN HERK, M. 1992. A fast algorithm for local minimum and maximum filters on rectangular and octagonal kernels. *Pattern Recognition Letters 13*, 7, 517–521.

WEISS, B. 2006. Fast median and bilateral filtering. In *Proc. SIGGRAPH '06*, 519–526.