Spatial domain methods

There are mainly two methods for image-enhancement: one deals with images in the spatial domain, the other one deals with images in the frequency domain. The first method is based on the processing of individual pixels in an image, the second is based on modifying the Fourier transform of an image.

Spatial domain methods

Here, image processing functions can be expressed as :

$$g(x, y) = T(f(x, y)),$$

with f(x,y) the input image, g(x,y) the processed image (i.e. the result or output image) and an operator on f , defined over some neighbourhood N of (x,y). For N we mostly use a rectangular subimage that is centered at (x,y).

a) N is a 1×1 neighbourhood (point-processing)

In this case, N encompasses exactly one pixel. The operator T then becomes a gray-level transformation function, which we express as :

$$s = T(r),$$

with r,s the gray-levels of f(x,y) and g(x,y). Using this technique, we can achieve some interesting effects like contrast-stretching and bi-level mapping (here an image is converted so that it only contains black and one color white). The trick is to define T such that it darkens the gray-levels under a certain treshold k and brightens the gray-levels above this treshold. If the darkening and brightening are constants (black and white), a black-and-white image is produced. Because s is only dependent on the value (i.e. the gray-level) of T in 1 pixel, this technique is called ‘point-processing’.

b) N is a m×m neighbourhood (spatial filtering)

In this case, N encompasses a small region. Note that this method is not restricted to doing only enhancement, but it can also be used to smoothen images, etc. The general approach is that the value of g(x,y) is determined by the values of in a predefined neighbourhood (i.e. the mask/filter) of (x,y). Typical values for m range from 3 to even 10. The general name for these processes is ‘mask processing’ or ‘filtering’.

Frequency domain methods

These methods are principally based on the convolution theorem. It can be understood as follows :

Suppose g(x,y) is an image formed by the convolution of an image f(x,y) and linear, position invariant operator h(x,y):

$$g(x, y) = h(x, y) * f(x, y).$$

Applying the convolution theorem yields :

$$G(u, v) = H(u, v) \cdot F(u, v),$$

in which F ,G and H are the Fourier transforms of f, g and h respectively. Applying the inverse Fourier transform on G(u,v) yields the output image :

$$g(x, y) = \mathcal{F}^{-1}(H(u, v) \cdot F(u, v))$$

An example is H(u,v) which emphasizes the high-frequency components of f(u,v) so that g(x,y) becomes an image in which the edges are accentuated.

Viewed from the theory of linear systems (see figure 1), some interesting features can be seen : h(x,y) is called a system whose function is to produce an out-put image g(x,y) from an input image f(x,y). Equivalent with this is the Fourier notation of this operation.

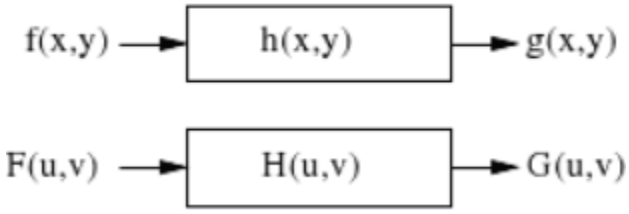


Figure 1: Linear systems.

Grey scale manipulation and histogram equalization

Grey scale manipulation

The simplest form of operation is when the operator *T* only acts on a pixel neighbourhood in the input

image, that is only depends on the value of *F* at (*x,y*). This is a *grey scale transformation* or mapping. The simplest case is thresholding where the intensity profile is replaced by a step function, active at a chosen threshold value. In this case any pixel with a grey level below the threshold in the input image gets mapped to 0 in the output image. Other pixels are mapped to 255. Other grey scale transformations are outlined in figure 1 below.

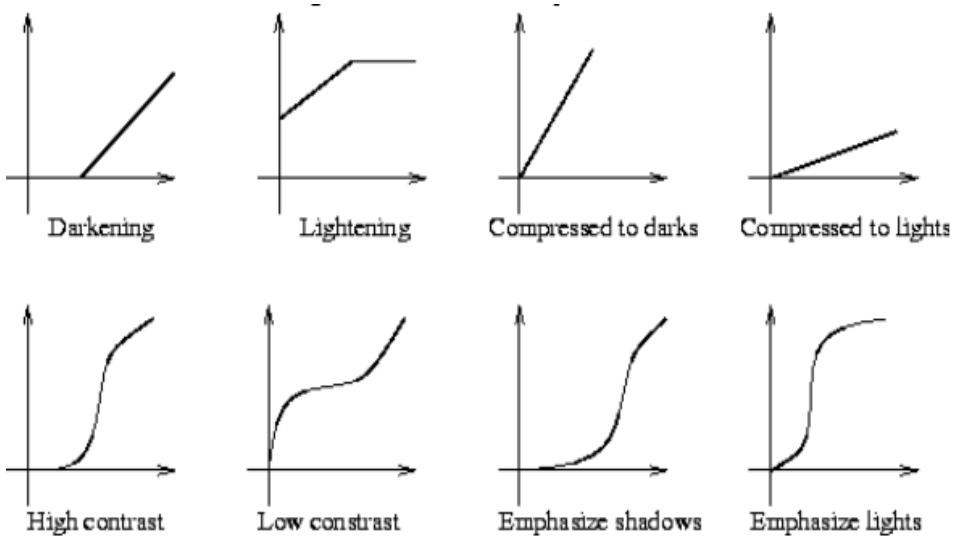


Figure 1: Tone-scale adjustments.

Histogram equalization

Histogram equalization is a common technique for enhancing the appearance of images. Suppose we have an image which is predominantly dark. Then its histogram would be skewed towards the lower end of the grey scale and all the image detail is compressed into the dark end of the histogram. If we could `stretch out' the grey levels at the dark end to produce a more uniformly distributed histogram then the image would become much clearer.

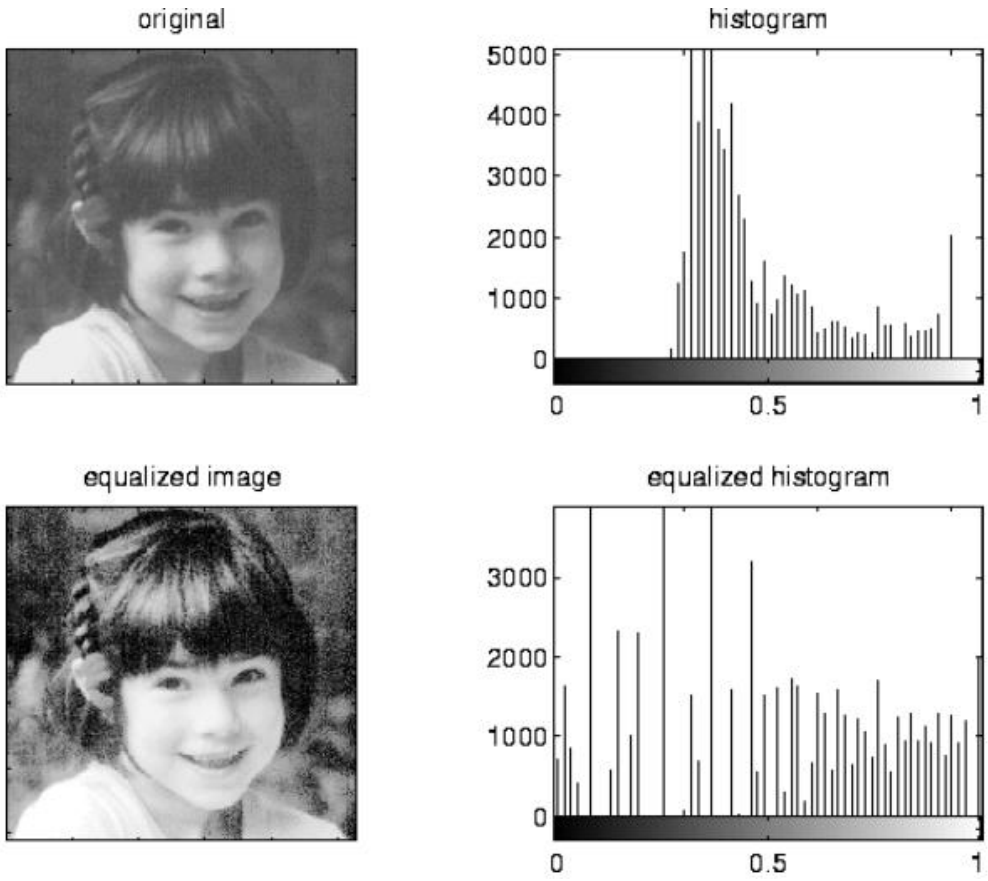


Figure 2: The original image and its histogram, and the equalized versions. Both images are quantized to 64 grey levels.

Histogram equalization involves finding a grey scale transformation function that creates an output image with a *uniform histogram.* We must find a transformation *T* that maps grey values *r* in the input image *F* to grey values *s* = *T*(*r*) in the transformed image .

It is assumed that

- T is single valued and monotonically increasing, and
- 0<= T(r)<=1 for 0<=r<=1.

Image subtraction and Image averaging

Image subtraction
This technique has numerous applications in image enhancement and segmentation (where an image is decomposed into several 'interesting' pieces like edges and regions). The fundamentals are based on the subtraction of two images defined as the computation of the difference between every pair of corresponding pixels in the two images. It can be expressed as :

- $$g(x, y) = f(x, y) - h(x, y)$$

An interesting application is used in medicine : h(x,y) is then called a mask which is subtracted from a series of images $f_i(x,y)$ after which some interesting images are obtained. As an example, by doing so, it is possible to watch a dye propagate through a person's head-arteries. Each time the difference is calculated, the regions in the images and that look the same are darkened, whilst the differences become more accentuated (they are not subtracted out of the resulting image).

Image averaging
Suppose you have a noisy image g(x,y), formed by the addition of a certain amount of noise *n*(x,y) to an original image f(x,y) :

- $$g(x, y) = f(x, y) + \eta(x, y)$$

It is assumed that at every pair (x,y) the noise is uncorrelated (thus uniform over the image) and has an average value of zero. The goal is to reduce the noise-effects by adding a set of noisy images {g(x,y)}
Suppose we have an image formed by averaging Å noisy images :

- $$\overline{g}(x, y) = \frac{1}{M} \sum_{i=1}^{M} g_i(x, y).$$

We now calculate the expected value of $\overline{g}$ which is :

- $$\begin{aligned} E\{\overline{g}(x, y)\} &= E\left\{\frac{1}{M} \sum_{i=1}^{M} g_i(x, y)\right\} \\ &= \frac{1}{M} E\left\{\sum_{i=1}^{M} g_i(x, y)\right\} \\ &= \frac{1}{M} \sum_{i=1}^{M} E\{g_i(x, y)\} \end{aligned}$$

Spatial filtering

This technique treated each pixel at location (x, y) in the input image f(x, y) separately. The spatial mask that defined a neighborhood for each pixel so that local enhancement could be done. In this concept of local enhancement is extended to the idea of using filters so that specific effects like the blurring/smoothing and sharpening of an image can be achieved.

Basically, there are two types of filters : linear filters and nonlinear filters.

1. Linear filters

Linear filters are based on the concepts discussed in the introductory chapter which made notion of the concepts impulse response function and transfer function. The first is the Fourier transform of the second. There are mainly three types of linear filters :

- **lowpass filters** : (see figure 1) these filters eliminate high frequencies, resulting in the removal of edges and sharpness in images, thus blurring/- smoothing the overall image,
- **highpass filters** : (see figure 2) these filters eliminate low frequencies, resulting in sharper images (the edges are more pronounced),
- **bandpass filters** : (see figure 3) these filters are a combination of the above two types. They are mostly used for image restoration, rather than for image enhancement.

In the following figures, the curves on the left are the frequency domain filters and those on the right are their corresponding spatial domain filters.

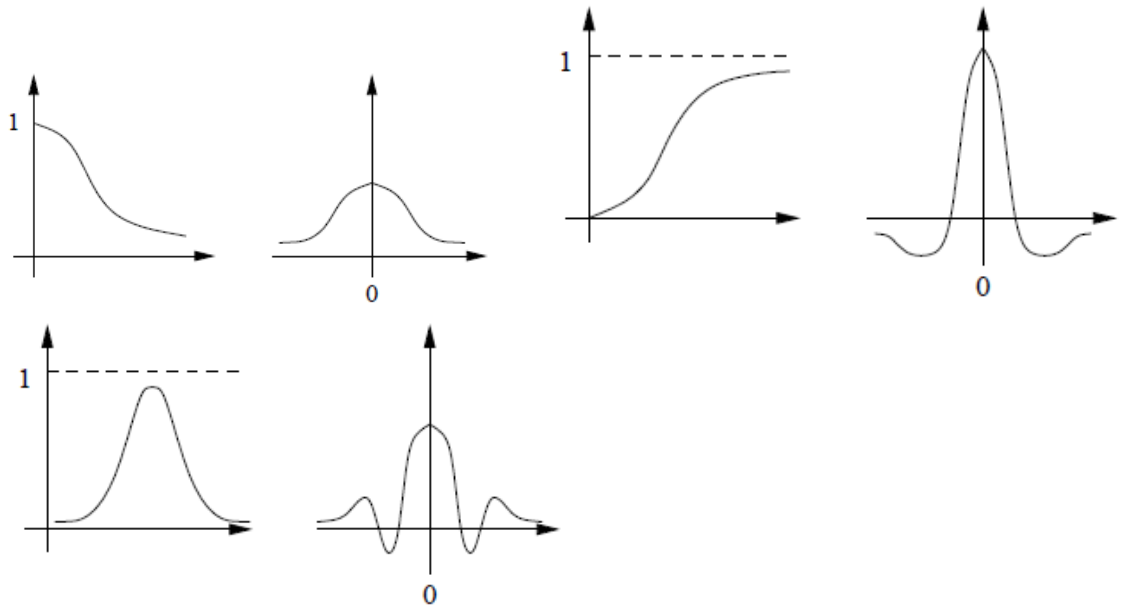


Figure: (1) lowpass filters (2) highpass filters (3) bandpass filters.

A linear filter operating on an M by N neighbourhood is defined by M×N mask coefficients. These coefficients are used in a sum that is calculated for each
pixel at location (x; y) in the input image :

$$R = w_1 z_1 + w_2 z_2 + \ldots + w_k z_k,$$

where k is the number of coefficients in the mask (e.g. a 3× 3 mask has 9 coefficients), z_1,z_2,..... z_k are the gray-levels of the pixels under the mask, w_1, w_2, ...,w_k are the coefficients (also called weights) of the mask and R is the response of the linear mask. The goal is to replace the gray-level of the pixel at location (x; y) by the value calculated for R. This is done for each pixel in the input image, thus the mask is moved from pixel to pixel.

Nonlinear filters

These filters operate also on some neighbourhood of every pixel at a location (x; y), however, the nonlinearity is expressed in the computation of R which can now be for example the maximum gray-level, the median gray-level or the minimum gray-level of all the pixels in the neighbourhood.

$$
\begin{aligned}
&= \frac{1}{M}\sum_{i=1}^{M} E\{f(x,y)+\eta_i(x,y)\} \\
&= \frac{1}{M}\sum_{i=1}^{M} (E\{f(x,y)\}+E\{\eta_i(x,y)\}) \\
&= \frac{1}{M}\sum_{i=1}^{M} (f(x,y)+0) \\
&= \frac{1}{M} M f(x,y) \\
&= f(x,y).
\end{aligned}
$$

Smoothing filters

Smoothing filters are used in image blurring and noise reduction. The first is a preprocess-technique that may remove small details from images so that at a later time object extraction can be done (this has to do with image segmentation).

Lowpass spatial filtering

In the spatial domain, all the coefficients of this filter has are positive. A Gaussian filter can be used, but in most cases good results are achieved when all the coefficients have the same positive value. For example, consider a 3 × 3 filter : here all the coefficients have a value of 1. But there is a problem when designing the filter like this. When R is computed, the response would be the sum of the gray-levels for nine pixels, which has the effect that R lies outside the valid gray-level range. A solution to this problem is quickly find : divide all the coefficients by a factor of nine. Generally, a M ×N filter has for its coefficients the values 1=(M × N).

Note that the response R is always the average of all the pixels in the considered neighbourhood. For this reason, lowpass spatial filtering is also called neighbourhood averaging.

Median filtering

When blurring/smoothing images, the previous method scores very well. When however, our goal is to reduce the noice in a certain image, the lowpass method fails (because it only blurs). The solution is to use a median filter which replaces the gray-level of each pixel by the median of the gray-levels of all the pixels in the neighbourhood.
Median filtering is very effective in removing sharp 'spikes' from an image. When noise is introduced in an image, lowpass filtering just 'blends' the noise, whilst median filtering succeeds in removing most of the noise. If necessary, several passes with a median filter may be needed.

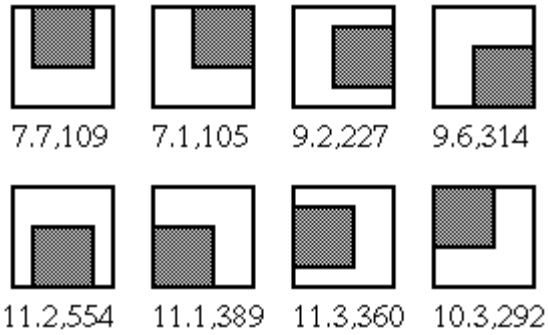
Note that median filtering actually forces pixels with very distinct gray-levels to have a gray-level that is more like its neighbours. Also note that when the noise is very concentrated in an area, it is possible that a median filter can not filter out all the noise. In this case, more passes of the filter are needed.

Nagao's algorithm

- Define a set of K neighborhoods around the pixel under consideration, for example, the eight 3x3 neighborhoods in N, NE, E, SE, S, SW, W, and NW directions. The pixel should be included as a corner in each of these neighborhoods.
- Find the mean and variance for each of the K neighborhoods.
- Replace pixel by one of the means corresponding to the smallest :

$$y[m,n] = \mu_i, \quad \text{iff} \quad \sigma_i^2 = min\{\sigma_1^2, \cdots, \sigma_K^2\}$$

10	22	3	4	10
17	5	10	9	9
11	10	5	2	12
12	26	6	21	9
3	10	17	4	10



In this example, the pixel in question is replaced by the mean 7.1 from the northeast neighborhood with minimum variance 105.