3 Simple Noise Reduction

3.1.1 Gaussian white noise

Gaussian white noise can be suppressed by applying a low-pass filter (e.g. Gaussian filter, uniform filter). This has the drawback that, together with the noise, edges (high frequency information) are also blurred. A better solution for this is anisotropic diffusion. This is a technique where blurring is applied far from the edges of the image while close to the edges almost no blurring is applied.

3.1.2 Exercise 11

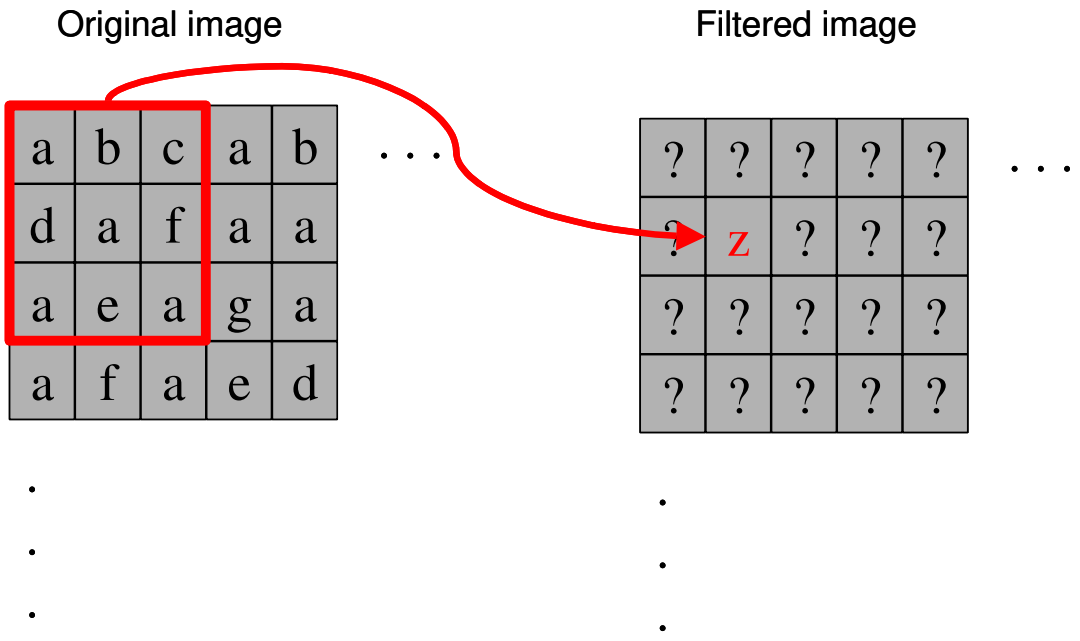
Load *lenagray.tif* and convert to double. Add Gaussian noise with sigma=0.1 with the imnoise command. Execute anisotropic diffusion with use of the function *anisodiff.m*. Choose iterations equal to 70 and k equal to 0.1. After this, apply normal Gaussian blurring (filtering (conv2) with a Gaussian filter (fspecial('gaussian',...,...))) on the image. The choice of sigma=sqrt(2*t1) with t1=7 is the equivalent of anisotropic diffusion (internally 70 iterations correspond with t1=7). Compare both results. Pay particular attention to the edges.

3.1.3 Salt and Pepper noise (binary noise)

When an image is suffering from salt and pepper noise the image is contaminated with black and white pixels (see figure 'Lena' in next page). This type of noise can be removed in a simple way by using median filtering.

Median filtering is a non-linear operation where the target pixel is the median of the source pixel and its 8 neighbouring pixels. The following figure clarifies this.

```
z=median([a b c d a f a e a])
```



3.1.4 Exercise 12 – Median filter

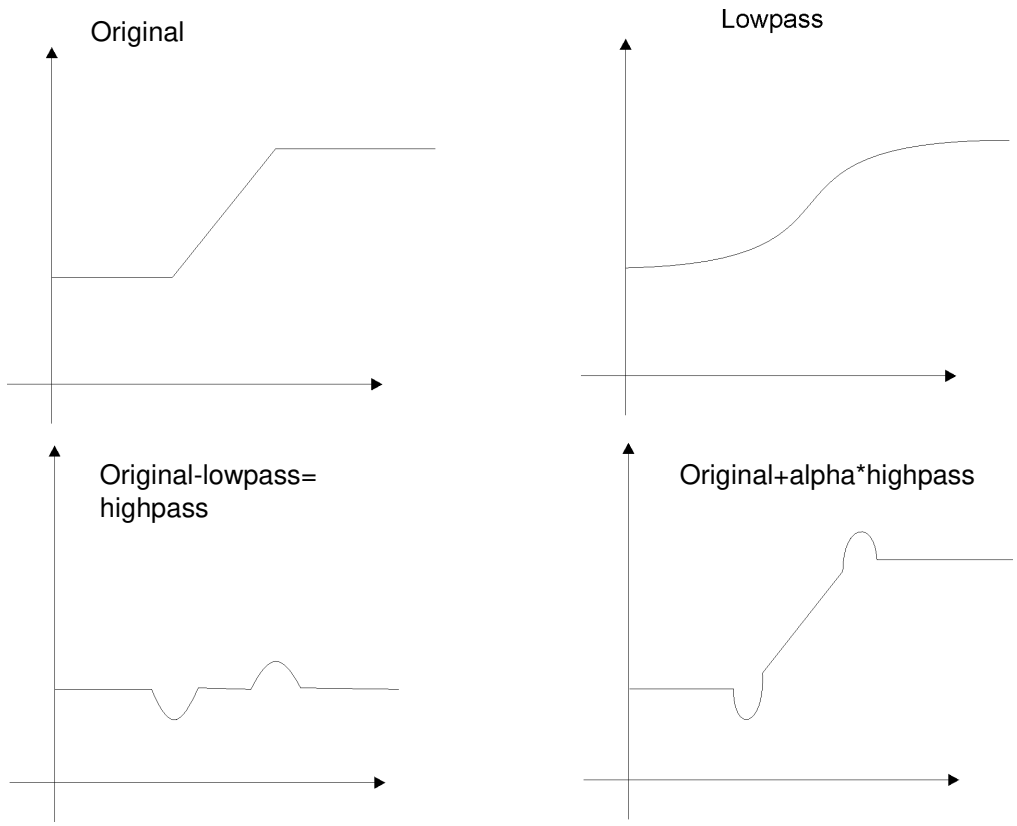
Load the greyscale image *cameraman.tif*. Convert to double format. Add salt and pepper noise with the imnoise command. Now use the medianfilter command to apply median filtering. Look at the result. Also look at the implementation of the medianfilter command.

3.2 Unsharp Masking

Unsharp masking is an operation that is applied to strengthen the edges in an image. The operation takes the original image and adds a signal which is proportional to the difference between the original image and a lowpass filtered version of this image. This is equivalent to taking the original image and adding a highpass filtered version of this image. The unsharp masking operation is described as:

$$u(m,n) = v(m,n) + \lambda \cdot g(m,n)$$

where v is the original image, u the image after unsharp masking and g the highpass filtered image. To obtain g the laplacian is often used. λ is always larger than 0. The figure below helps to understand the operation:



3.2.1 Exercise 13 - Unsharp masking

Load the image *aerial.tif* and convert it to double. The image has very low contrast. Therefore, apply histogram stretching first (use imadjust). Look at the image again (notice the weak edges). Use *imsharpen* to apply an unsharp masking filter on the image and look at the result.

3.2.2 Exercise 14 - Unsharp masking

Execute the process of unsharp masking by yourself (see 3.2) on *aerial.tif* (after conversion and imadjust). Use the *sharp* filter as high–pass filter in the procedure. Apply it to the image with *conv2(image,filter,'same')* and compare the result with the original.

$$sharp = \frac{1}{9}\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

3.2.3 Exercise 15 - Unsharp masking

Check the influence of noise on the unsharp masking operation. Use the greyscale image *aerial.tif.* After the usual conversion to double, apply histogram stretching and add Gaussian noise. Apply unsharp masking on the resulting image. Look at the result of the unsharp masking operation. What do you notice? Try to reduce this effect by suppressing the noise with the *uniform* filter before applying unsharp masking. Look at the result. What is your conclusion?

$$uniform = \frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

3.2.4 Wiener filtering

The Wiener filter is used for deblurring an image in the case when the blur kernel (point-spread function) is known.

Follow the next demo for a brief overview:
https://nl.mathworks.com/help/images/examples/deblurring-images-using-a-wiener-filter.html
OR in MATLAB command:
openExample('images/WienerImageDeblurringExample')