The Unsharp Mask Process
in Photographic Image Processing

Douglas A. Kerr

Issue 2
June 5, 2010

ABSTRACT AND INTRODUCTION

Unsharp mask refers to a process used in both film and digital photographic processing to increase the apparent “sharpness” of an image. In this article we explain the working of both film and digital versions of the process, and discuss the meaning of the name. The article does not attempt to teach techniques for the effective use of the process.

BACKGROUND

Image masks

This article is in part about the term *mask* and how it gets into the matter under discussion. Accordingly, I thought it would be helpful to give a little background on the concept of an *image mask*.

Suppose that we have a film negative of a shot of Joe standing in front of his garage and a negative of a nice shot of the White House. We would like to make a composite print that seems to show Joe in front of the White house.

Suppose we have a sheet of developed photographic film that we have somehow arranged to be transparent (“clear”) exactly where Joe’s image is on the “Joe and garage” negative and opaque everywhere else. We’ll call this our “A mask”.

We make a contact print of this on another sheet of negative film, and thus end up with a sheet of developed film that is opaque exactly where Joe’s image is on the “Joe and garage” negative and transparent everywhere else. We’ll call this our “B mask”.

We “sandwich” the A mask with the “Joe and garage” negative in the negative carrier of our enlarger, and make an exposure onto printing paper.

We don’t develop the paper now, but if we did, we would have an image of Joe on a white background—the A mask has prevented any light from reaching the paper from outside Joe’s image.

Then we “sandwich” the B mask with the “White House” negative in the negative carrier of our enlarger, and make another exposure onto printing paper. This has no effect on the latent image of Joe—the B mask has prevented any light from the area corresponding to Joe's image from reaching the paper.

We develop the paper, and (assuming everything was done precisely) we have a print of Joe in front of the White House.

Note that we chose to emphasize what the *opaque* region of these masks does: they **block** light from the unwanted parts of the image being "exposed". Of course the complementary aspect of the mask (its *transparent* region) is equally important—it **lets through** light from the wanted parts of the image. But the name "mask" evokes the "blocking" outlook on the process. As a result, some people feel that the term "mask" only really applies to the opaque portion—the transparent portion is just "not mask", traveling with the mask on the same piece of acetate film stock.

A further subtlety enters the picture when we imagine a part of the mask that is "semitransparent". A semitransparent region "lets through" the image there, but "diluted". This can play a role in more complex operations, which we will see later.

When we get into the digital world, we find this same concept implemented digitally. There, a mask used in a way similar to what we described in film work is called a *digital image mask*. The qualifier "image" reminds us that, in the same context, we often find other masks that block, or let through, things that are not "image" (perhaps "things that are done to the image"). Some masks serve both image and non-image masking purposes.

A map in the digital world can be thought of as a "map" of the image area with a numerical value at each pixel location, which we can think of as running from 0 to 1. Often we describe the regions of the map in which the transparency is 1 as *transparent*, those where the transparency is 0 as *opaque*.

In the digital world, we often find that a single mask can play the role of both the "A" and "B" masks in our example above. That is, it can let image material from source A (Joe and garage) through a certain region (where Joe's image is), and image material from source B (the White House) through the remainder, all in one operation.

Typically, where the mask transparency is 1, the "A source" material **is** let through, and if there is a "B source" involved, that material **is not** let through. (So for the "B source" material, a mask "transparency" value of 1 corresponds to *opaque*.)

Where the mask transparency is 0, the "A source" material **is not** let through, and if there is a "B source" involved, that material **is** let through.

Now suppose in some region the mask transparency is 0.40 (for example). Here is what happens. Note that this usually works in terms of RGB values, not in terms of the luminance they imply.

For example, if, at the point of interest, the A source image has R=200, then in the new image created by the mask operation, at that point, there will be a contribution to the R value from the A source of 80 RGB units [0.4*200]. If the B source at that point has an R value of 150, then in the new image at that point there will be a contribution from the B source of 90 RGB units [(1-0.4)*150]. Thus the actual final R value at that point in the result will be 170 [80+90].

Especially when we consider such intermediate values of transparency, a mask that works with two sources is sometimes called a *blending mask*.

Sharpness

Especially in the digital world, we generally apply the *unsharp mask* process to increase the “sharpness” of an image. But just what is that?

“Sharpness” is a non-quantifiable property that reflects the impact of two properties that can be quantified:

- *Resolution*, which we can think of simplistically as how many reversals of brightness can the system capture per millimeter of sensor, or across the entire sensor.

- *Acutance*, which simplistically means how faithfully does the system preserve a “sharp” change in luminance (at an edge of some object in the image).

We find that the application of the *unsharp mask* process to an image generally increases “perceived acutance”, and thus “perceived sharpness”. We’ll see later just what this means.

UNSHARP MASK IN FILM IMAGE PROCESSING

The *unsharp mask* process first emerged in the world of film photography, and its name comes from that practice. I’ll discuss it briefly just for historical continuity. It is easiest to imagine a black-and-white context.

We take our image negative and make a “contact print” of it on negative film stock, in such a way that the image there is slightly blurred. The usual way to do that is to put the receiving film against the face of the image negative opposite the emulsion. If the light source is diffuse, the spacing between the negative emulsion and the emulsion of the receiving film (owing to the thickness of the negative base material) causes the image to be slightly blurred.

We develop this film image, which will be a “positive” (more transparent where the actual scene was lighter, and thus the negative is more opaque).

We then sandwich the positive image with the original negative in the negative carrier of our enlarger when we make a print. We can view this film positive as a *mask*, which of course has a range of transparencies (the complement of the transparencies of the negative proper).

Rather than trying to show density profiles along a path across the “edge” (as we will do later in the digital application of the technique), we will look at the result of this in the spatial frequency domain.

When we blur an image, we essentially apply a *spatial lowpass filter* to the image. That is, the fine detail (corresponding to high spatial frequency components) is partially suppressed by the blurring; the coarse detail (corresponding to low spatial frequency components) is not suppressed.

When we sandwich the positive “mask” with our negative, it in effect partially cancels out the transparency variations in the negative that convey the image. But the high-frequency information in the positive mask is (as we just discussed) has relatively less amplitude than the low-frequency information.

Thus the high-frequency information in the negative is “less canceled out” than the low frequency information. The resulting relative increase in the high-frequency components (compared to the low-frequency components) is tantamount to a “sharpening” of the image. (Of course, the exposure in printing has to be adjusted to accommodate the overall lower transparency of the “sandwich”.)

Our positive film copy is reasonably called an “unsharp mask”: it is *unsharp* (blurred), and it is used as a *mask* (its transparency “modulating” the image light passing through the negative).

Relish this—it’s the last time here we’ll see the term used so aptly.

UNSHARP MASK IN DIGITAL IMAGE PROCESSING

The *unsharp mask* process is an available tool in many general-purpose image editing software packages, as well as many special-purpose packages (such as those used for photomicrography, astrophotography, and such). We don’t usually have access to the details of the algorithms used in the general-purpose editors. But the algorithms used in some of the other applications are documented, so we can come to some general conclusions as to how all the algorithms probably work. Accordingly, I will start by describing the working of the "classical" digital unsharp mask algorithm.

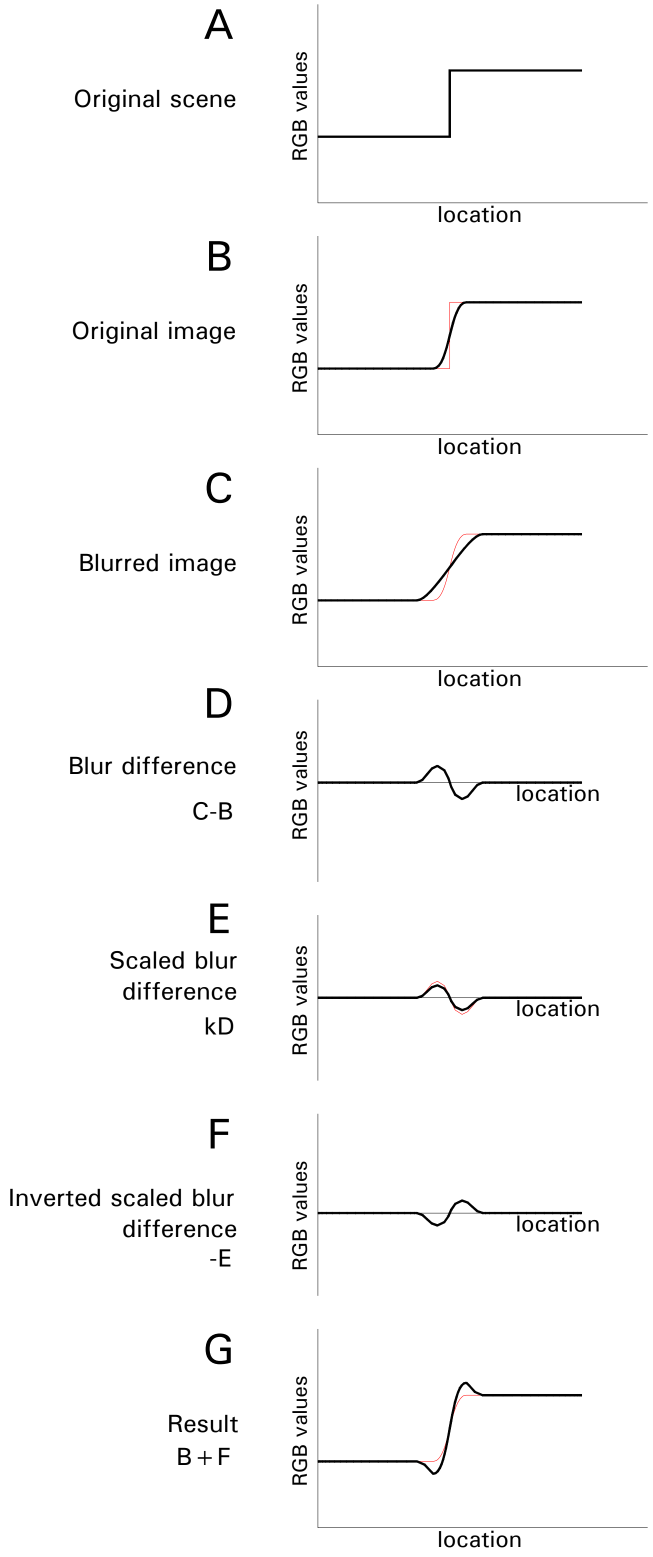

Figure 1. "Classical" unsharp mask algorithm

The “classical” digital unsharp mask algorithm

The operation of this algorithm is shown on figure 1. Note the following cautions:

- The drawings show “continuous” functions of luminance along a path across an edge of the image. In reality, we have a “discrete” function in the sense that the luminance can only change pixel-by-pixel, and only by integral steps of the digital number that represents luminance. The “continuous function” portrayal, though, seems to most clearly reveal just what is going on in the process.

- The curves are “hand drawn” and are intended to illustrate the principles. They are not necessarily “mathematically accurate”.

We start with panel A, which shows the plot of luminance along a line across a sharp edge on the scene itself (a test chart, perhaps). We will refer to the corresponding function of luminance vs. position as “function A”.

In panel B, we see the luminance profile on our digital image (function B). The “sharpness” of the transition has been compromised by the finite resolution of the camera (and perhaps by other phenomena). Our application of the unsharp mask process is motivated by our desire to overcome the visual degradation in perceived image sharpness caused by this phenomenon. (We show function A in red for comparison.)

In panel C, we see a copy of our starting image (function B) that has been blurred, as part of the algorithm, by applying a Gaussian blur function. (This is a classical mathematically-defined blurring process.) (We show function B in red for comparison.)

In panel D, we have a function generated by subtracting function C from function B at each point along the path.

In panel E, we have function D inverted.

In panel F, we see function E added to our original image, function B. Function E is what is recorded as the result of the process. Function B (the original image) is shown in red for comparison.

Let’s consider what we have here. Did we get back the “sharp” transition the actual scene has? No. We basically still have the more gradual transaction of our original image, but with an “overshoot” on each end.

This overshoot makes the human eye interpret the transition as “sharper” than it really is (that is, it has a higher apparent acutance). Thus, the overall image “looks sharper” than it did before application of the unsharp mask process.

What part of this process can reasonably be called a “mask” (unsharp or otherwise)? Nothing. The name comes forward only out of historical respect.

An alternate unsharp mask algorithm

The photographic site “Cambridge in Colour” provides a nice description of an unsharp mask algorithm different from the one I just described. It is entirely possible that this is the algorithm actually used, for example, in the Photoshop image editing software. You can see the explanation here:

http://www.cambridgeincolour.com/tutorials/unsharp-mask.htm

I think this explanation leaves out a couple of little details.

In any case, figure 2 shows my understanding of the algorithm Cambridge in Color seems to be describing.

As before, panel A shows the luminance function along a track across a sharp transition on the scene, and panel B shows the actual image (slightly blurred by resolution limitations and other factors).

Also as before, we see in panel C a blurred copy of our original image, constructed by the algorithm using the Gaussian blur function.

And as before, in panel D, we see the function developed by subtracting function B from function C.

In panel E, we have taken the absolute value of function D; that is, we have a positive value whether, at that location, function B is larger than function C or *vice versa*.

In panel F, we have scaled function E and expressed it on a scale of 0-1 (as a *mask transparency factor*). (Here, we have a *bona fide* mask.)

In panel G, we see an “enhanced contrast” copy of our original image (function B). Note that it has a higher luminance value than the original image for the “light side” of the transition (to the right), and a lower luminance on the “dark side” (to the left). (We see the original image function in red for comparison.)

In panel H, we have used function F as a “blending mask”, combining functions B and G. (We see functions B and G in two different colors to help us follow the action here.)

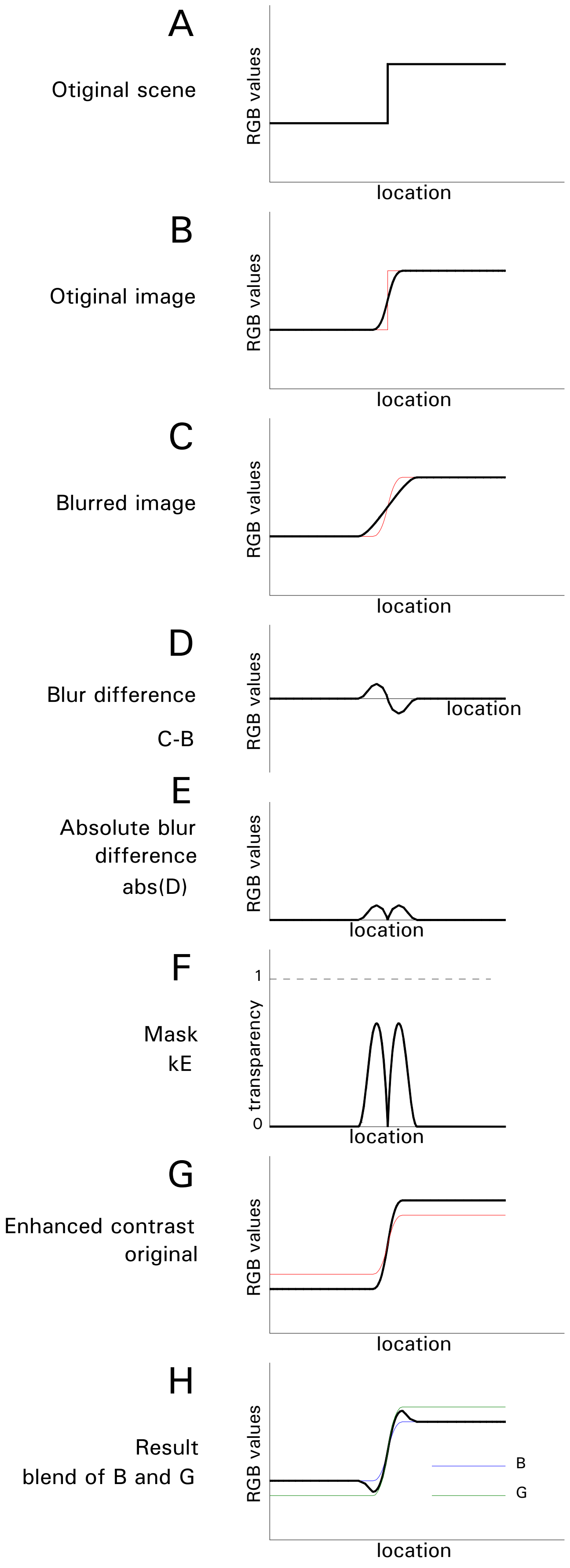

Figure 2. Alternate unsharp mask algorithm

Any place where the mask transparency (function F), the resulting function will have the value of function G (the "enhanced contrast original"). (We don't have any such place on the function in this example.) Any place where the mask transparency is 0, the resulting function will have the value of function B (the "original"). Any place where the transparency is (for example) 0.4, the resulting function will be reckoned as 0.4 times the value of function G plus 0.6 (that is, 1-0.4) times the value of function B.

We see that, generally speaking, this result (function H) looks about the same as the result (function G) of the classical algorithm.

I'm not sure if this is really a different algorithm, or just a more complicated way of describing the same one. If it is a different one, I'm not sure what advantage it has (since I do not have enough details of it to model it mathematically, nor do I have the energy if I did).

One advantage, though, is that its story actually has a "mask" in it.

What does the Cambridge in Color description gloss over?

The description does not mention the "rectification" of the blur difference function (as we see in panel E), although we see it if we analyze on a pixel basis their illustration of the difference "map". (Among other clues is of course that this "map", working as it does in RGB values, could not have negative values.)

The description does not suggest the scaling of the blur difference function to become the "blending mask" (my function F). For example, what "RGB" value of the "difference" map would correspond to letting through 100% of function G and none of function B (the largest possible adjustment.

I mention these niggles not to criticize the folks at Cambridge in Colour, who have done a great service with their article, but just for the benefit of those masochists that might be trying to follow the whole train of events and can't quite get there from here.

Parameters

Normally, when an unsharp mask procedure is available in an image editor or equivalent, there are three parameters that can be set to vary its performance. Amazingly enough, these are generally described by the same set of names in the different software packages in which they appear:.

- **Amount**. This controls the degree of the adjustment applied to the original image. It works by controlling the scaling from function D to function E in the algorithm of figure 1, or from function E to function F in the algorithm of figure 2.

- **Radius**. The controls the “breadth” of the Gaussian blur used to create function C (in either algorithm), and thus the “breadth” of function D. Larger values of this parameter make the sharpening more prominent, but smaller values ensure that closely-spaced boundaries (“finer detail”) receive full benefit of sharpening.

- **Threshold**. Not shown in the figures is that the function D (in either algorithm) is “center clipped”. That means that values whose absolute value would be less than the Threshold value are replaced by zero. The practical effect of this is that the algorithm declines to make any adjustment to edges that have a small luminance step. (Such adjustments are often visually counter-productive.)

FURTHER DETAILS

There are numerous details of the application of the unsharp mask process within modern image-processing applications that are beyond the intent and scope of this article. For example, we typically can elect to have the algorithm operate (independently) on the three “channels” of the image color information (R, G, and B), or we can choose to have the color reorganized into luminance and chromaticity aspects and apply the sharpening to the luminance aspect only.

THE GAUSSIAN BLUR FUNCTION

We have described the development of function C from function B (the original image) by the application of a “Gaussian Blur” function.

Rigorously, a Gaussian blur is produced by taking every point in an image and turning it into a “Gaussian spread function” that extends in all directions from the point itself. The superposition of all those spread functions, from all the points in the original image, is the “result”.

Although the actual function is three-dimensional, we may choose to first apply a “one-dimensional” form of the function in one direction, and then apply the one dimensional function in the other direction to the that result. The two final results are theoretically identical.

In our practical discussion, we will only consider application of the function in one dimension (horizontal).

Of course, the Gaussian spread function is continuous, while we need only a discrete function (having values only at pixel locations). And the points that are involved are only those corresponding to pixel locations.

In many implementations of the “classical” unsharp mask algorithm, the discrete spread function used (when the “width” of the spread is set to its lowest practical value, usually described as “radius = 1”) is as follows (successive values are at successive pixel locations). The location of the “source pixel” is indicated with a dagger:

```
                         †
 . . . 0.00  0.00  0.25  0.50  0.25  0.00  0.00 . . .
                         |-radius->
                         (1 pixel)
```

That is, in the buildup of the result, from this one pixel, at the location of the point itself we have a “color value” of 0.50 times the value at the point; one pixel to the left we have a value 0.25 times that of our point, and one pixel to the right, we have a value 0.25 times that of our point.

We can see why we consider the elemental blur pattern produced by this process to have a radius of 1 pixel (as set).

Our final result is just the sum, at each pixel, of all these contributions (which, for this basic function, come from that pixel itself and from the pixels to its left and right)

The scaling of these factors is such that, when we add up the contributions at each pixel of the result, if the initial color value is the same at all the pixels, the color value of the result is that same value at each pixel. (That is, if we blur an image of a uniform color, we get a image of that same uniform color.)

Now, if we set the *radius* to 2 (to get “twice as wide a blur”), the program uses a “wider” blur function that is just the sum of our basic blur function, a copy of itself shifted to the right by one pixel position, and another copy shifted left by one pixel position. We can see that buildup here:

```
                          †
Basic  0.000 0.000 0.250 0.500 0.250 0.000 0.000 . . .
Copy 1 0.000 0.000 0.000 0.250 0.500 0.250 0.000 . . .
Copy 2 0.000 0.250 0.500 0.250 0.000 0.000 0.000 . . .
Sum    0.000 0.250 0.750 1.000 0.750 0.250 0.000 . . .
Scaled 0.000 0.083 0.250 0.333 0.250 0.083 0.000 . . .
                         |---- radius ---->
                             (2 pixels)
```

The (re)scaling is to arrange that the overall result still has consistent scaling, as described in the first example (“if we blur a uniform color image...”).

We can see why we consider the blur pattern produced by this process to have a radius of 2 pixels (as requested).

The “breadth” of a Gaussian spread function is characterized by the parameter σ (lower-case Greek *sigma*). Yes, this is the same symbol used for *standard deviation* in statistics, and in fact the Gaussian spread function is exactly the same as the *probability density function* of the normal statistical distribution.

For the mathematically-convenient discrete spread matrix first given above (for “radius = 1”), the result is the same (at the discrete pixel centers) as for an actual Gaussian spread function with a sigma of about 0.849. The second discrete spread matrix shown above (“radius = 2”) corresponds to a Gaussian spread function with a sigma of about 1.698, twice that for the “radius = 1” case.

Thus, the construction of a discrete spread function for larger values of radius by the summation of multiple copies of the “basic” discrete spread function appears to yield a discrete Gaussian function of correspondingly greater sigma. The mathematical properties of a Gaussian function in fact predict this would be so. (And how very handy that is in the software!)

#