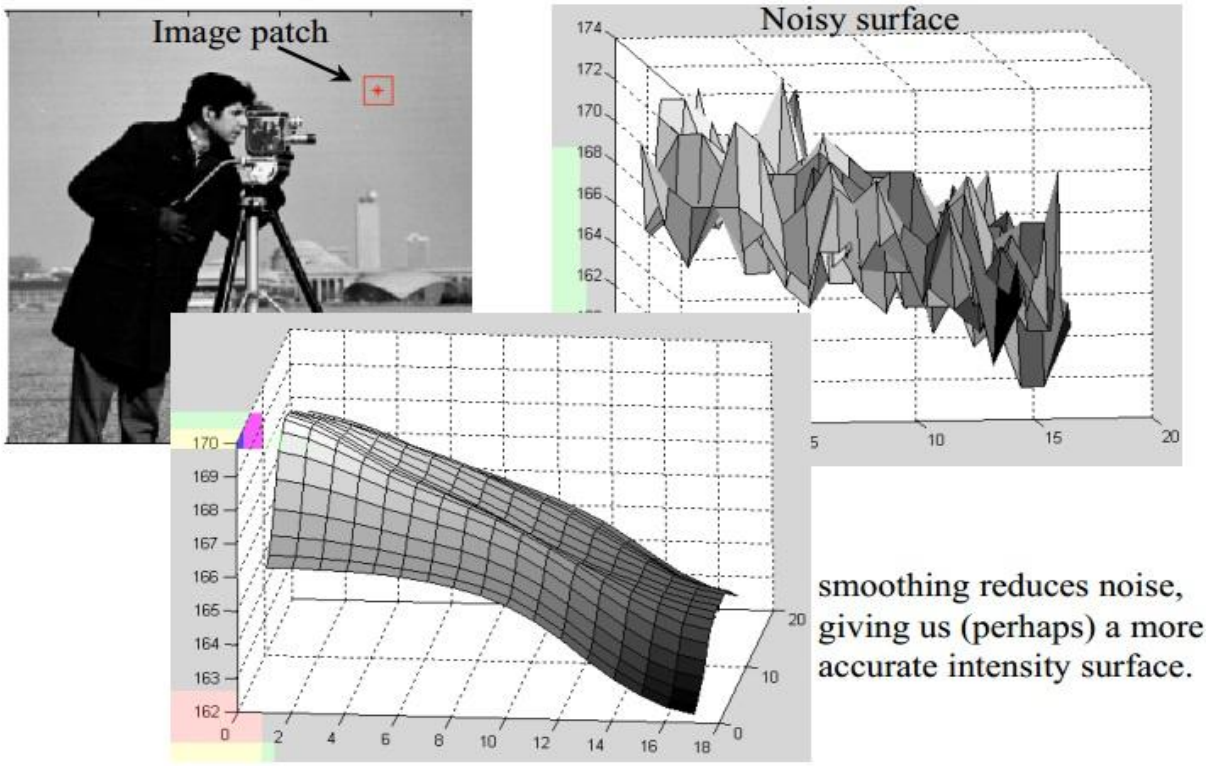
Smoothing

The premise of data smoothing is that one is measuring a variable that is both slowly varying and also corrupted by random noise. Then it can sometimes be useful to replace each data point by some kind of local average of surrounding data points. Since nearby points measure very nearly the same underlying value, averaging can reduce the level of noise without (much) biasing the value obtained.

Smoothing Reduces Noise



Why Averaging Reduces Noise

• Intuitive explanation: variance of noise in the average is smaller than variance of the pixel noise (assumingzero-mean Gaussian noise).

• Sketch of more rigorous explanation:

$$A = \frac{1}{m^2}\sum_{i=1}^{m^2} I_m$$

$$I_m = s_m + n_m \text{ with n being i.i.d. } G(0,\sigma^2)$$

$$E(A) = \frac{1}{m^2}\sum s_m$$

$$var(A) = E\left[(A - E(A))^2\right] = \frac{\sigma^2}{m}$$

Important Point about Smoothing

Averaging attenuates noise (reduces the variance), leading to a more “accurate” estimate.

However, the more accurate estimate is of the mean of a local pixel neighborhood! This might not be what you want. Balancing act: smooth enough to "clean up"the noise, but not so much as to remove important image gradients.

Mean Filter

Common Names: Mean filtering, Smoothing, Averaging, Box filtering

Brief Description

Mean filtering is a simple, intuitive and easy to implement method of *smoothing* images, *i.e.* reducing the amount of intensity variation between one pixel and the next. It is often used to reduce noise in images.

How It Works

The idea of mean filtering is simply to replace each pixel value in an image with the mean (`average') value of its neighbors, including itself. This has the effect of eliminating pixel values which are unrepresentative of their surroundings. Mean filtering is usually thought of as a convolution filter. Like other convolutions it is based around a kernel, which represents the shape and size of the neighborhood to be sampled when calculating the mean. Often a 3×3 square kernel is used, as shown in Figure 1, although larger kernels (*e.g.* 5×5 squares) can be used for more severe smoothing. (Note that a small kernel can be applied more than once in order to produce a similar but not identical effect as a single pass with a large kernel.)

$$\begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix}$$

Figure 1 3×3 averaging kernel often used in mean filtering

Computing the straightforward convolution of an image with this kernel carries out the mean filtering process.

Guidelines for Use

Mean filtering is most commonly used as a simple method for reducing noise in an image.

We illustrate the filter using

The image

shows the original corrupted by Gaussian noise with a mean of zero and a standard deviation ( ) of 8.

The image

shows the effect of applying a 3×3 mean filter. Note that the noise is less apparent, but the image has been `softened'. If we increase the size of the mean filter to 5×5, we obtain an image with less noise and less high frequency detail, as shown in

The same image more severely corrupted by Gaussian noise (with a mean of zero and a of 13) is shown in

The image

is the result of mean filtering with a 3×3 kernel.

An even more challenging task is provided by

. It shows an image containing `salt and pepper' shot noise.

The image

shows the effect of smoothing the noisy image with a 3×3 mean filter. Since the shot noise pixel values are often very different from the surrounding values, they tend to significantly distort the pixel average calculated by the mean filter.

Using a 5×5 filter instead gives

This result is not a significant improvement in noise reduction and, furthermore, the image is now very blurred.

These examples illustrate the two main problems with mean filtering, which are:

- A single pixel with a very unrepresentative value can significantly affect the mean value of all the pixels in its neighborhood.
- When the filter neighborhood straddles an edge, the filter will interpolate new values for pixels on the edge and so will blur that edge. This may be a problem if sharp edges are required in the output.

Both of these problems are tackled by the median filter, which is often a better filter for reducing noise than the mean filter, but it takes longer to compute.

In general the mean filter acts as a lowpass frequency filter and, therefore, reduces the spatial intensity derivatives present in the image. We have already seen this effect as a `softening' of the facial features in the above example. Now consider the image

which depicts a scene containing a wider range of different spatial frequencies. After smoothing once with a 3×3 mean filter we obtain

Notice that the low spatial frequency information in the background has not been affected significantly by filtering, but the (once crisp) edges of the foreground subject have been appreciably smoothed. After filtering with a 7×7 filter, we obtain an even more dramatic illustration of this phenomenon in

Compare this result to that obtained by passing a 3×3 filter over the original image three times in

Common Variants

Variations on the mean smoothing filter discussed here include *Threshold Averaging* wherein smoothing is applied subject to the condition that the center pixel value is changed only if the difference between its original value and the average value is greater than a preset threshold. This has the effect that noise is smoothed with a less dramatic loss in image detail.

Other convolution filters that do not calculate the mean of a neighborhood are also often used for smoothing. One of the most common of these is the Gaussian smoothing filter.

High Pass vs Low Pass Filters

Lowpass filter (smoothing)

A low pass filter is used to pass low-frequency signals. The strength of the signal is reduced and frequencies which are passed is higher than the cut-off frequency. The amount of strength reduced for each frequency depends on the design of the filter. Smoothing is low pass operation in the frequency domain.

Following are some lowpass filters:

Original image

1. Ideal Lowpass Filters

The ideal lowpass filter is used to cut off all the high-frequency components of Fourier transformation.

Below is the transfer function of an ideal lowpass filter.

$$H(u,v) = \begin{cases} 1 & if\ D(u,v) \le D0 \\ 0 & if\ D(u,v) > D0 \end{cases}$$

$$D(u,v) = \left[\left(u - \frac{M}{2}\right)^2 + \left(v - \frac{N}{2}\right)^2\right]^{\frac{1}{2}}$$

ideal lowpass

2. Butterworth Lowpass Filters

Butterworth Lowpass Filter is used to remove high-frequency noise with very minimal loss of signal components.

$$H(u,v) = \frac{1}{1+\left[\frac{D(u,v)}{D0}\right]^{2n}}$$

Butterworth Lowpass

3. Gaussian Lowpass Filters

The transfer function of Gaussian Lowpass filters is ahown below:

$$H(u,v) = e^{-D^2(u,v)/2D_0^2} \quad (8)$$

Gaussian Lowpass

Highpass filters (sharpening)

A highpass filter is used for passing high frequencies but the strength of the frequency is lower as compared to cut off frequency. Sharpening is a highpass operation in the frequency domain. As lowpass filter, it also has standard forms such as Ideal highpass filter, Butterworth highpass filter, Gaussian highpass filter.

What does "image sharpening" mean?

To understand why Photoshop's High Pass filter is so good at sharpening images, it helps to understand how image sharpening works in general.

Much like a good magic trick, image sharpening is an illusion. It works by increasing contrast along the edges in your image. Photoshop considers an *edge* to be any area where there's a big, sudden change in brightness between neighboring pixels.

Increasing contrast along the edges makes the light side of the edge lighter and the dark side darker. Your brain then interprets the increased contrast as "sharper". The more we boost edge contrast, the sharper the image appears.

High-boost Filtering

i. High-boost filter is a sharpening second order derivative filter.

ii. High-boost filter image is obtained by subtracting LPF image from the scaled input image.

HBF image = k(original image) – LPF image

= (k-1)original image + original image – LPF

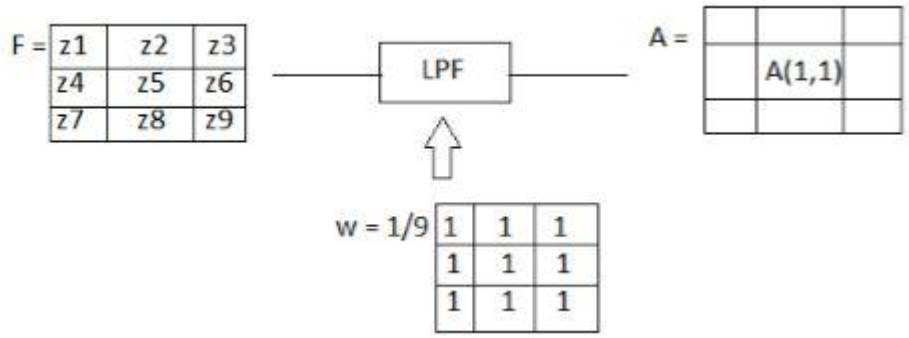
= (k-1)original image + HPF image

where k is any positive scaling factor.

For k-1, HBF image = HPF image, therefore for HBF image k > 1 let us derive HBF mask by considering a digital image F

$ F = \begin{bmatrix} z_1 & z_2 & z_3 \ z_4 & z_5 & z_6 \ z_7 & z_8 & z_9 \ \end{bmatrix} $

Step 1: Find LPF image using Low Pass Averaging filter mask

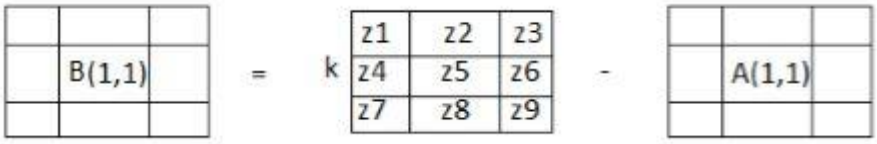


To find A(x=1, y=1)

A(1,1)=$\frac{1}{9}[z_1+z_2+z_3+z_4+z_5+z_6+z_7+z_8+z_9]$

Step 2: Find HBF image

HBF image = k(original image) – LPF image



To find B(x=1, y=1)

B(1,1)=kz_5-A(1,1)

B(1,1)=$kz_5-\frac{1}{9}[z_1+z_2+z_3+z_4+z_5+z_6+z_7+z_8+z_9]$

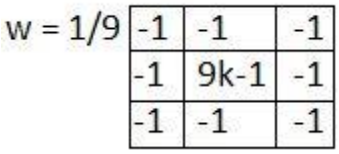
B(1,1)=$(k-\frac{1}{9}) z_5-\frac{z_1}{9}-\frac{z_2}{9}-\frac{z_3}{9}-\frac{z_4}{9}-\frac{z_5}{9}-\frac{z_6}{9}-\frac{z_7}{9}-\frac{z_8}{9}-\frac{z_9}{9}$

Step 3: Find HBF mask

$w_1=-\frac{1}{9} w_2=-\frac{1}{9} w_3=-\frac{1}{9}w_4=-\frac{1}{9} w_5=\frac{9k-1}{9}$

$w_6=-\frac{1}{9} w_7=-\frac{1}{9} w_8=-\frac{1}{9} w_9=-\frac{1}{9}$

The HBF mask is given by,



Original image

Gaussian highpass

Butterworth highpass

Ideal highpass