University of Antwerp
Department of Mathematics and Computer Science

An introduction to image enhancement in the spatial domain.

Sven Maerivoet

November, 17th 2000

Contents

Chapter 1

Introduction

Fourier transformations are nice, but the real fun begins when they can be used in the process of enhancing existing images. The results of these enhancements are dependent on the type of a specific application, which means that they are very much problem-oriented.

There are mainly two methods for image-enhancement : one deals with images in the *spatial domain*[1] (say $f(x, y)$), the other one deals with images in the *frequency domain* (say $F(u, v)$). 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.

1.1 Spatial domain methods

Here, image processing functions can be expressed as :

$$g(x, y) = T(f(x, y)), \tag{1.1}$$

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

1.1.1 $\mathcal{N}$ is a 1×1 neighbourhood (point-processing)

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

[1] Spatial domain refers to the aggregate of pixels composing an image.

$$s = T(r), \tag{1.2}$$

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'*.

1.1.2 $\mathcal{N}$ is a $m \times m$ neighbourhood (spatial filtering)

In this case, $\mathcal{N}$ encompasses a small region. Note that this method isn't 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 f 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'*.

1.2 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 a linear, position invariant operator $h(x, y)$:

$$g(x, y) = h(x, y) * f(x, y). \tag{1.3}$$

Applying the convolution theorem yields :

$$G(u, v) = H(u, v) \cdot F(u, v), \tag{1.4}$$

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 :

$$g(x, y) = \mathcal{F}^{-1}(H(u, v) \cdot F(u, v)). \tag{1.5}$$

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 1.1), some interesting features can be seen : $h(x,y)$ is called a system whose function is to produce an output image $g(x,y)$ from an input image $f(x,y)$. Equivalent with this is the Fourier notation of this operation.

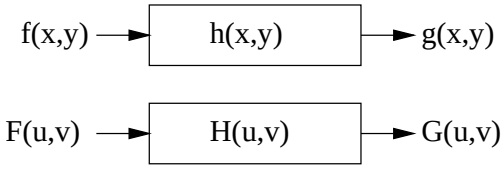

Figure 1.1: Linear systems.

Now suppose that $h(x,y)$ is unknown and that we apply a *unit impulse function* $\delta(t)$ (*this is the 1D-case*) :

$$\begin{aligned} \delta(t) &= 0 \qquad \text{if } t \neq 0 \\ \delta(t) &= +\infty \qquad \text{if } t = 0. \end{aligned} \tag{1.6}$$

Note that $\delta(t)$ is not a real function, it is only defined indirectly :

$$\begin{aligned} \int_a^b x(t)\delta(t-t_0)dt &= x(t_0) \qquad \text{if } a < t_0 < b \\ &= 0 \qquad \text{else.} \end{aligned} \tag{1.7}$$

If we calculate it's Fourier transform, we obtain 1 for $F(u,v)$. So in the Fourier system of equation 1.4, we multiply $H(u,v)$ with the constant 1 which of course results in $H(u,v)$. From this we conclude that $G(u,v) = H(u,v)$. Applying equation 1.5 we obtain that $g(x,y) = h(x,y)$. Thus, the system is completely specified by its response to a unit impulse function.

In the above derivation we call $h(x,y)$ the *impulse response function* and $H(u,v)$ the *transfer function*. Equation 1.3 describes a process called *spatial convolution* and $h(x,y)$ is sometimes referred to as the *spatial convolution mask* (note that the mask must be symmetric about it's origin because a convolution involves flipping an image about the origin).

1.3 Conclusion

The material presented in this chapter shows what different methods are available for image enhancement. Important to know is that usually a certain amount of trail-and-error is involved in settling on an approach for image processing.

The remainder of this seminar focuses on enhancement in the spatial domain and describes the two types of methods : pixels can be treated independently of their neighbours (‘*point processing*’), or as being dependent on their neighbours (‘*spatial filtering*’).

Chapter 2

Point processing

As described earlier, the techniques presented in this chapter focus on considering methods that are based only on the intensity of single pixels. I use the notation s and r to describe the intensity of the pixels before and after processing respectively.

2.1 Simple intensity transformations

2.1.1 Image negatives

The transformation is very simple :

$$s = T(r), \tag{2.1}$$

with

$$T(r) = (L - 1) - r. \tag{2.2}$$

Here L is the number of gray-levels in the input image. The result of this transformation is that low intensities are made high and vice versa.

2.1.2 Contrast stretching

This technique can enhance low-contrast images. The idea is to increase the dynamic range of the gray-levels by using a piecewise linear curve. This curve $T(r)$

can be a *tresholding function* that creates a binary image. The main goal of contrast stretching is to group gray-levels. Note that $T(r)$ should be single valued and monotonically increasing so that it preserves the order of gray-levels.

2.1.3 Compression of dynamic range

When the dynamic range of a processed image can't be fully displayed, a way to compress the dynamic range of the pixel values is available :

$$s = c \log(1+ \mid r \mid), \tag{2.3}$$

with c a scaling constant. The desired compression is performed by the logarithmic function. An example of its use is when a Fourier spectrum is scaled linearly for displaying on an 8-bit system : the brightest values will dominate the display. After transforming and scaling a significant increase in visible detail can be the result.

2.1.4 Gray-level slicing

When we want to highlight a specific range of gray-levels in an image, there are mainly two methods available :

- give a high value for all the gray-levels in the specified range and a very low value for all the other gray-levels,
- or give a high value for all the gray-levels in the specified range and *preserve* the background and gray-level tonalities in the image.

In both methods, the desired range is made constant in intensity. The first methods turns all the other gray-levels in a low constant, whereas the second method keeps their values by using a linear function.

2.1.5 Bit-plane slicing

When the intensity of each pixel of an image is defined by several bits, we may highlight some specific intensities by considering only those pixels in the image that have certain bits set. This technique results in a decomposition of an image into several 'bit-planes' that each contribute to the total image. One may find that

for example the highest order bits are dominant in the picture whereas the lowest order bits don't give visually significant extra information (and image compression can thus be done using this technique).

2.2 Histogram processing

This technique is based on statistical methods. The main theme is the notion of the *histogram of a digital image*, which is defined as a discrete function $p(r_k) = n_k/n$ with r the kth gray-level (which lies in the range $[0, L-1]$), n_k the number of pixels in the image with that gray-level and n the total number of pixels in the image.

By this definition, $p(r_k)$ gives an estimate of the probability of occurrence of gray-level r_k in the digital image. Plotting this function gives a visual indication of the image : dark (most gray-levels are low), bright (most gray-levels are high), low-contrast (the gray-levels are very localized resulting in a small dynamic range) or high-contrast (the gray-levels are spread out resulting in a wider dynamic range). The shape of the histogram can thus be an indication of whether or not image enhancement is possible.

2.2.1 Histogram equalization

Suppose r represents the gray-levels in the image to be enhanced. For now assume that all the possible gray-levels are continuous and lie in the interval $[0, 1]$ (and thus $r = 0$ is black and $r = 1$ is white). Assume $s = T(r)$ and the following conditions hold : (a) $T(r)$ is single-valued and monotonically increasing for $0 \leq r \leq 1$ and (b) $0 \leq T(r) \leq 1$ for $0 \leq r \leq 1$. The inverse transformation T^{-1} is denoted by $r = T^{-1}(s)$ for $0 \leq s \leq 1$ (and it is assumed that T^{-1} also satifies the above conditions (a) and (b)).

If we move to the domain of statistics, we can look at the gray-levels as continuous random values in the interval $[0, 1]$, which can be characterized by their *probability density functions* (PDF) $p_r(r)$ and $p_s(s)$ (and thus p_r and p_s are different functions).

It follows that – knowing p_r and $T(r)$ and T^{-1} satisfies the above condition (a) – the probability density function of the transformed gray-levels can be expressed as :

$$p_s(s) = \left[p_r(r) \frac{dr}{ds} \right]_{r=T^{-1}(s)} . \tag{2.4}$$

Knowing this, we now modify the appearance of the image by controlling the probability density function of its gray-levels via the transformation function $T(r)$.

Consider the transformation function :

$$s = T(r) = \int_0^r p_r(w)dw \quad \text{with } 0 \leq r \leq 1, \tag{2.5}$$

with w a dummy-variable needed for the integration. Equation 2.5 is also known as the *cumulative distribution function* (or CDF for short) of r. If we take the derivative of s with respect to r we obtain :

$$\frac{ds}{dr} = p_r(r). \tag{2.6}$$

Substituting this equation for dr/ds into equation 2.4 yields :

$$p_s(s) = 1 \quad \text{with } 0 \leq s \leq 1, \tag{2.7}$$

which is a uniform density function. Note that the result is *independent* of the inverse transformation function. This last remark is very important since analytically obtaining $T^{-1}(s)$ is not always easy.

Thus using a transformation function equal to the CDF of r produces an image whose gray-levels have a uniform density. The result is an increase in the dynamic range of the pixels that can significally contribute to the image's visual enhancement.

Now in order to be useful in digital image processing, the previous technique must be formulated in its *discrete form*. The probabilities (of the discrete gray-levels) are now :

$$p_r(r_k) = \frac{n_k}{n} \quad \text{with } 0 \leq r_k \leq 1 \text{ and } k = \{0, \ldots, L-1\}, \tag{2.8}$$

with L the number of gray-levels in the image. The discrete form of equation 2.5 is :

$$\begin{aligned} s_k &= T(r_k) = \sum_{j=0}^{k} \frac{n_j}{n} \\ &= \sum_{j=0}^{k} p_r(r_j) \quad \text{with } 0 \leq r_k \leq 1 \text{ and } k = \{0, \ldots, L-1\}. \end{aligned} \tag{2.9}$$

The inverse transformation is denoted by :

$$r_k = T^{-1}(s_k) \quad \text{with } 0 \leq s_k \leq 1. \tag{2.10}$$

The transformation function $T(r_k)$ can be computed directly from the image by using equation 2.9.

The main advantage of this technique (histogram equalization) is that pictures with a very poor dynamic range (thus having very small variations in their gray-levels) can be enhanced such that all their original gray-levels are 'smeared out' in the spectrum from black to white (note that white is always reached). This leads to more visible details in the output image. A disadvantage of this method is that visual artefacts can be introduced resulting in graininess and patchiness.

2.2.2 Histogram specification

One critique at the previous technique is that the method doesn't lend itself to interactive image enhancements. By interactive I mean choosing the kind of histogram the output image will have (in section 2.2.1 the resulting histogram was *always* approximated uniform). More interactive processing means that we'll be able to highlight certain gray-levels (like in section 2.1.4).

To see how this is done, suppose we are working again with continuous gray-levels and that $p_r(r)$ and $p_z(z)$ are the original and desired probability density functions respectively. Now suppose we apply histogram equalization on the original image (by using equation 2.5) and that we have the desired image available. We could also equalize this last image's gray-levels by applying equation 2.5. The result of all this is that we now have obtained $s = T(r)$ and $v = G(z)$.

To retrieve the gray-levels of z, we apply the inverse process $z = G^{-1}(v)$. Now remember that these gray-levels z are precisely the gray-levels we are searching for ! Because histogram equalization always results in the same uniform histogram, $p_s(s)$ and $p_v(v)$ are equal. If, in the inverse process, we now use the uniform gray-levels s obtained from the original image instead of using v, the resulting gray-levels $z = G^{-1}(s)$ would have the desired probability density function. If $G^{-1}(s)$ is single-valued, we can summarize as follows :

1. equalize the gray-levels of the original image,
2. specify the desired probability density function and obtain the transformation function using equalization on the desired image,

3. apply the inverse transformation function to the gray-levels obtained in the first step.

The result is an output image with its gray-levels characterized by a certain specified probability density $p_z(z)$.

Note that two transformation functions are needed with this technique (namely first $T(r)$ and then $G^{-1}(s)$, they can be simply combined into one single function :

$$z = G^{-1}(s) \Rightarrow z = G^{-1}(T(r)). \tag{2.11}$$

Note that although equation 2.11 is rather easily applied to discrete variables, it can become a painstaking task when dealing with continuous variables, because then the results have to be derived analytically.

Histogram specification as a means for image enhancement has its own difficulty, namely constructing a meaningful desired histogram. One solution is to sample a function (such as the well-known Gauss-curve) as the histogram to use, another solution is to manually specify the shape of the histogram and calculating the histogram from this information.

Histogram specification generally yields better results than histogram equalization. The latter one generates mostly higher contrast images while the former one results images with a more balanced appearance. The big advantage of specification over equalization is the higher degree of flexibility.

2.2.3 Local enhancement

The techniques described in sections 2.2.1 and 2.2.2 were considered from a global viewpoint : the transformation of one pixel depends on the histogram of the overall image. Overall enhancement works great in this way, but sometimes local enhancement is desired (i.e. enhancing details over small areas). The number of pixels in those small areas have almost no influence on the computation of the global transformation so the result is not always local enhancement.

Using histograms

The trick to get around this problem is to use small rectangular masks. The center of such a mask is moved from pixel to pixel over the entire image and in each step a small neigbourhood of a pixel is examinated. This examination involves calculating the histogram for the set of pixels in this neighbourhood (note that

several methods exist to optimize the calculation because most of the time the mask moves one row or one column at a time and most of the information in the histogram can be re-used). Afterwards, histogram equalization or specification can be used as the result for the pixel under consideration.

Using the mean and variance

Instead of using histograms, one can also use the *mean* and *variance* of the pixel intensities for a pixel considered in some defined neighbourhood. The mean is then known as a measure for average brightness, whilst the variance is then known as a measure for contrast.

Such a (local) transformation can be described as :

$$g(x,y) = A(x,y) \cdot [f(x,y) - m(x,y)] + m(x,y), \tag{2.12}$$

with

$$A(x,y) = k\frac{M}{\sigma(x,y)} \quad \text{with } 0 < k < 1. \tag{2.13}$$

In the above equations, $f(x,y)$ is the input image, $g(x,y)$ is the output image, $m(x,y)$ is the gray-level mean (computed in a neighbourhood centered at (x,y)), $\sigma(x,y)$ is the gray-level standard deviation (computed in the same region), M is the global mean of $f(x,y)$ and k is a constant.

Thus, the values of A, m and σ depend on the values of the pixels in the neighbourhood of (x,y). Considering equations 2.12 and 2.13, we see that applying the *gain factor* A to the difference results in the amplifciation of local variations. Because this A is inversely proportional to the standard deviation (or squared, the variance) of the intensity, the result is that areas with low contrast receive a larger gain (because then σ is smaller so $1/\sigma$ is larger). The final term $m(x,y)$ is added because the intensity levels need to be restored to the average level in the neighbourhood.

2.3 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 :

$$g(x,y) = f(x,y) - h(x,y). \tag{2.14}$$

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 f_i and h that look the same are darkened, whilst the differences become more accentuated (they are not subtracted out of the resulting image).

2.4 Image averaging

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

$$g(x,y) = f(x,y) + \eta(x,y). \tag{2.15}$$

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 $\{g_i(x,y)\}$.

Suppose we have an image formed by averaging M noisy images :

$$\overline{g}(x,y) = \frac{1}{M}\sum_{i=1}^{M} g_i(x,y). \tag{2.16}$$

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

$$\begin{aligned} E\{\overline{g}(x,y)\} &= E\left\{\frac{1}{M}\sum_{i=1}^{M} g_i(x,y)\right\} \\ &= \frac{1}{M}E\left\{\sum_{i=1}^{M} g_i(x,y)\right\} \\ &= \frac{1}{M}\sum_{i=1}^{M} E\{g_i(x,y)\} \end{aligned} \tag{2.17}$$

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

Note that I assumed that in equation 2.17 each $g_i(x,y)$ is formed by a specified noise $\eta_i(x,y)$. For the variance it follows that :

$$\sigma^2_{\overline{g}(x,y)} = \frac{1}{M}\sigma^2_{\eta(x,y)}. \tag{2.18}$$

The former result can be proven by using the relation $\sigma^2(X) = E\{(X - E\{X\})^2\}$. Taking the square root of both sides in equation 2.18 yields :

$$\sigma_{\overline{g}(x,y)} = \frac{1}{\sqrt{M}}\sigma_{\eta(x,y)}. \tag{2.19}$$

We can clearly see that if, in equation 2.19, M increases, the standard deviation σ decreases and thus the variability of the pixels at each location (x,y) decreases. Now because the expected value of $\overline{g}$ is actually f, it follows that $\overline{g}$ approaches f when the number of noisy images used increases.

A popular implementation is used in electron-microscopy at the EMAT (RUCA), it was devised by prof. dr. Dirk Van Dyck.

Chapter 3

Spatial filtering

The previous chapter concerned itself with image enhancement by using point processing. This technique treated each pixel at location (x, y) in the input image $f(x, y)$ separately. The concept introduced in section 2.2.3 used a spatial mask that defined a neighbourhood for each pixel so that local enhancement could be done. In this chapter, 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.

3.1 Introduction

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

3.1.1 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 3.1) 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 3.2) these filters eliminate low frequencies, resulting in sharper images (the edges are more pronounced),

- **bandpass filters** : (see figure 3.3) 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.

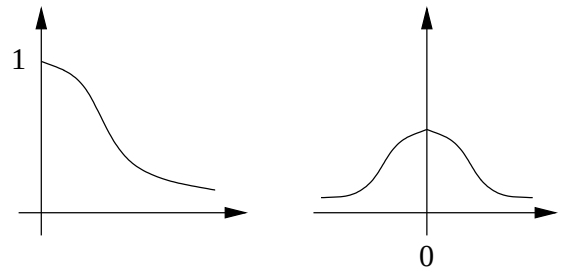

Figure 3.1: A lowpass filter

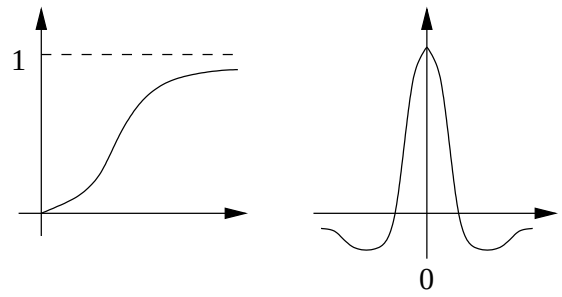

Figure 3.2: A highpass filter

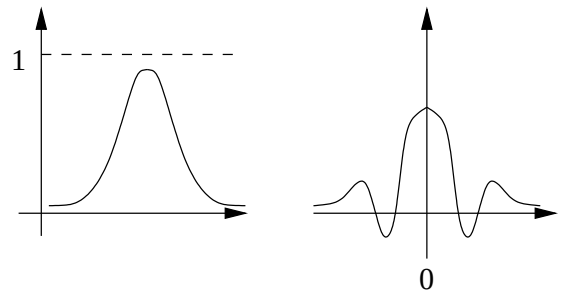

Figure 3.3: A bandpass filter

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

$$R = w_1 z_1 + w_2 z_2 + \ldots + w_k z_k, \tag{3.1}$$

where k is the number of coefficients in the mask (e.g. a 3×3 mask has 9 coefficients), $z_1, z_2, \ldots, z_k$ are the gray-levels of the pixels under the mask, $w_1, w_2, \ldots, w_k$ 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.

3.1.2 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.

3.2 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).

3.2.1 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 $M \times N$ filter has for its coefficients the values $1/(M \cdot N)$.

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*.

3.2.2 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[1] 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.

3.3 Sharpening filters

When sharpening an image, what we actually are trying to do is highlighting some specific details in the image (for example to increase the contrast by accentuating the edges). In what follows, several methods are discussed.

3.3.1 Basic highpass spatial filtering

When looking at figure 3.2, we can see that the filter in the spatial domain has a positive value near its center and negative values in its neighbourhood. A classic implementation of a 3×3 filter looks as follows :

$-\frac{1}{9}$	$-\frac{1}{9}$	$-\frac{1}{9}$
$-\frac{1}{9}$	$\frac{8}{9}$	$-\frac{1}{9}$
$-\frac{1}{9}$	$-\frac{1}{9}$	$-\frac{1}{9}$

Note that the sum of the coefficients is zero : this means that when the mask is operating on an area with slowly varying gray-levels, the output of the mask is zero (or very small) and thus low frequencies are removed from the image. Also

[1]The median of a set of values $\{x_1, x_2, \ldots, x_n\}$ is the value x_i that is found in the middle when partially ordering all the values. If n is even, the median is the average of the two values in the middle.

note that the zero-frequency term is eliminated by such a filter. The result is that global contrast of the image is significantly reduced.

Very important is the fact that when reducing the average value of an image's gray-levels to zero, there are some negatively valued gray-levels. Since only positive values are allowed, scaling and/or clipping needs to be done to get the resulting gray-levels in a valid interval (say $[0, L-1]$ with L the number of gray-levels).

3.3.2 High-boost filtering

Remember the problem with the previous method ? Negative gray-levels were introduced in the result. To conquer this problem several methods are available. However, there is an alternative and that is to use *high-boost filtering*. With this technique, high frequencies are intensified whilst the other, lower, frequencies are relatively maintained (whereas with basic highpass filtering these values tend to become zero, resulting in images with a dark background).

The approach is this : a highpass filtered image can be viewed as the difference of an original image and a lowpass filtered version :

$$\text{highpass} = \text{original} - \text{lowpass}. \tag{3.2}$$

The above 'equation' is valid. This can be seen by substituting the values of the respective filters :

$$\frac{8}{9}z_5 + \sum_{i=1, i\neq 5}^{9} -\frac{1}{9}z_i = \text{original} - \sum_{i=1}^{9}\frac{1}{9}z_i, \tag{3.3}$$

which results in :

$$\frac{8}{9}z_5 = \text{original} - \frac{1}{9}z_5, \tag{3.4}$$

and thus :

$$z_5 = \text{original}, \tag{3.5}$$

which is of course true. Now multiplying the original image by an amplification factor A yields the sought high-boost filter :

$$\begin{aligned} \text{high-boost} &= (A)\text{original} - \text{lowpass} \\ &= (A-1)\text{original} + (\text{original} - \text{lowpass}) \\ &= (A-1)\text{original} + \text{highpass}. \end{aligned} \tag{3.6}$$

When $A = 1$ we get the already familiar highpass filter. When $A > 1$, the low frequencies of the original image are restored in the resulting image, which gives us the advantage that the resulting image will bare more resemblance to the original than was the case with basic highpass filtering. The first line of equation 3.6 is referred to as *unsharp masking*.

High-boost filtering has advantages over normal highpass filtering because the resulting image is not darkened, as is the case in the latter filter-method. However, boosting the higher frequencies may also result in the revealing of noise.

3.3.3 Derivative filters

The disadvantage of the previous methods is that they are based on averaging the gray-levels of the pixels in a region, which actually results in a blurring of the image. Because averaging uses a sum (and therefore – in the continuous case – it can also be expressed as *integrating*), one might expect that a *derivation* has the opposite effect (the result will be a sharper image).

Much of the time, differentation is expressed by using the gradient (nabla) :

$$\nabla \mathbf{f} = \tau \left[\frac{\partial f}{\partial x} \;\; \frac{\partial f}{\partial y} \right], \tag{3.7}$$

where $\nabla \mathbf{f}$ is the gradient of the function $f(x, y)$ at the coordinates (x, y) (note that the gradient is a *vector*).

If we compute the magnitude of this vector, we get :

$$\nabla f = \text{mag}(\nabla \mathbf{f}) = \left[\left(\frac{\partial f}{\partial x} \right)^2 + \left(\frac{\partial f}{\partial y} \right)^2 \right]^{\frac{1}{2}}. \tag{3.8}$$

Note that the magnitude in equation 3.8 is based on the Euclidean metric and that numerous approaches to image differentation are based on it. The general approach is to let equation 3.8 be approximated at a certain point (i.e. z_5) in the mask. Several approximations are :

$$\nabla f \approx \left[(z_5 - z_8)^2 + (z_5 - z_6)^2\right]^{1/2} \quad (3.9)$$

$$\nabla f \approx |\, z_5 - z_8 \,| + |\, z_5 - z_6 \,| \quad (3.10)$$

$$\nabla f \approx \left[(z_5 - z_9)^2 + (z_6 - z_8)^2\right]^{1/2} \quad (3.11)$$

$$\nabla f \approx |\, z_5 - z_9 \,| + |\, z_6 - z_8 \,| . \quad (3.12)$$

Note that in equations 3.10 and 3.12 the Manhattan-metric was used instead of the Euclidean metric. Note also that equations 3.9 and 3.10 differ from equations 3.11 and 3.12 in the way that the former two equations use the differences $(z_5 - z_8)$ for the x-direction and $(z_5 - z_6)$ for the y-direction, whilst the latter two equations use the cross-differences $(z_5 - z_9)$ and $(z_6 - z_8)$.

All four equations can be implemented by using masks of size 2×2. One method is to use *Roberts cross-gradient operators* (which are two masks, see tables 3.1 and 3.1). The response of these two masks (see below) is taken, the absolute values are computed and the results are summed.

1	0
0	-1

Table 3.1: Roberts masks (1)

0	1
-1	0

Table 3.2: Roberts masks (2)

But, as masks of even size (in the case of Roberts, 2) are awkward to implement, odd sizes are more commonly used. Examples are the 3×3 masks of Prewitt and Sobel (the Sobel-masks are used in image segmentation which encompasses the process of extracting geometrical information out of an image, like finding the edges). Keep in mind though, that these masks are used for approximating the magnitude of the gradient of f in (x, y), which is now expressed as :

$$\begin{aligned} \nabla f \approx\ & |\, (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3) \,| + \\ & |\, (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7) \,| . \end{aligned} \quad (3.13)$$

The difference between the rows approximates the magnitude of the gradient in the x-direction, the difference of the columns approximates the magnitude of the gradient in the y-direction.

Considering the Roberts, Prewitt and Sobel masks we also find that their coefficients sum to zero, as we could expect from a derivative filter which should give a response of zero in regions of a constant gray-level in an image.

Bibliography

[Dyc99] Dirk Van Dyck. *Inleiding tot de Digitale Communicatie*. UIA - Universitaire Instelling Antwerpen, 1999.

[GW93] Rafael C. Gonzalez and Ricard E. Woods. Image enhancement. In *Digital Image Processing*, pages 161–201. Addison-Wesley Publishing Company, 3rd edition, 1993.