Image Processing: The Fundamentals. Maria Petrou and Panagiota Bosdogianni

Print ISBN 0-471-99883-4 Electronic ISBN 0-470-84190-7

Chapter 4

Image Enhancement

What is image enhancement?

Image enhancement is the process by which we try to improve an image so that it looks **subjectively** better. We do not really know how the image should look, but we can tell whether it has been improved or not, by considering, for example, whether more detail can be seen, or whether unwanted flickering has been removed, or the contrast is better etc.

How can we enhance an image?

The approach largely depends on what we wish to achieve. In general, there are tw o major approaches: those which reason about the statistics of the grey values of the image, and those which reason about the spatial frequency content of the image.

Which methods of the image enhancement reason about the grey level statistics of an image?

- Methods that manipulate the histogram of the image for the purpose of increasing its contrast.
- The method of principal component analysis of a multispectral image for obtaining a grey level version of it with the maximum possible contrast.
- Methods based on rank order filtering of the image for the purpose of removing noise.

What is the histogram of an image?

The histogram of an image is a discrete function that is formed by counting the number of pixels in the image that hav e a certain grey value. When this function is normalized to sum up to 1 for all the grey level values, it can be treated as a probability density

function that expresses how probable is for a certain grey value to be found in the image. Seen this way, the grey value of a pixel becomes a random variable which takes values according to the outcome of an underlying random experiment.

When is it necessary to modify the histogram of an image?

Suppose that we cannot see much detail in the image. The reason is most likely that pixels which represent different objects or parts of objects tend to have grey level values which are very similar to each other. This is demonstrated with the example histograms shown in *Figure* 4.1. The histogram of the "bad" image is very narrow, while the histogram of the "good" image is more spread.

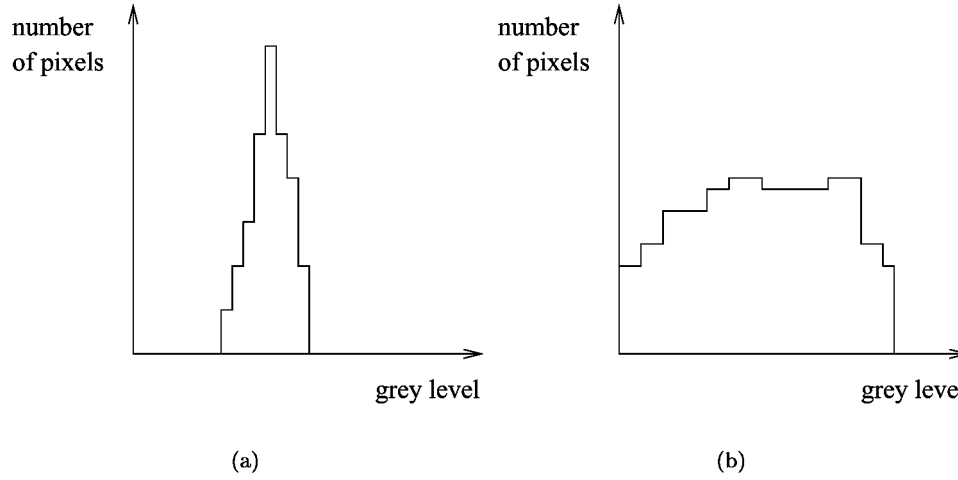


Figure 4.1: (a) is the histogram of a "bad" image while (b) is the histogram of a "good" image.

How can we modify the histogram of an image?

Suppose that the grey levels in the original image are given by the values a variable r obtains and in the new image by the values a variable s obtains. We would like to find a transformation $s = T(r)$ such that the probability density function $p_r(r)$ in *Figure* 4.1a is transformed into a probability density function $p_s(s)$ which looks like that in *Figure* 4.1b, say.

Since $p_r(r)$ is the probability density function of random variable r, the number of pixels with grey level values in the range r to $r + dr$ is $p_r(r)dr$. The transformation we seek will transform this range to $[s, s + ds]$. The total number of pixels in this range will remain the same but in the enhanced image this number will be $p_s(s)ds$:

$$p_s(s)ds = p_r(r)dr \tag{4.1}$$

This equation can be used to define the transformation T that must be applied to variable r to obtain variable s, provided we define function $p_s(s)$.

What is histogram equalization?

Histogram equalization is the process by which we make all grey values in an image equally probable, i.e. we set $p_s(s) = c$, where c is a constant. Transformation $s = T(r)$ can be calculated from equation (4.1) by substitution of $p_s(s)$ and integration. We integrate from 0 up to an arbitrary value of the corresponding variable, making use of the fact that equation (4.1) is valid for any range of values. These limits are equivalent of saying that we equate the **distribution** functions of the two random variables s and r:

$$\int_0^s c ds = \int_0^r p_r(r) dr \Rightarrow s = \frac{1}{c} \int_0^r p_r(x) dx \tag{4.2}$$

Here, in order to avoid confusion we replaced the dummy variable of integration by x. *Figures* 4.2a-4.2d show an example of applying this transformation to a low contrast image. Notice how narrow the histogram 4.2b of the original image 4.2a is. After histogram equalization, the histogram in 4.2d is much more spread, but contrary to our expectations, it is not flat, i.e. it does not look "equalized".

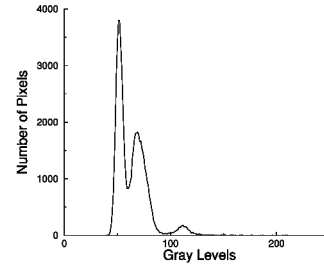
Why do histogram equalization programs usually not produce images with flat histograms?

In the above analysis, we tacitly assumed that variables r and s can take continuous values. In reality, of course, the grey level values are discrete. In the continuous domain there is an infinite number of numbers in any interval $[r, r + dr]$. In digital images we have only a finite number of pixels in each range. As the range is stretched, and the number of pixels in it is preserved, there is only this finite number of pixels with which the stretched range is populated. The histogram that results is spread over the whole range of grey values, but it is far from flat.

Is it possible to enhance an image to have an absolutely flat histogram?

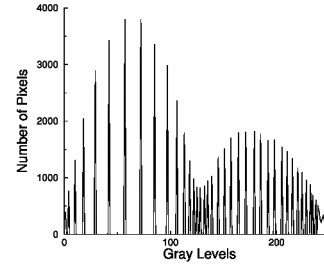
Yes, if we randomly re-distribute the pixels across neighbouring grey values. This method is called histogram equalization with *random additions*. We can better follow it if we consider a very simple example. Let us assume that we have N_1 pixels with value g_1 and N_2 pixels with value g_2. Let us say that we wish to stretch this histogram so that we have $(N_1 + N_2)/3$ pixels with grey value $\tilde{g}_1$, $(N_1 + N_2)/3$ pixels with grey value $\tilde{g}_2$ and $(N_1 + N_2)/3$ pixels with grey value $\tilde{g}_3$. Let us also assume that we have worked out the transformation that leads from g_i to $\tilde{g}_i$. After we apply this transformation, we may find that we have $\tilde{N}_1$ pixels with grey value $\tilde{g}_1$, $\tilde{N}_2$ pixels with grey value $\tilde{g}_2$, $\tilde{N}_3$ pixels with grey value $\tilde{g}_3$, and that $\tilde{N}_1 > (N_1 + N_2)/3$, $\tilde{N}_2 < (N_1 + N_2)/3$ and $\tilde{N}_3 < (N_1 + N_2)/3$. We may pick at random $(N_1 + N_2)/3 - \tilde{N}_3$ pixels with value $\tilde{g}_2$ and give them value $\tilde{g}_3$. Then we may pick at random $\tilde{N}_1 - (N_1 + N_2)/3$

(a) Original image



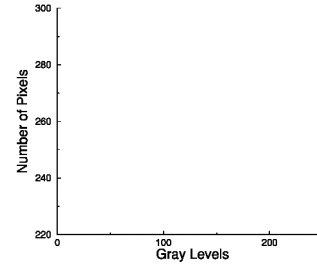
(b) Original histogram

(c) After histogram equalization



(d) Histogram after histogram equalization

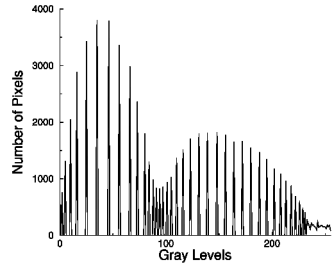
(e) After histogram equalization with random additions



(f) Histogram after histogram equalization with random additions

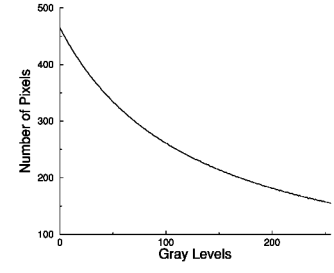
Figure 4.2: Enhancing the image of a bathtub cleaner by histogram equalization

(a) After histogram hyperbolization



(b) Histogram after histogram hyperbolization

(c) After histogram hyperbolization with random additions



(d) Histogram after histogram hyperbolization with random additions

Figure 4.3: Histogram hyperbolization with $\alpha = 0.5$ applied to the image of *Figure* 4.2a

with value $\tilde{g}_1$ and give them value $\tilde{g}_2$. The result will be a perfectly flat histogram. An example of applying this method can be seen in *Figures* 4.2e and 4.3c.

What if we do not wish to have an image with a flat histogram?

We may define $p_s(s)$ to be any function we wish. Then:

$$\int_0^s p_s(y)dy = \int_0^r p_r(x)dx$$

where in order to avoid confusion we have used x and y as the dummy variables of integration. This equation defines s directly in terms of r. Since $p_s(y)$ is known (the desired histogram), one can solve the integral on the left hand side to find a function

f_1 of s. Similarly, the integral on the right hand side can be performed to yield a function f_2 of r; i.e.

$$f_1(s) = f_2(r) \Rightarrow s = f_1^{-1} f_2(r) \tag{4.3}$$

In practice, one may define an intermediate transformation by:

$$w \equiv \int_0^r p_r(x)dx \tag{4.4}$$

This transformation gives the values w of the equalized histogram of the given image. Clearly:

$$\int_0^s p_s(y)dy = w \tag{4.5}$$

This defines another transformation such that $w = T_2(s)$ while we actually need s in terms of w. So this is a three-step process:

1. Equalize the histogram of the given image.
2. Specify the desired histogram and obtain the transformation $w = T_2(s)$.
3. Apply the inverse of the above transformation to the equalized histogram.

An example of applying this method to image 4.2a can be seen in *Figures* 4.3a and 4.3b. This histogram has been produced by setting $p_s(s) = \alpha e^{-\alpha s}$ where α is some positive constant. The effect is to give more emphasis to low grey level values and less to the high ones. This effect is barely visible in *Figure* 4.3b because it is masked by the discretization effect. However, in *Figure* 4.3d it can be seen clearly because the method of random additions was used.

Why should one wish to perform something other than histogram equalization?

One may wish to emphasize certain grey values more than others, in order to compensate for a certain effect; for example, to compensate for the way the human eye responds to the different degrees of brightness. This is a reason for doing histogram hyperbolization: it produces a more pleasing picture.

Example 4.1

The histogram of an image can be approximated by the probability density function

$$p_r(r) = Ae^{-r}$$

where r is the grey-level variable taking values between 0 and b, and A is a normalizing factor. Calculate the transformation $s = T(r)$, where s is the grey level value in the transformed image, such that the transformed image has probability density function

$$p_s(s) = Bse^{-s^2}$$

where s takes values between 0 and b, and B is some normalizing factor.

The transformation $s = T(r)$ can be calculated using equation (4.1).

$$Bse^{-s^2}ds = Ae^{-r}dr$$

We integrate both sides of this equation to obtain the relationship between the distribution functions of variables s and r. To avoid confusion we use as dummy variables of integration y on the left hand side and x on the right hand side:

$$B\int_0^s ye^{-y^2}dy = A\int_0^r e^{-x}dx \tag{4.6}$$

The left hand side of (4.6) is:

$$\int_0^s ye^{-y^2}dy = \frac{1}{2}\int_0^s e^{-y^2}dy^2 = -\frac{1}{2}e^{-y^2}\Big|_0^s = \frac{1-e^{-s^2}}{2} \tag{4.7}$$

The right hand side of (4.6) is:

$$\int_0^s e^{-x}dx = -e^{-x}\Big|_0^r = 1 - e^{-r} \tag{4.8}$$

We substitute from (4.7) and (4.8) into (4.6) to obtain:

$$\frac{1-e^{-s^2}}{2} = \frac{A}{B}(1-e^{-r}) \Rightarrow$$
$$e^{-s^2} = 1 - \frac{2A}{B}(1-e^{-r}) \Rightarrow$$
$$-s^2 = ln\left[1 - \frac{2A}{B}(1-e^{-r})\right] \Rightarrow$$
$$s = \sqrt{-ln\left[1 - \frac{2A}{B}(1-e^{-r})\right]}$$

What if the image has inhomogeneous contrast?

The approach described above is global, i.e. we modify the histogram which refers to the whole image. However, the image may have variable quality at various parts. For example, it may have a wide shadow band right in the middle, with its top and bottom parts being adequately visible. In that case we can apply the above techniques locally: We scan the image with a window inside which we modify the histogram but

we alter only the value of the grey level of the central pixel. Clearly such a method is costly and various algorithms have been devised to make it more efficient.

Figure 4.4a shows a classical example of an image that requires local enhancement. The picture was taken indoors looking towards windows with plenty of ambient light coming through. All outdoor sections are fine, but in the indoor part the film was under-exposed. The result of global histogram equalization shown in *Figure* 4.4b is not bad, but it makes the outdoor parts over-exposed in order to allow us to see the details of the interior. The result of the local histogram equalization on the other hand, shown in *Figure* 4.4c, is overall a much more balanced picture. The window size used for this was 40×40, with the original image being of size 400×400. Notice that no part of the picture gives the impression of being over-exposed or under-exposed. There are parts of the image, however, that look damaged: at the bottom of the picture and a little at the top. They correspond to parts of the original film which received too little light to record anything. They correspond to flat black patches, and by trying to enhance them we simply enhance the film grain or the instrument noise. This effect is more prominent in the picture of the hanging train of Wupertal shown in *Figure* 4.5. Local histogram equalization (the result of which is shown in *Figure* 4.5c) attempts to improve parts of the picture that are totally black, in effect trying to amplify non-existing information. However, those parts of the image with some information content are enhanced in a pleasing way.

A totally different effect becomes evident in *Figure* 4.6c which shows the local histogram enhancement of a picture taken at Karlstejn castle in the Czech Republic, shown in *Figure* 4.6a. The castle at the back consists of flat grey walls. The process of local histogram equalization amplifies every small variation of the wall to such a degree that the wall looks like the rough surface of a rock. Further, on the left of the picture we observe again the effect of trying to enhance a totally black area. In this case, the result of global histogram equalization looks much more acceptable, in spite of the fact that if we were to judge from the original image, we would have thought that local histogram equalization would produce a better result.

Is there an alternative to histogram manipulation?

Yes, one may use the mean and standard deviation of the distribution of pixels inside a window. Let us say that the mean grey value inside a window centred at (x, y) is $m(x, y)$, the variance of the pixels inside the window is $\sigma(x, y)$, and the value of pixel (x, y) is $f(x, y)$. We can enhance the variance inside each such window by using a transformation of the form:

$$g(x, y) = A[f(x, y) - m(x, y)] + m(x, y) \tag{4.9}$$

where A is some scalar.

We would like areas which have low variance to have their variance amplified most. So we choose the amplification factor A inversely proportional to $\sigma(x, y)$:

$$A = \frac{kM}{\sigma(x, y)}$$

where k is a constant, and M is the average grey value of the image.

Figure 4.4d shows the results of applying this process to image 4.4a with $k = 3$ and window size 5×5. Note that although details in the image have become explicit, the picture overall is too dark and not particularly pleasing. *Figures* 4.5d and 4.6d show the results of applying the same process to the images 4.5a and 4.6a respectively, with the additional post-processing of histogram equalization.

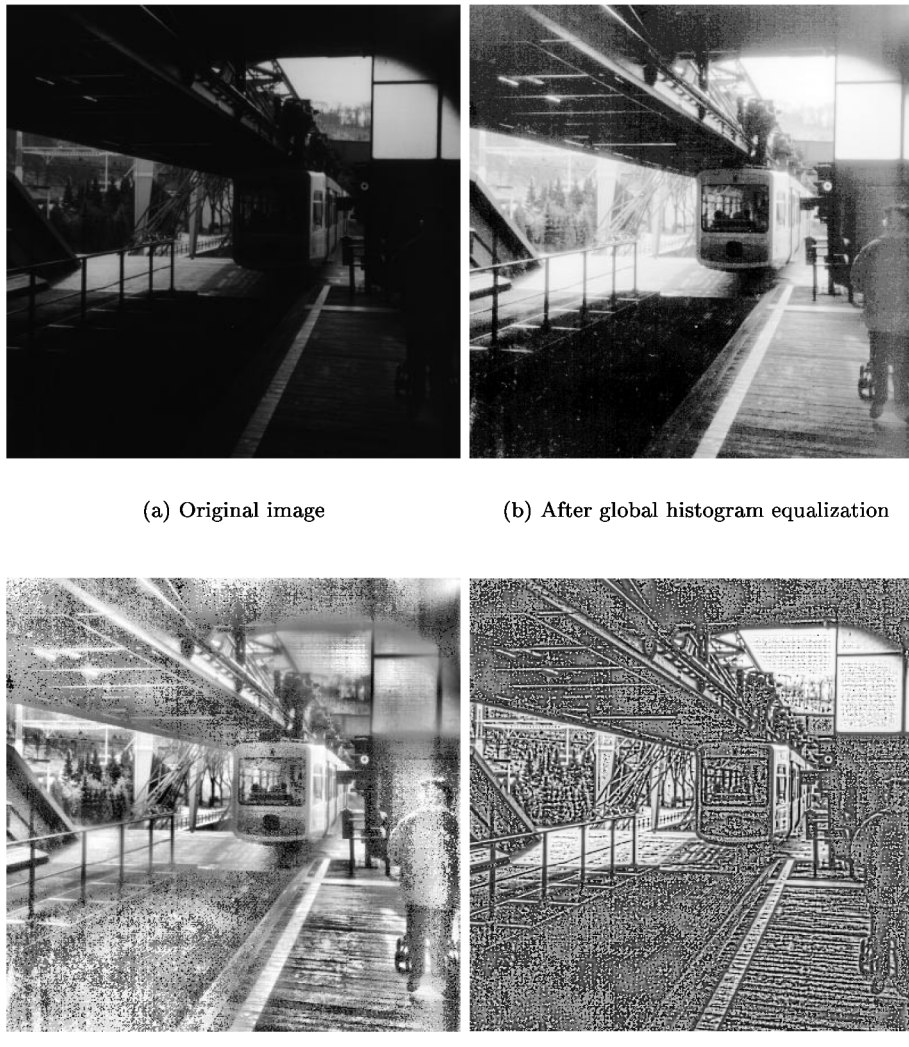
(a) Original image

(b) After global histogram equalization

(c) After local histogram equalization

(d) After local enhancement

Figure 4.4: Enhancing the image of a young train driver.



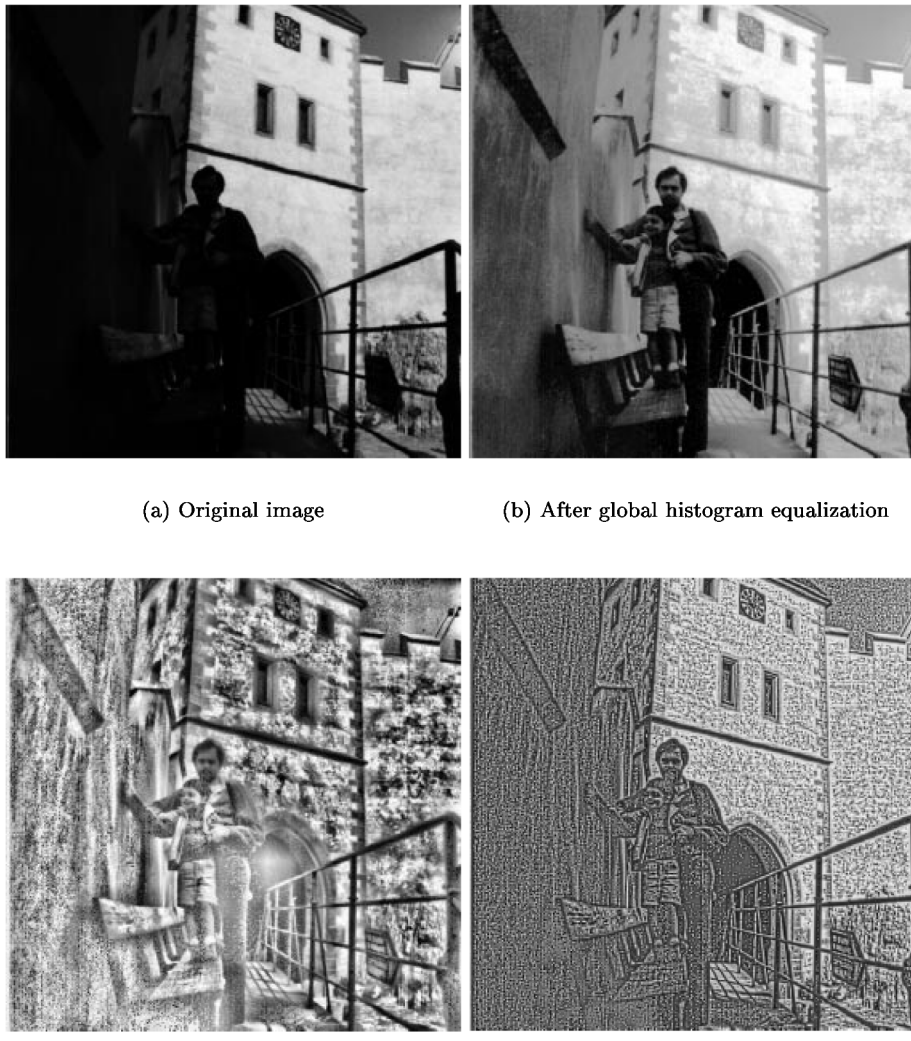
(a) Original image

(b) After global histogram equalization

(c) After local histogram equalization

(d) After local enhancement

Figure 4.5: Enhancing the image of the hanging train of Wupertal.



(a) Original image

(b) After global histogram equalization

(c) After local histogram equalization

(d) After local enhancement

Figure 4.6: Enhancing the image at the Karlstejn castle.

How can we improve the contrast of a multispectral image?

A *multispectral* or *multiband* or colour image consists of several arrays of the same scene, one for each spectral component. Each of these bands is a grey level image giving the intensity of light at the particular spectral component at the position of each

pixel. Suppose for simplicity that we have three spectral bands, Red, Green and Blue. Then each picture consists of three bands, three grey level images. Alternatively, we may say that each pixel carries three values, one for each spectral band. We can plot these triplets in a 3D coordinate space, called RGB because we measure the grey value of a pixel in each of the three bands along the three axes. The pixels of the colour image plotted in this space form a cluster.

If we were to use only one of these bands, we would like to choose the one that shows the most detail; i.e. the one with the maximum contrast, the one in which the values of the pixels are most spread.

It is possible that the maximum spread of the values of the pixels is not along any of the axes, but along another line (see *Figure* 4.7a). To identify this line we must perform *principal component analysis* or take the *Karhunen–Loeve transformation* of the image.

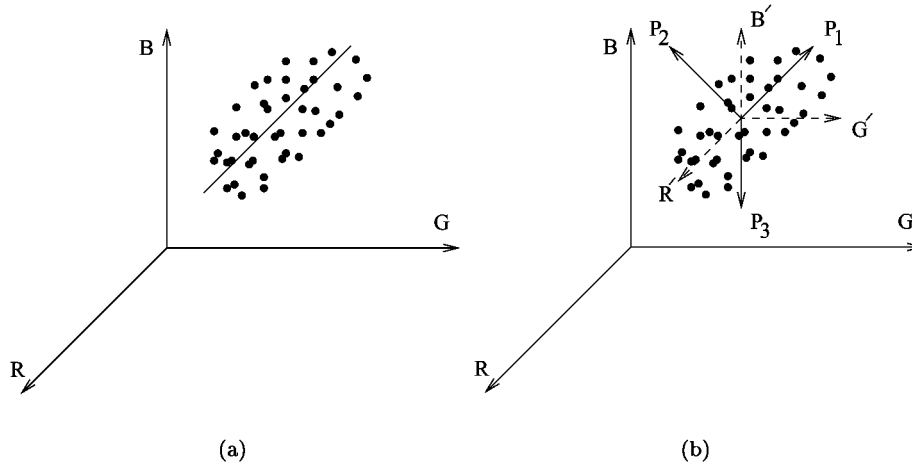


Figure 4.7: The pixels of a colour image form a cluster in the colour space. The maximum spread of this cluster may be along a line not parallel with any of the colour axes.

What is principal component analysis?

Principal component analysis (or Karhunen–Loeve transformation) identifies a linear transformation of the coordinate system such that the three axes of the new coordinate system coincide with the directions of the three largest spreads of the point distribution. In this new set of axes the data are uncorrelated. This means that if we form a grey image by using the values of the first co-ordinate of each pixel, it will contain totally uncorrelated information from the information that will be contained in the grey image formed by the second coordinate of each pixel and the information contained in the image formed by the third coordinate of each pixel.

What is the relationship of the Karhunen–Loeve transformation discussed here and the one discussed in Chapter 3?

They both analyse an ensemble of random outcomes into their uncorrelated components. However, in Chapter 3 the whole image was considered as the outcome of a random experiment, with the other random outcomes in the ensemble not available. Their lack of availability was compensated by the assumed ergodicity. So, although the ensemble statistics were computed over the single available image using spatial statistics, they were assumed to be averages computed over all random outcomes, i.e. all versions of the image. Here the values of a **single** pixel are considered to be the outcomes of a random experiment and we have at our disposal the whole ensemble of random outcomes made up from all the image pixels.

How can we perform principal component analysis?

To perform principal component analysis we must diagonalize the covariance matrix of our data. The autocovariance function of the outputs of the assumed random experiment is:

$$C(i,j) \equiv E\{(x_i(k,l) - x_{i0})(x_j(k,l) - x_{j0})\}$$

where $x_i(k,l)$ is the value of pixel (k,l) at band i, x_{i0} is the mean of band i, $x_j(k,l)$ is the value of the same pixel in band j, x_{j0} is the mean of band j, and the expectation value is over all outcomes of the random experiment, i.e. over all pixels of the image:

$$C(i,j) = \frac{1}{N^2}\sum_{k=1}^{N}\sum_{l=1}^{N}(x_i(k,l) - x_{i0})(x_j(k,l) - x_{j0}) \tag{4.10}$$

Since we have three bands, variables i and j take only three values to indicate R,G and B and the covariance matrix is a 3×3 matrix. For data that are uncorrelated, C is diagonal; i.e. $C(i,j) = 0$ for $i \neq j$. To achieve this we must transform our data using the transformation matrix A made up from the eigenvectors of the covariance matrix of the untransformed data. The process is as follows:

1. Find the mean of the distribution of points in the colour space, say point (R_0, G_0, B_0).
2. Subtract the mean grey level value from each corresponding band. This is equivalent to translating the RGB coordinate system to be centred at the centre of the pixel distribution (see axes $R'G'B'$ in *Figure* 4.7b).
3. Find the autocorrelation matrix $C(i,j)$ of the initial distribution (where i and j take the values R, G and B).
4. Find the eigenvalues of $C(i,j)$ and arrange them in **decreasing** order. Form the eigenvector matrix A, having the eigenvectors as rows.

5. Transform the distribution using matrix A. Each triplet $\mathbf{x} = \begin{pmatrix} R \\ G \\ B \end{pmatrix}$ is transformed into $\mathbf{y} = \begin{pmatrix} P_1 \\ P_2 \\ P_3 \end{pmatrix}$ by: $\mathbf{y} = A\mathbf{x}$; i.e. $y_k = \sum_i a_{ki} x_i$.

This is a linear transformation. The new "colours" are linear combinations of the intensity values of the initial colours, arranged so that the first principal component contains most of the information for the image (see *Figure* 4.7b).

What are the advantages of using principal components to express an image?

The advantages of using principal components are:

1. The information conveyed by each band is maximal for the number of bits used because the bands are uncorrelated and no information contained in one band can be predicted by the knowledge of the other bands.
2. If we want to use a monochrome version of the image, we can restrict ourselves to the first principal component only and be sure that it has the maximum contrast and contains the maximum possible information conveyed by a single band of the image.

An example of principal component analysis is shown in *Figure* 4.8. Although at first glance not much difference is observed between *Figures* 4.8a, 4.8b, 4.8c and 4.8d, at a more careful examination, we can see that the first principal component combines the best parts of all three bands: For example, the face of the boy has more contrast in 4.8b and 4.8c than in 4.8a, while his right leg has more contrast with his trousers in 4.8a and 4.8b than in 4.8c. In 4.8d we have good contrast in both these places. Similarly, the contrast between the trousers and the ground is non-existent in 4.8a and 4.8b but it is obvious in 4.8c. Image 4.8d shows it as well.

What are the disadvantages of principal component analysis?

The grey values in the bands created from principal component analysis have no physical meaning, as they do not correspond to any physical colours. As a result, the grey value of a pixel cannot be used for the classification of a pixel. This is particularly relevant to remote sensing applications, where often pixels are classified according to their grey values. In a **principal component band**, pixels that represent water, for example, may appear darker or brighter than other pixels in the image depending on the image content, while the degree of greyness of water pixels in the various **spectral bands** is always consistent, well understood by remote sensing scientists, and often used to identify them.

(a) Red component (b) Green component (c) Blue component

(d) First principal component (e) Second principal component (f) Third principal component

Figure 4.8: Example of principal component analysis of a colour image.

Example 4.2

Is it possible for matrix C below to represent the autocovariance matrix of a three-band image?

$$C = \begin{pmatrix} -1 & 0 & 1 \\ 0 & 1 & -2 \\ -2 & 2 & 0 \end{pmatrix}$$

This matrix cannot represent the autocovariance matrix of an image because from equation (4.10) it is obvious that C must be symmetric with positive elements along its diagonal.

Example 4.3

A three-band image consists of a green, blue and red band with mean $3, 2$ an 3 respectively. The autocovariance matrix of this image is given by:

$$B = \begin{pmatrix} 2 & 0 & 1 \\ 0 & 2 & 0 \\ 1 & 0 & 2 \end{pmatrix}$$

A pixel has intensity values $5, 3$ and 4 in the three bands respectively. What will be the transformed values of the same pixel in the three principal component bands?

First we must find the eigenvalues of matrix B:

$$\begin{vmatrix} 2-\lambda & 0 & 1 \\ 0 & 2-\lambda & 0 \\ 1 & 0 & 2-\lambda \end{vmatrix} = 0$$

$$\Rightarrow (2-\lambda)^3 - (2-\lambda) = 0 \Rightarrow (2-\lambda)\left[(2-\lambda)^2 - 1\right] = 0$$

$$\Rightarrow (2-\lambda)(1-\lambda)(3-\lambda) = 0$$

Therefore, $\lambda_1 = 3$, $\lambda_2 = 2$, $\lambda_3 = 1$. The corresponding eigenvectors are:

$$\begin{pmatrix} 2 & 0 & 1 \\ 0 & 2 & 0 \\ 1 & 0 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = 3 \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \Rightarrow \left| \begin{array}{l} 2x_1 + x_3 = 3x_1 \\ 2x_2 = 3x_2 \\ x_1 + 2x_3 = 3x_3 \end{array} \right| \Rightarrow \left| \begin{array}{l} x_1 = x_3 \\ x_2 = 0 \end{array} \right| \Rightarrow$$

$$\Rightarrow \mathbf{u_1} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \end{pmatrix}$$

$$\begin{pmatrix} 2 & 0 & 1 \\ 0 & 2 & 0 \\ 1 & 0 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = 2 \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \Rightarrow \left| \begin{array}{l} 2x_1 + x_3 = 2x_1 \\ 2x_2 = 2x_2 \\ x_1 + 2x_3 = 2x_3 \end{array} \right| \Rightarrow \left| \begin{array}{l} x_2 \text{ anything} \\ x_1 = x_3 = 0 \end{array} \right| \Rightarrow$$

$$\Rightarrow \mathbf{u_2} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} 2 & 0 & 1 \\ 0 & 2 & 0 \\ 1 & 0 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \Rightarrow \left| \begin{array}{l} 2x_1 + x_3 = x_1 \\ 2x_2 = x_2 \\ x_1 + 2x_3 = x_3 \end{array} \right| \Rightarrow \left| \begin{array}{l} x_1 = -x_3 \\ x_2 = 0 \end{array} \right| \Rightarrow$$

$$\Rightarrow \mathbf{u_3} = \begin{pmatrix} -\frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \end{pmatrix}$$

The transformation matrix A is:

$$\begin{pmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ 0 & 1 & 0 \\ -\frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \end{pmatrix}$$

Pixel $\begin{pmatrix} 5 \\ 3 \\ 4 \end{pmatrix}$ *will transform to:*

$$\begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ 0 & 1 & 0 \\ -\frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 5 \\ 3 \\ 4 \end{pmatrix} = \begin{pmatrix} \frac{9}{\sqrt{2}} \\ 3 \\ -\frac{1}{\sqrt{2}} \end{pmatrix}$$

Example 4.4

A 4×4 three-band image is given:

$$R = \begin{pmatrix} 3 & 3 & 5 & 6 \\ 3 & 4 & 4 & 5 \\ 4 & 5 & 5 & 6 \\ 4 & 5 & 5 & 6 \end{pmatrix} \quad G = \begin{pmatrix} 3 & 2 & 3 & 4 \\ 1 & 5 & 3 & 6 \\ 4 & 5 & 3 & 6 \\ 2 & 4 & 4 & 5 \end{pmatrix} \quad B = \begin{pmatrix} 4 & 2 & 3 & 4 \\ 1 & 4 & 2 & 4 \\ 4 & 3 & 3 & 5 \\ 2 & 3 & 5 & 5 \end{pmatrix}$$

Calculate its three principal components and verify that they are uncorrelated.

First we calculate the mean of each band:

$$\begin{aligned} R_0 &= \frac{1}{16}(3+3+5+6+3+4+4+5+4+5+5+6+4+5+5+6) \\ &= \frac{73}{16} = 4.5625 \\ G_0 &= \frac{1}{16}(3+2+3+4+1+5+3+6+4+5+3+6+2+4+4+5) \\ &= \frac{60}{16} = 3.75 \\ B_0 &= \frac{1}{16}(4+2+3+4+1+4+2+4+4+3+3+5+2+3+5+5) \\ &= \frac{50}{16} = 3.375 \end{aligned}$$

Next we calculate the elements of the covariance matrix as:

$$C_{RR} = \frac{1}{16}\sum_{k=1}^{4}\sum_{l=1}^{4}(R(k,l)-R_0)^2 = 0.996094$$

$$C_{RG} = \frac{1}{16}\sum_{k=1}^{4}\sum_{l=1}^{4}(R(k,l)-R_0)(G(k,l)-G_0) = 0.953125$$

$$C_{RB} = \frac{1}{16}\sum_{k=1}^{4}\sum_{l=1}^{4}(R(k,l)-R_0)(B(k,l)-B_0) = 0.726563$$

$$C_{GG} = \frac{1}{16}\sum_{k=1}^{4}\sum_{l=1}^{4}(G(k,l)-G_0)^2 = 1.9375$$

$$C_{GB} = \frac{1}{16}\sum_{k=1}^{4}\sum_{l=1}^{4}(G(k,l)-G_0)(B(k,l)-B_0) = 1.28125$$

$$C_{BB} = \frac{1}{16}\sum_{k=1}^{4}\sum_{l=1}^{4}(B(k,l)-B_0)^2 = 1.359375$$

Therefore, the covariance matrix is:

$$C = \begin{pmatrix} 0.996094 & 0.953125 & 0.726563 \\ 0.953125 & 1.937500 & 1.281250 \\ 0.726563 & 1.28125 & 1.359375 \end{pmatrix}$$

The eigenvalues of this matrix are:

$$\lambda_1 = 3.528765 \quad \lambda_2 = 0.435504 \quad \lambda_3 = 0.328700$$

The corresponding eigenvectors are:

$$\mathbf{u_1} = \begin{pmatrix} 0.427670 \\ 0.708330 \\ 0.561576 \end{pmatrix} \quad \mathbf{u_2} = \begin{pmatrix} 0.876742 \\ -0.173808 \\ -0.448457 \end{pmatrix} \quad \mathbf{u_3} = \begin{pmatrix} 0.220050 \\ -0.684149 \\ 0.695355 \end{pmatrix}$$

The transformation matrix therefore is:

$$A = \begin{pmatrix} 0.427670 & 0.708330 & 0.561576 \\ 0.876742 & -0.173808 & -0.448457 \\ 0.220050 & -0.684149 & 0.695355 \end{pmatrix}$$

We can find the principal components by using this matrix to transform the values of every pixel. For example, for the first few pixels we find:

$$\begin{pmatrix} 5.654302 \\ 0.314974 \\ 1.389125 \end{pmatrix} = \begin{pmatrix} 0.427670 & 0.708330 & 0.561576 \\ 0.876742 & -0.173808 & -0.448457 \\ 0.220050 & -0.684149 & 0.695355 \end{pmatrix} \begin{pmatrix} 3 \\ 3 \\ 4 \end{pmatrix}$$

$$\begin{pmatrix} 3.822820 \\ 1.385694 \\ 0.682562 \end{pmatrix} = \begin{pmatrix} 0.427670 & 0.708330 & 0.561576 \\ 0.876742 & -0.173808 & -0.448457 \\ 0.220050 & -0.684149 & 0.695355 \end{pmatrix} \begin{pmatrix} 3 \\ 2 \\ 2 \end{pmatrix}$$

$$\begin{pmatrix} 5.948065 \\ 2.516912 \\ 1.133874 \end{pmatrix} = \begin{pmatrix} 0.427670 & 0.708330 & 0.561576 \\ 0.876742 & -0.173808 & -0.448457 \\ 0.220050 & -0.684149 & 0.695355 \end{pmatrix} \begin{pmatrix} 5 \\ 3 \\ 3 \end{pmatrix}$$

We use the first element of each transformed triplet to form the first principal component of the image, the second element for the second principal component, and the third for the third one. In this way we derive:

$$P_1 = \begin{pmatrix} 5.654302 & 3.822820 & 5.948065 & 7.645640 \\ 2.552915 & 7.498631 & 4.958820 & 8.634631 \\ 6.790301 & 7.364725 & 5.948065 & 9.623876 \\ 4.250490 & 6.656395 & 7.779546 & 8.915546 \end{pmatrix}$$

$$P_2 = \begin{pmatrix} 0.314974 & 1.385694 & 2.516912 & 2.771389 \\ 2.007960 & 0.844099 & 2.088630 & 1.547035 \\ 1.017905 & 2.169300 & 2.516912 & 1.975317 \\ 2.262436 & 2.343106 & 1.446188 & 2.149123 \end{pmatrix}$$

$$P_3 = \begin{pmatrix} 1.389125 & 0.682562 & 1.133874 & 1.365131 \\ 0.671360 & 0.240878 & 0.218468 & -0.223219 \\ 0.925027 & -0.234424 & 1.133874 & 0.692187 \\ 0.902617 & 0.449725 & 1.840433 & 1.376336 \end{pmatrix}$$

To confirm that these new bands contain uncorrelated data we shall calculate their autocovariance matrix. First we find the mean of each band: P_{10}, P_{20}, P_{30}. *Then we compute:*

$$C_{P_1P_1} = \frac{1}{16}\sum_{i=1}^{4}\sum_{j=1}^{4}(P_1(i,j) - P_{10})^2 = 3.528765$$

$$C_{P_1P_2} = \frac{1}{16}\sum_{i=1}^{4}\sum_{j=1}^{4}(P_1(i,j) - P_{10})(P_2(i,j) - P_{20}) = 0.0$$

$$C_{P_1P_3} = \frac{1}{16}\sum_{i=1}^{4}\sum_{j=1}^{4}(P_1(i,j) - P_{10})(P_3(i,j) - P_{30}) = 0.0$$

$$C_{P_2P_2} = \frac{1}{16}\sum_{i=1}^{4}\sum_{j=1}^{4}(P_2(i,j) - P_{20})^2 = 0.435504$$

$$C_{P_2P_3} = \frac{1}{16}\sum_{i=1}^{4}\sum_{j=1}^{4}(P_2(i,j) - P_{20})(P_3(i,j) - P_{30}) = 0.0$$

$$C_{P_3P_3} = \frac{1}{16}\sum_{i=1}^{4}\sum_{j=1}^{4}(P_3(i,j) - P_{30})^2 = 0.328700$$

We see that this covariance matrix is diagonal, so it refers to uncorrelated data.

Example 4.5

For the image in Example 4.4 show that the first principal component has more contrast than any of the original bands.

The contrast of an image can be characterized by the range of grey values it has. We can see that the contrast of the original image was 3 *in the red band,* 5 *in the green band and* 4 *in the blue band. The range of values in the first principal component is* $9.623876 - 2.552915 = 7.070961$. *This is larger than any of the previous ranges.*

Some of the images with enhanced contrast appear very noisy. Can we do anything about that?

Indeed, this is the case for images 4.4c, 4.5c and 4.6c, where there are large uniformly coloured regions, which happen to cover entirely the window inside which local histogram equalization takes place. Then the grey values of these pixels are stretched to the full range of $0-255$, and the noise is significantly enhanced. We have to use then some noise reduction techniques to post-process the image. The technique we use depends on the type of noise that is present in the image.

What are the types of noise present in an image?

There are various types of noise. However, they fall into two major classes: *additive* and *multiplicative* noise. An example of multiplicative noise is variable illumination. This is perhaps the most common type of noise in images. Additive noise is often assumed to be *impulse noise* or *Gaussian noise. Figure* 4.9a shows an image corrupted with impulse noise and *Figure* 4.9b shows an image corrupted with additive zero-mean Gaussian noise.

Impulse noise alters at random the value of some pixels. In a binary image this means that some black pixels become white and some white pixels become black. This is why this noise is also called *salt and pepper noise.* Additive zero-mean Gaussian noise means that a value drawn from a zero-mean Gaussian probability density function is added to the true value of every pixel.

(a) Image with impulse noise

(b) Image with additive Gaussian noise

(c) Median filtering of (a)

(d) Median filtering of (b)

(e) Smoothing of (a) by averaging

(f) Smoothing of (b) by averaging

Figure 4.9: Examples of filtering to remove noise.

We use *rank order filtering* to remove impulse noise and *smoothing* to reduce Gaussian noise.

What is a rank order filter?

A rank order filter is a filter the output value of which depends on the ranking of the pixels according to their grey values inside the filter window. The most common rank order filter is the *median filter*.

Figure 4.10 shows the result of trying to remove the noise from output images 4.4c and 4.5c by median filtering.

(a) (b)

Figure 4.10: Improving images 4.4c and 4.5c by using median filtering (median filter size $= 3 \times 3$).

What is median filtering?

The median is the grey level value which divides a distribution in two equally numbered populations. For example, if we use a 5×5 window, we have 25 grey level values which we order in an increasing sequence. Then the median is the thirteenth value. This has the effect of forcing points with distinct intensities to be more like their neighbours, thus eliminating intensity spikes which appear isolated.

Figure 4.9c shows image 4.9a processed with a median filter and with a window of size 5×5, while *Figure* 4.9d shows image 4.9b (which contains Gaussian noise) having been processed in the same way. It is clear that the median filter removes the impulse noise almost completely.

What if the noise in an image is not impulse?

The most common type of noise in images is Gaussian. We can remove Gaussian noise by *smoothing* the image. For example, we may replace the value of each pixel by the average value inside a small window around the pixel. *Figures* 4.9e and 4.9f show the result of applying this process to images 4.9a and 4.9b respectively. The size of the window used is the same as for the median filtering of the same images, i.e. 5×5. We note that this type of filtering is much more effective for the Gaussian noise, but produces bad results in the case of impulse noise. This is a simple form of *lowpass filtering* of the image.

Why does lowpass filtering reduce noise?

Usually, the noise which is superimposed on the image is uncorrelated. This means that it has a flat spectrum. On the other hand, most images have spectra which have higher values in the low frequencies and gradually reducing values for high frequencies. After a certain frequency, the spectrum of a noisy signal is dominated by the noise component (see *Figure* 4.11). So, if we use a lowpass filter, we kill off all the noise-dominated high-frequency components. At the same time, of course, we kill also the useful information of the image buried in these high frequencies. The result is a clean, but blurred image. The process is as follows:

1. Find the Fourier transform of the image.
2. Multiply it with a function which does not alter frequencies below a certain cutoff frequency but which kills off all higher frequencies. In the 2-dimensional frequency space, this ideal lowpass filter is schematically depicted in *Figure* 4.12.
3. Take the inverse Fourier transform of this product.

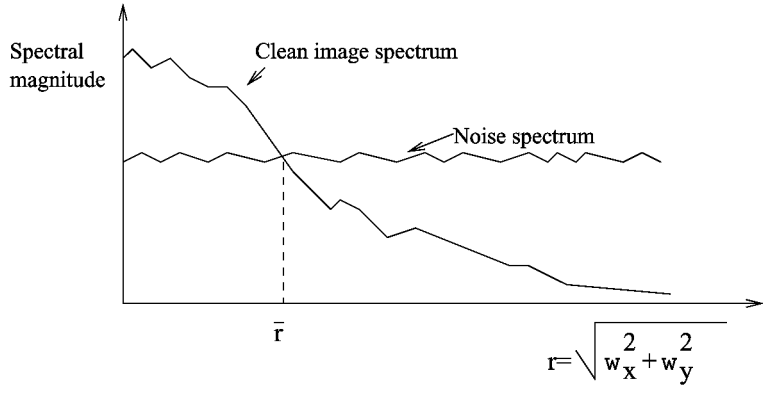


Figure 4.11: After a certain frequency, the spectrum of the noisy image is dominated by the noise. Ideally the cutoff frequency of the lowpass filter should be at $r_0 = \bar{r}$.

Multiplication of the two frequency spectra is equivalent to convolution of the actual functions. So, what we can do instead of the above procedure, is to find the 2-dimensional function in real space which has as its Fourier transform the ideal lowpass filter, and convolve our image with that function. This would be ideal, but it does not work in practice because the function the Fourier transform of which is the ideal lowpass filter is infinite in extent.

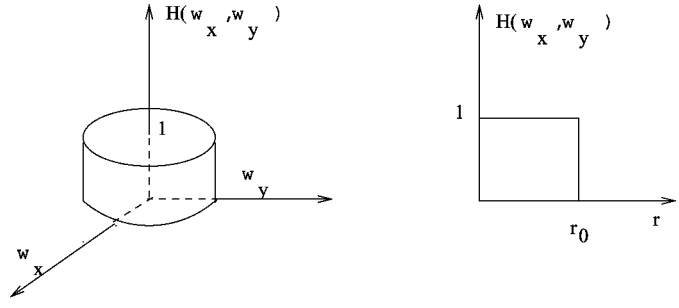


Figure 4.12: The ideal lowpass filter in 2D in the frequency domain. On the right a cross-section of this filter with cutoff frequency r_0 ($r \equiv \sqrt{\omega_x^2 + \omega_y^2}$).

What if we are interested in the high frequencies of an image?

It is possible that we may want to enhance the small details of a picture instead of ironing them out. Such a process is called *sharpening* and it enhances small fluctuations in the intensity of the image, noise included.

One way to achieve this is to calculate at each pixel the local gradient of the intensity using numerical difference formulae. If one wants to be more sophisticated, one can use the filtering approach we discussed in the context of smoothing. Only now, of course, the filter should be highpass and allow the high-frequency components to survive while killing the low-frequency components.

What is the ideal highpass filter?

The ideal highpass filter in the frequency domain is schematically depicted in *Figure* 4.13.

Filtering with such a filter in the frequency domain is equivalent to convolving in real space with the function that has this filter as its Fourier transform. There is no finite function which corresponds to the highpass filter and one has to resort to various approximations.

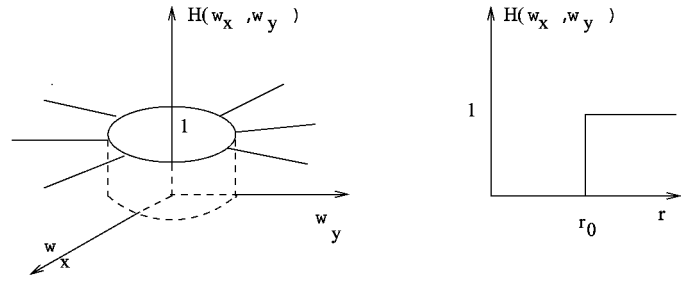


Figure 4.13: The spectrum of the ideal highpass filter is 1 everywhere, except inside a circle of radius r_0 in the frequency domain, where it is 0. On the right, a cross-section of such a filter. Here $r \equiv \sqrt{\omega_x^2 + \omega_y^2}$.

How can we improve an image which suffers from variable illumination?

This is a problem which can be dealt with if we realize that every image function $f(x, y)$ is the product of two factors: an illumination function $i(x, y)$ and a reflectance function $r(x, y)$ that is intrinsic to the imaged surface:

$$f(x, y) = i(x, y)r(x, y) \tag{4.11}$$

Illumination is generally of uniform nature and yields low-frequency components in the Fourier transform of the image. Different materials (objects) on the other hand, imaged next to each other, cause sharp changes of the reflectance function, which causes sharp transitions in the intensity of an image. These sharp changes are associated with high-frequency components. We can try to separate these two factors by first taking the logarithm of equation (4.11) so that the two effects are additive rather than multiplicative: $\ln f(x, y) = \ln i(x, y) + \ln r(x, y)$

Then we filter this logarithmic image by what is called a *homomorphic filter*. Such a filter will enhance the high frequencies and suppress the low frequencies so that the variation in the illumination will be reduced while edges (and details) will be sharpened. The cross-section of a homomorphic filter looks like the one shown in *Figure* 4.14.

Figures 4.15a and 4.16a show two images with smoothly varying illumination from left to right. The results after homomorphic filtering shown in *Figures* 4.15b and 4.16b constitute a clear improvement with the effect of variable illumination greatly reduced and several details particularly in the darker parts of the images made visible. These results were obtained by applying to the logarithm of the original image, a filter with the following transfer function:

$$H(\omega_x, \omega_y) = \frac{1}{1 + e^{-s(\sqrt{\omega_x^2 + \omega_y^2} - \omega_0)}} + A$$

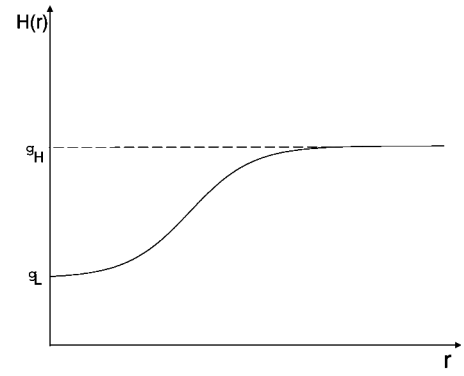


Figure 4.14: A cross-section of a homomorphic filter as a function of polar frequency, $r \equiv \sqrt{\omega_x^2 + \omega_y^2}$.

(a) Original image (b) After homomorphic filtering

Figure 4.15: Example of homomorphic filtering Haroula at the keyboard.

with $s = 1$, $\omega_0 = 128$ and $A = 10$. The parameters of this filter are related as follows to the parameters γ_H and γ_L of *Figure* 4.14:

$$\gamma_L = \frac{1}{1 + e^{s\omega_0}} + A, \qquad \gamma_H = 1 + A$$

(a) Original image (b) After homomorphic filtering

Figure 4.16: Homomorphic filtering Mrs Petrou at home.

Can any of the objectives of image enhancement be achieved by the linear methods we learned in Chapter 2?

Yes, often image smoothing and image sharpening is done by convolving the image with a suitable filter. This is the reason we prefer the filters to be finite in extent: Finite convolution filters can be implemented as linear operators applied on the image.

Example 4.6

You have a 3×3 image which can be represented by a 9×1 vector. Derive a matrix which, when it operates on this image, smoothes its columns by averaging every three successive pixels, giving them weights $\frac{1}{4}, \frac{1}{2}, \frac{1}{4}$. To deal with the border pixels assume that the image is repeated periodically in all directions.

Let us say that the original image is

$$\begin{pmatrix} g_{11} & g_{12} & g_{13} \\ g_{21} & g_{22} & g_{23} \\ g_{31} & g_{32} & g_{33} \end{pmatrix}$$

and its smoothed version is

$$\begin{pmatrix} \tilde{g}_{11} & \tilde{g}_{12} & \tilde{g}_{13} \\ \tilde{g}_{21} & \tilde{g}_{22} & \tilde{g}_{23} \\ \tilde{g}_{31} & \tilde{g}_{32} & \tilde{g}_{33} \end{pmatrix}$$

Let as also say that the smoothing matrix we wish to identify is A, with elements a_{ij}:

$$\begin{pmatrix} \tilde{g}_{11} \\ \tilde{g}_{21} \\ \tilde{g}_{31} \\ \tilde{g}_{12} \\ \tilde{g}_{22} \\ \tilde{g}_{32} \\ \tilde{g}_{13} \\ \tilde{g}_{23} \\ \tilde{g}_{33} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{19} \\ a_{21} & a_{22} & \dots & a_{29} \\ a_{31} & a_{32} & \dots & a_{39} \\ a_{41} & a_{42} & \dots & a_{49} \\ a_{51} & a_{52} & \dots & a_{59} \\ a_{61} & a_{62} & \dots & a_{69} \\ a_{71} & a_{72} & \dots & a_{79} \\ a_{81} & a_{82} & \dots & a_{89} \\ a_{91} & a_{92} & \dots & a_{99} \end{pmatrix} \begin{pmatrix} g_{11} \\ g_{21} \\ g_{31} \\ g_{12} \\ g_{22} \\ g_{32} \\ g_{13} \\ g_{23} \\ g_{33} \end{pmatrix}$$

From the above equation we have:

$$\begin{aligned} \tilde{g}_{11} = {} & a_{11}g_{11} + a_{12}g_{21} + a_{13}g_{31} + a_{14}g_{12} + a_{15}g_{22} \\ & + a_{16}g_{32} + a_{17}g_{13} + a_{18}g_{23} + a_{19}g_{33} \end{aligned} \tag{4.12}$$

From the definition of the smoothing mask, we have:

$$\tilde{g}_{11} = \frac{1}{4}g_{31} + \frac{1}{2}g_{11} + \frac{1}{4}g_{21} \tag{4.13}$$

Comparison of equations (4.12) and (4.13) shows that we must set:

$$a_{11} = \frac{1}{2}, a_{12} = \frac{1}{4}, a_{13} = \frac{1}{4}, a_{14} = a_{15} = \ldots = a_{19} = 0$$

Working in a similar way for a few more elements, we can see that the matrix we wish to identify has the form:

$$A = \begin{pmatrix} \frac{1}{2} & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{4} & \frac{1}{4} \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{2} \end{pmatrix}$$

What is the "take home" message of this chapter?

With image enhancement we try to make images look better according to subjective criteria. Most of the methods used are non-linear, and therefore they cannot be described within the framework developed in the previous chapters. Some of the methods used are not only non-linear, but also inhomogeneous.

Contrast enhancement of a grey image can be achieved by manipulating the grey values of the pixels so that they become more diverse. This can be done by defining a transformation that converts the distribution of the grey values to a pre-specified shape. The choice of this shape may be totally arbitrary. If, however, we are dealing with a multiband image, we can define a line in the multidimensional space (where we measure the grey values of a pixel in the different bands along the corresponding axes) such that when we project the data on this line, their projected values are maximally spread (see *Figure* 4.7a). This line defines the first principal axis of the data ($\equiv$ pixel values) in the multidimensional space. We can create a grey image by assigning to each pixel its projected value along the first principal axis. This is called the first principal component of the multiband image. From all grey bands we can construct from the multiband image, the first principal component has the maximum contrast and contains the most information. However, the grey values of the pixels do not have physical meaning as they are linear combinations of the grey values of the original bands. The basic difference between histogram manipulation and principal component analysis is that in the former the values the pixels carry are changed arbitrarily, while in the latter the joint probability density function of the pixel values remains the same; we simply "read" the data in a more convenient way, without introducing information that is not already there (compare, for example, the enhanced images 4.6c and 4.8d).

We can also enhance an image in a desirable way by manipulating its Fourier spectrum: We can preferentially kill frequency bands we do not want. This can be achieved with the help of convolution filters that have pre-specified spectra. It is necessary, therefore, to understand how we can develop and use filters appropriate for each task. This is the topic of the next chapter.