Title: Digital Image Processing

Eduardo A. B. da Silva and Gelson V. Mendonça

Prog. de Engenharia Elétrica e Depto. de Eletrônica

COPPE/EE/Federal University of Rio de Janeiro,

Caixa Postal 68504, Rio de Janeiro, 21945-970

Brazil

Digital image processing consists of the manipulation of images using digital computers. Its use has been increasing exponentially in the last decades. Its applications range from medicine to entertainment, passing by geological processing and remote sensing. Multimedia systems, one of the pillars of the modern information society, rely heavily on digital image processing.

The discipline of digital image processing is a vast one, encompassing the digital signal processing techniques as well as techniques which are specific to images. An image can be regarded as a function $f(x, y)$ of two continuous variables x and y. In order to be processed digitally, it has to be *sampled* and transformed into a matrix of numbers. Since a computer represents the numbers using finite precision, these numbers have to be *quantized* in order to be represented digitally. Digital image processing consists of the manipulation of those finite precision numbers. The processing of digital images can be divided into several classes, as image *enhancement*, image *restoration*, image *analysis* and image *compression*. In image *enhancement* an image is manipulated, mostly by heuristic techniques, so that a human viewer can extract useful information from it. Image *restoration* techniques aim at processing corrupted images from which there is a statistical or mathematical description of the degradation, so that it can be reverted. Image *analysis* techniques permit that an image be processed so that information can be automatically extracted from it. Examples of image analysis are image segmentation, edge extraction, texture and motion analysis. An important characteristic of images is the huge amount of information required to represent them. Even a gray-scale image of moderate resolution, say, 512×512, needs $512 \times 512 \times 8 \approx 2 \times 10^6$ bits for its representation. Therefore, in order to be practical to store and transmit digital images, one needs to perform some sort of *compression*, whereby the redundancy of the images is exploited for reducing the number of bits needed in its representation.

In what follows, we provide a brief description of digital image processing techniques.

Section 1 deals with image sampling and Section 2 describes image quantization. In Section 3 some image enhancement techniques are given. Section 4 analyses image restoration. Image compression, or *coding*, is presented in Section 5. Finally, Section 6 introduces the main issues involved in image analysis.

1 Image Sampling

An analog image can be seen as a function $f(x, y)$ of two continuous space variables, x and y. The function $f(x, y)$ represents a variation of gray level along the spatial coordinates. Since each color can be represented as a combination of 3 primaries (R, G, B), a color image can be represented by three functions, $f_R(x, y)$, $f_G(x, y)$ and $f_B(x, y)$, one for each color component.

A natural way to translate an analog image into the discrete domain is to sample each variable, that is, we make $x = n_x T_x$ and $y = n_y T_y$. This way, we get a digital image $f_D(n_1, n_2)$ such that

$$f_D(n_x, n_y) = f(n_x T_x, n_y T_y) \tag{1}$$

Likewise the one-dimensional case, one has to devise conditions under which an analog image can be recovered from its samples. In order to do that, we will first generalize the notions of frequency content and Fourier transform for two-dimensional signals.

One can represent an image in the frequency domain by using a two-dimensional Fourier transform. It is a straightforward extension of the one-dimensional Fourier transform, as can be seen in the equations below

$$F_(\Omega_x, \Omega_y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-j(\Omega_x x + \Omega_y y)} dx dy \tag{2}$$

$$f(x, y) = \frac{1}{4\pi^2} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(\Omega_x, \Omega_y) e^{j(\Omega_x x + \Omega_y y)} d\Omega_x d\Omega_y \tag{3}$$

The function $e^{j(\Omega_x x+\Omega_y y)}$ is a complex sinusoid in two dimensions. It represents a two-dimensional pattern that is sinusoidal both along the x (horizontal) and y (vertical) directions. Ω_x and Ω_y are the spatial frequencies along the x and y directions, respectively. An example can be seen in Figure 1, where it is depicted the two-dimensional sinusoid $\cos(\Omega_x x+\Omega_y y)$, with $\Omega_x = 2\Omega_y$. One can note that there are twice as many cycles in the x direction than in the y direction. The period in the x direction is $\frac{2\pi}{\Omega_x}$ and in the y direction is $\frac{2\pi}{\Omega_y}$.

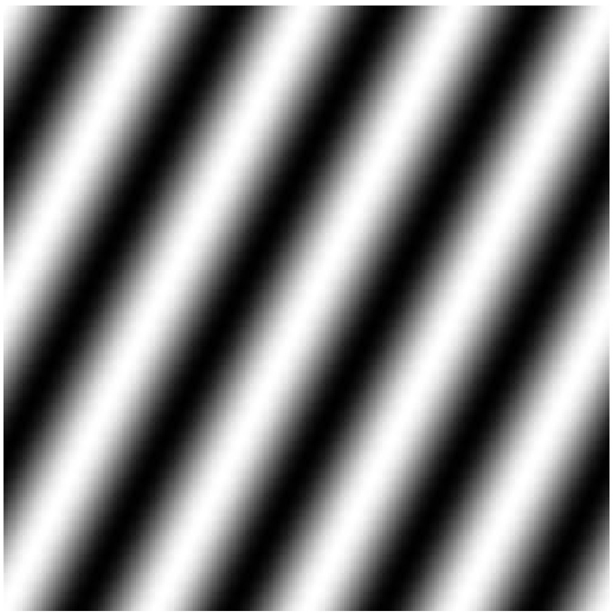


Figure 1: Two-dimensional sinusoid with $\Omega_x = 2\Omega_y$. Its expression is $f(x,y) = \frac{1}{2}\left(e^{j(\Omega_x x+\Omega_y y)} + e^{-j(\Omega_x x+\Omega_y y)}\right)$.

Therefore, equation (3) means that any two-dimensional signal is equivalent to an infinite sum of two-dimensional complex sinusoids. The term that multiplies the sinusoid of frequency (Ω_x, Ω_y) is its Fourier transform, $F(\Omega_x, \Omega_y)$, and is given by equation (2).

Having defined the meaning of frequency content of two-dimensional signals, we are now ready to state conditions under which an image can be recovered from its samples. It is well known that most useful digital images have the significant part of the amplitude spectrum concentrated at low frequencies. Therefore, an image in general can be modeled as a bandlimited two-dimensional signal, that is, its Fourier transform $F(\Omega_x, \Omega_y)$ is approximately zero outside a bounded region, as shown in Figure 2.

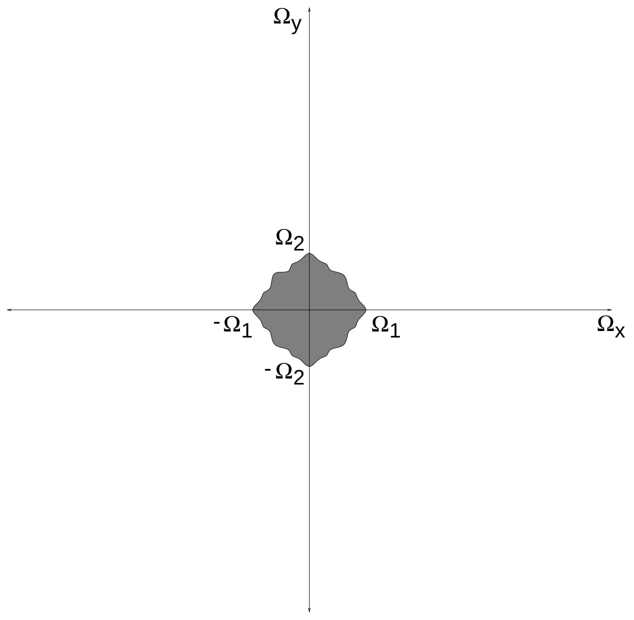


Figure 2: Support region of the Fourier transform of a typical image.

When the sampling process is performed according to equation (1), it is equivalent to sampling the continuous signal using a rectangular grid as in Figure 3. As in the one-dimensional case [25], one can show that the spectrum of the digital image is composed of replicas of the analog image spectrum, repeated with a period of $\frac{2\pi}{T_x}$ along the horizontal frequency axis, and $\frac{2\pi}{T_y}$ along the vertical frequency axis, as depicted in Figure 4.

The mathematical expression for the spectrum of the sampled signal is then

$$F_s(\Omega_x, \Omega_y) = \frac{1}{T_x T_y} \sum_{k_x=-\infty}^{\infty} \sum_{k_y=-\infty}^{\infty} F\left(\Omega_x - \frac{2\pi k_x}{T_x}, \Omega_y - \frac{2\pi k_y}{T_y}\right) \tag{4}$$

From the above equation and Figure 4 we note that if the sampling intervals do not satisfy the Nyquist conditions $\Omega_1 \leq \frac{2\pi}{T_x}$ and $\Omega_2 \leq \frac{2\pi}{T_y}$, then the analog signal cannot be reconstructed from its samples. This is due to the overlap of the replicated frequency spec-

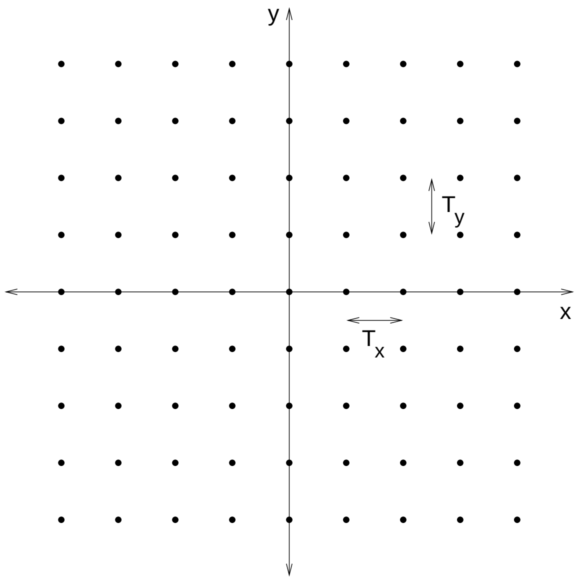


Figure 3: Sampling a two-dimensional signal using a rectangular grid.

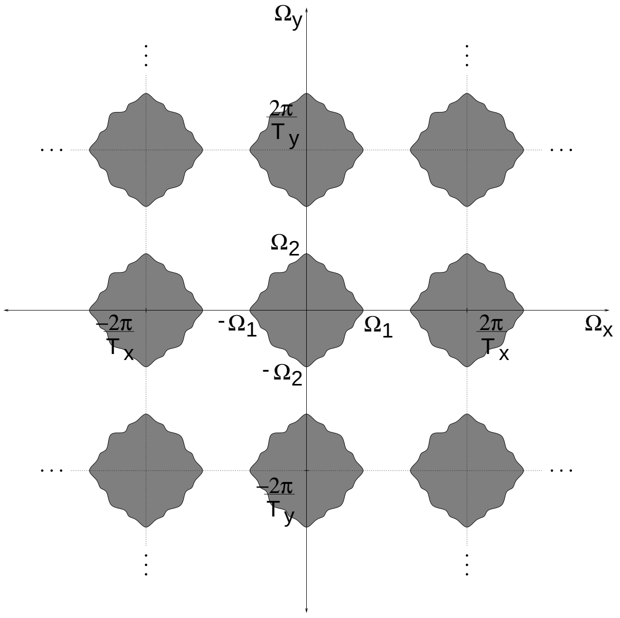


Figure 4: Spectrum of the two-dimensional discrete signal generated by sampling using a rectangular grid.

tra, which, likewise the one-dimensional case, is known as aliasing. Aliasing is illustrated in Figure 5.

Provided that aliasing is avoided, we have that, for obtaining the image signal back

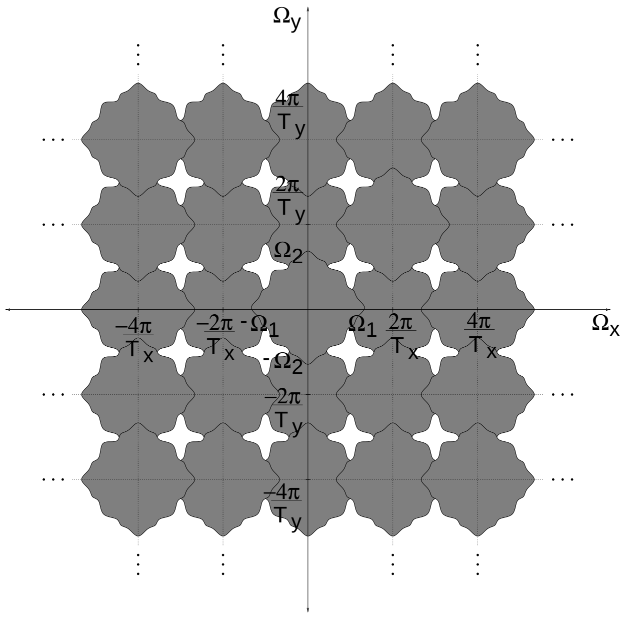


Figure 5: Aliased spectrum of the two-dimensional discrete signal generated by sampling using a rectangular grid.

into its analog form, the sampled signal has to be processed by a two-dimensional lowpass filter. This filter should ideally have a transfer function with amplitude spectrum equal to 1 in the support region of the Fourier transform of the original image and zero outside that region.

In Figure 6a we see the original LENA image. It has been sampled without aliasing. Figure 6b shows the LENA image sampled at a rate much smaller than the minimum sampling frequencies, in the horizontal and vertical directions, necessary to avoid aliasing. Aliasing can be clearly noted. In Figure 6c, the bandwidth of the image LENA has been reduced prior to sampling so that aliasing is avoided after sampling. Figure 6d shows the filtered image after sampling. We can notice that despite the blur on the image due to the lower bandwidth, there is indeed less aliasing, as indicated by the smoother edges.

Non-rectangular grid sampling: A two-dimensional signal does not necessarily need to be sampled using a rectangular grid like the one described by equation (1) and Figure 3.

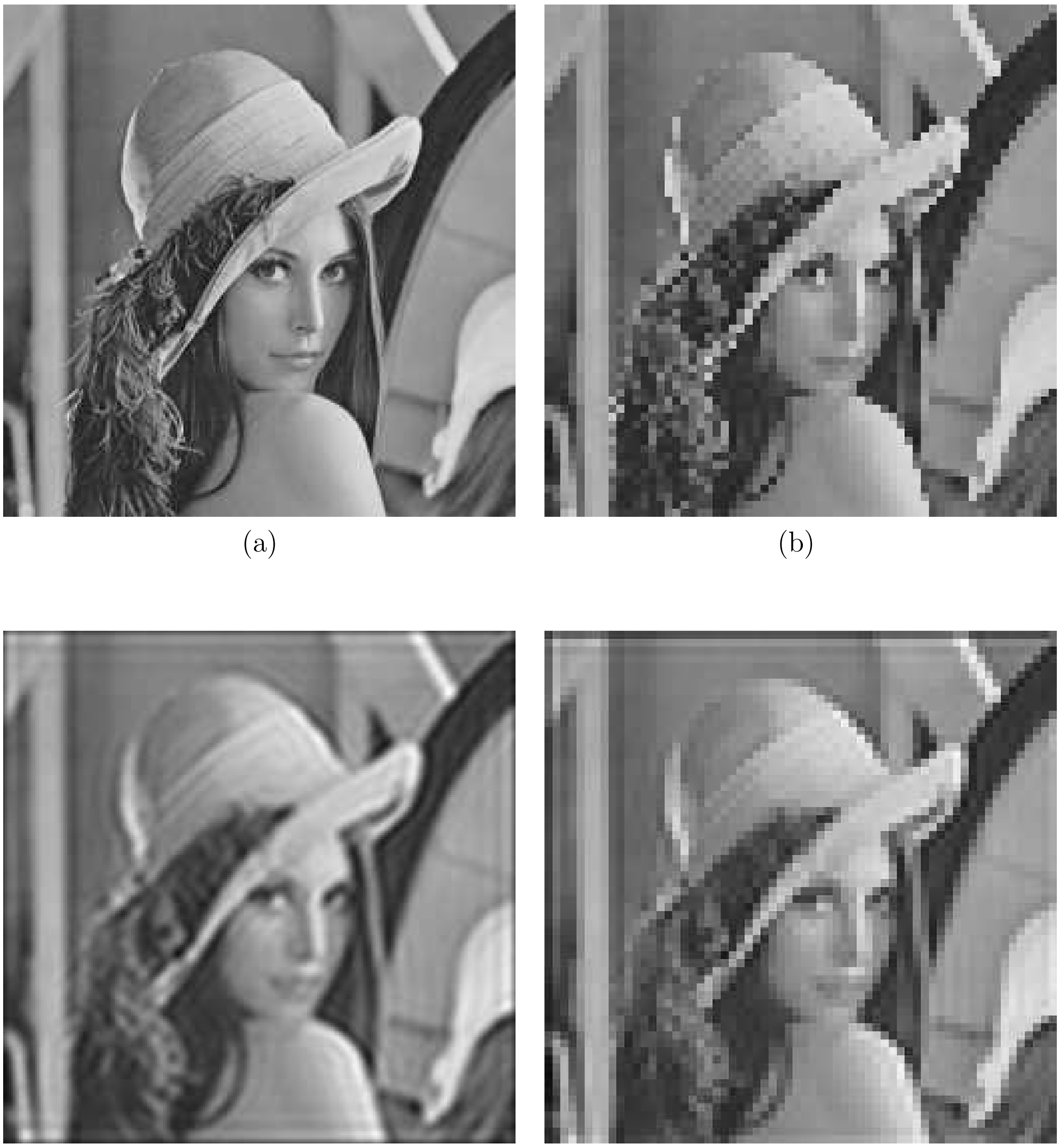


Figure 6: LENA image: (a) Original; (b) Sampled at below the sampling frequencies; (c) Lowpass filtered; (d) Lowpass filtered and sampled

For example, a non-rectangular grid as in Figure 7 can be used. The non-rectangular grid sampling can be best described by using vector and matrix notation. Let:

$$\mathbf{t} = \begin{pmatrix} t_x \\ t_y \end{pmatrix} \quad \mathbf{n} = \begin{pmatrix} n_x \\ n_y \end{pmatrix} \quad \mathbf{V} = \begin{pmatrix} v_{xx} & v_{xy} \\ v_{yx} & v_{yy} \end{pmatrix} = \begin{pmatrix} \mathbf{v}_x & \mathbf{v}_y \end{pmatrix} \quad \mathbf{\Omega} = \begin{pmatrix} \Omega_x \\ \Omega_y \end{pmatrix} \tag{5}$$

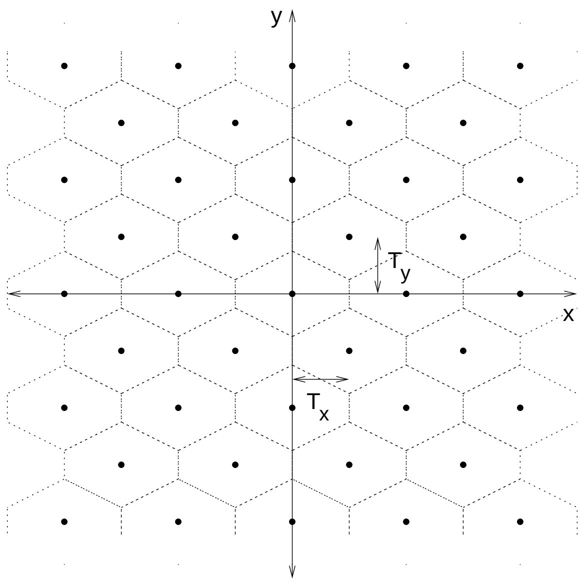


Figure 7: Sampling a two-dimensional signal using a non-rectangular grid.

The sampling process then becomes

$$f_D(\mathbf{n}) = f_D(n_x, n_y) = f(n_x\mathbf{v}_x + n_y\mathbf{v}_y) = f(\mathbf{V}\mathbf{n}) \tag{6}$$

In the case of Figure 7,

$$\mathbf{V} = \begin{pmatrix} T_x & T_x \\ T_y & -T_y \end{pmatrix} \tag{7}$$

It can be shown that in this case, the spectrum of the sampled signal becomes [24]

$$F_s(\Omega) = \frac{1}{|\det(\mathbf{V})|} \sum_{\mathbf{k}} F(\mathbf{\Omega} - 2\pi(\mathbf{V}^{-1})^t\mathbf{k}) \tag{8}$$

The matrix $2\pi(\mathbf{V}^{-1})^t = \mathbf{U}$ gives the periodicity, in the two-dimensional frequency plane, of the spectrum repetitions.

For the sampling grid defined in equation (7), we have that the periodicity matrix in the frequency plane is

$$\mathbf{U} = \begin{pmatrix} \frac{\pi}{T_x} & \frac{\pi}{T_x} \\ \frac{\pi}{T_y} & -\frac{\pi}{T_y} \end{pmatrix} \tag{9}$$

Then, the spectrum becomes as in Figure 8.

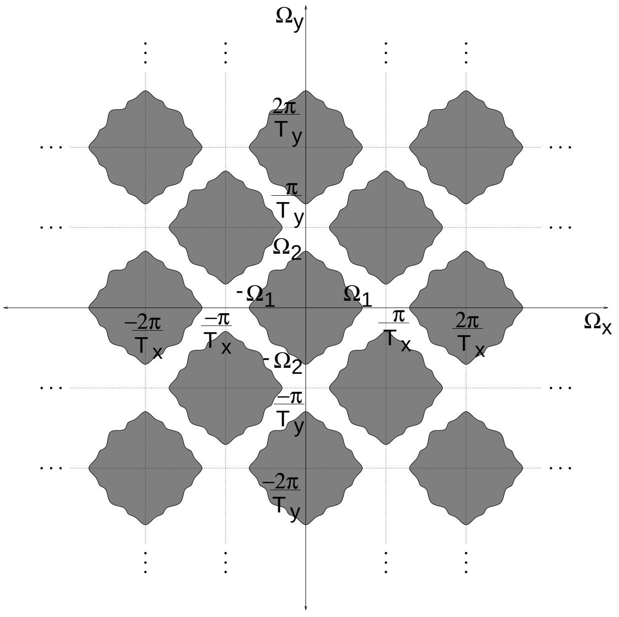


Figure 8: Spectrum of the two-dimensional discrete signal generated by sampling using a non-rectangular grid.

One particular non-rectangular grid of interest arises when we make $T_y = \sqrt{3}T_x$. We refer to it as *hexagonal* sampling. One advantage of such sampling grid over the rectangular one is that, for signals with isotropic frequency response, hexagonal sampling needs 13.4% fewer samples than the rectangular one for representing them without aliasing [24]. Such sampling grid is also useful when the continuous signals have a baseband as in Figure 2, as can be observed from Figures 4, 5 and 8. In fact, given the support region of a baseband signal, one can always determine the best sampling grid so that aliasing is avoided with the smallest possible density of samples.

2 Image Quantization

In order to be represented in a digital computer, every signal, once sampled, must be quantized. A digital image, for example, is a matrix of numbers represented with finite precision in the machine being used to process it. More generally, a quantizer is a function $Q(\cdot)$ such that

$$Q(x) = r_l \quad \text{for} \quad t_{l-1} < x \leq t_l \quad \text{for } l = 1, \ldots, L \tag{10}$$

The numbers r_l, $l = 1, \ldots, L$ are referred to as the *reconstruction levels* of the quantizer and t_l, $l = 0, \ldots, L$ are referred to as its *decision levels*. A typical quantizer is depicted in Figure 9.

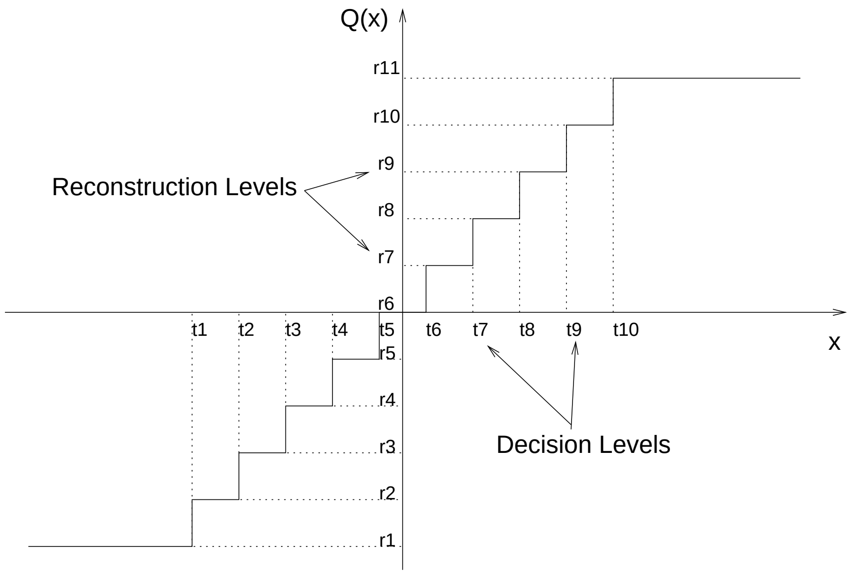


Figure 9: Typical quantizer.

In many cases, the decision and reconstruction levels of a quantizer are optimized such that the mean squared error between the quantizer's input and output is minimized. This is usually referred to as a Lloyd-Max quantizer [18]. In digital image processing, one often uses linear quantizers. For a linear quantizer, the reconstruction and decision levels satisfy the following relations:

$$t_0 = -\infty \tag{11}$$

$$t_{l+1} - t_l = q, \qquad \text{for } l = 1, \ldots, L \tag{12}$$

$$r_{l+1} = t_l + \frac{q}{2} \quad \text{for } l = 1, \ldots, L \tag{13}$$

$$t_L = \infty \tag{14}$$

The parameter q is referred to as the *quantizer step size.*

The mean square error of the linear quantizer, assuming that the quantization error $e(n) = x(n) - Q[x(n)]$ is uniformly distributed, is given by [18]

$$E[e_2(n)] = \frac{q^2}{12} \tag{15}$$

Since the human eye cannot in general recognize more than around 256 gray levels, usually 8 bits per pixel are enough for representing a gray-scale image. In the case of color images, one uses 8 bits for each color component. For RGB images, this gives 24 bits/pixel.

In many occasions, one needs to quantize an image with less then 8 bits/pixel. In Figure 10a, it is shown the image LENA quantized with 4 bits/pixel. One can notice the effect of false contours. One way to overcome this problem is to perform *dithering* [18]. It consists in adding pseudo-random noise to the image prior to quantization. Before display, this noise can be either subtracted from the image or not. This is illustrated in Figures 10a to 10c. Figure 10b shows the LENA image with a pseudo-random noise uniformly distributed in the interval [-25,25] quantized with 8 levels. Figure 10c shows the image in Figure 10b with the pseudo-random noise subtracted from it. One can notice that in both cases there is a great improvement in visual quality when comparing with Figure 10a. The false contours almost disappear.

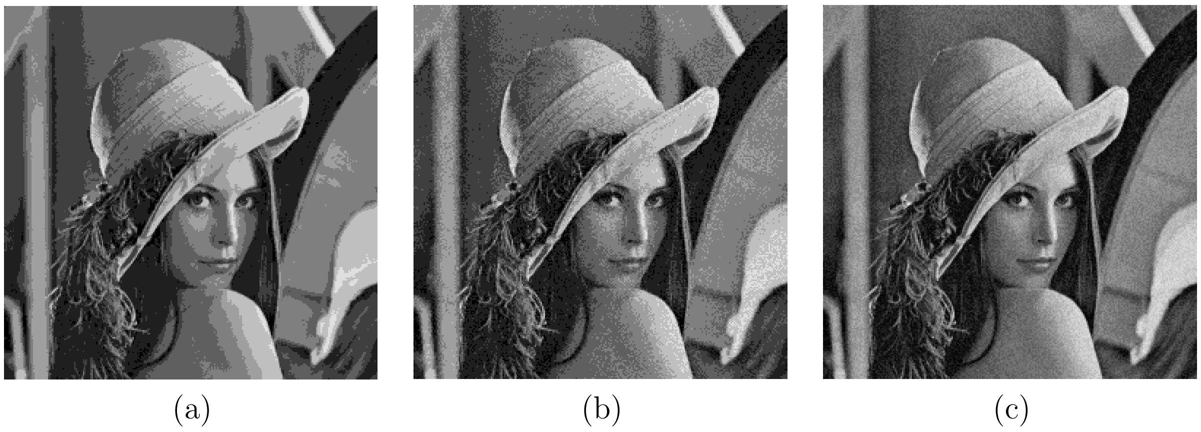


Figure 10: LENA image: (a) Quantized with 8 levels; (b) Quantized with 8 levels with noise added prior to quantization; (c) The image in (b) with the noise also subtracted after quantization.

3 Image Enhancement

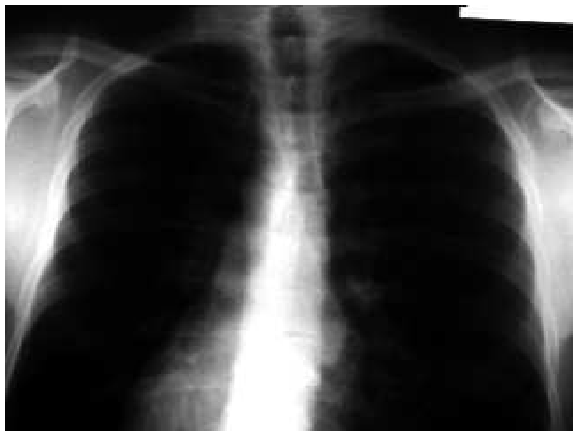
Image enhancement is the name given to a class of procedures by which an image is processed so that it can be better displayed or improved for a specific analysis. This can be obtained at the expense of other aspects of the image. There are many enhancement techniques that can be applied to an image.

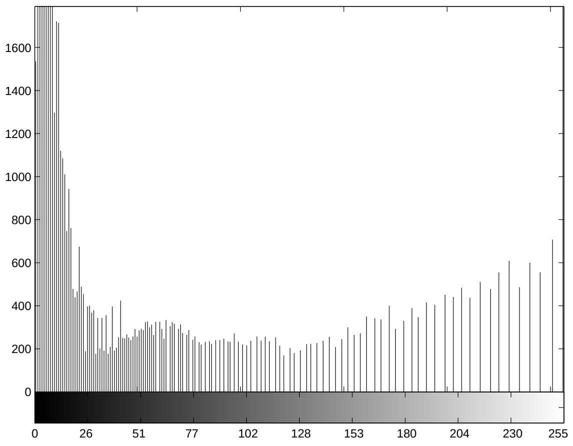
In order to give a flavor of the image enhancement techniques, in what follows we are going to give two examples of image enhancement techniques, namely, histogram manipulation and image filtering.

3.1 Histogram manipulation

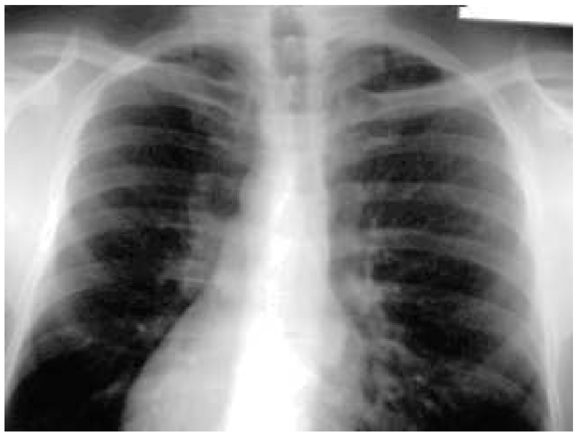
The histogram of an image is a function that maps each gray level of an image to the number of times it occurs in the image. For example, the image in Figure 11a has the histogram shown in Figure 11b.

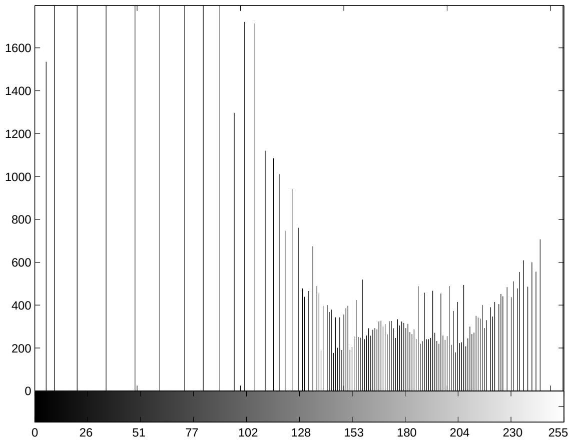
One should note that the pixels have, in general, its gray levels in the integer range


(a)



(b)


(c)



(d)

Figure 11: (a) Original image; (b) Histogram of the original image; (c) Image with equalized histogram; (d) Histogram of the equalized image.

$[0, 255]$.

Histogram equalization

By looking at Figure 11a, one notices that the image is too dark. This can be confirmed by its histogram in Figure 11b, where one can see that the most frequent gray levels have low values. In order to enhance the appearance of the image, one would need to re-map the images gray levels so that they become more uniformly distributed. Ideally, one would need to apply a transformation that would make the histogram of the image look uniform. If $F_U(u) = \int_0^u p_U(x)dx$ is the distribution function of the image, then this transformation would be $y = F^{-1}(x)$ [9]. In practice, since the pixels can attain only integer values, this operation cannot be performed exactly, and some sort of quantization must be carried out [18].

Figure 11c shows the image with equalized histogram and Figure 11d shows its histogram. Note that the quality of the image is far superior to the original one, and the histogram is much more uniform then the one in Figure 11b.

Another similar histogram manipulation technique is *histogram specification*, where we try to make the histogram of an image to be as similar as possible to a given one [9, 18]. Some texts refer to *histogram matching.* It is a kind of histogram specification technique in which the histogram of an image is matched to the one of another image. It can be used, for example, when we have two images of the same scene taken from two different sensors.

3.2 Linear Filtering

Linear filtering is one of the most powerful image enhancement methods. It is a process in which part of the signal frequency spectrum is modified by the transfer function of the filter. In general the filters under consideration are linear and shift-invariant, and thus the output images are characterized by the convolution sum between the input image and the filter impulse response, that is

$$y(m,n) = \sum_{i=0}^{M}\sum_{j=0}^{N} h(m-i,n-j)x(i,j) = h(m,n) * *x(m,n) \tag{16}$$

where:

- $y(m,n)$ is the output image;
- $h(m,n)$ is the filter impulse response;

- $x(m, n)$ is the input image.

For example, lowpass filtering has the effect of smoothing an image, as can be observed from Figure 6c. On the other hand, highpass filtering usually sharpens the edges of an image. They can even be used for edge detection, which is used in image analysis algorithms.

The image filtering can be carried out either in the spatial domain, as in equation (16), or in the frequency domain, using the Discrete Fourier Transform (DFT) [24, 25]. For it, we use the well know property that the DFT of the circular convolution of two sequences is equal to the product of the DFTs of the two sequences. That is, for $y(m, n)$ defined as in equation (16), provided that a DFT of sufficient size is used, we have that

$$\mathrm{DFT}\left\{y(m,n)\right\} = \mathrm{DFT}\left\{h(m,n)\right\}\mathrm{DFT}\left\{x(m,n)\right\} \tag{17}$$

Therefore, one can perform image filtering in the frequency domain by modifying conveniently the DFT of the image, and taking the inverse transformation. This is exemplified in Figures 12a to 12d.

In Figure 12a we see an image contaminated with periodic noise. Figure 12b shows the DFT of this image. One can clearly see the periodic noise as two well defined points on the DFT of the image. For convenience, arrows are pointing to them. In Figure 12c, one can see the DFT of the image with the periodic noise removed, that is, the frequency locations corresponding to the periodic noise were made equal to zero. Figure 12d shows the inverse DFT of the image in Figure 12c. One can notice that the noise has been effectively removed, improving a great deal the quality of the image.

Image enhancement is an extensive topic. In this text we just aimed at giving a flavor of it by providing typical examples. For a more detailed treatment of it, see [9, 18].

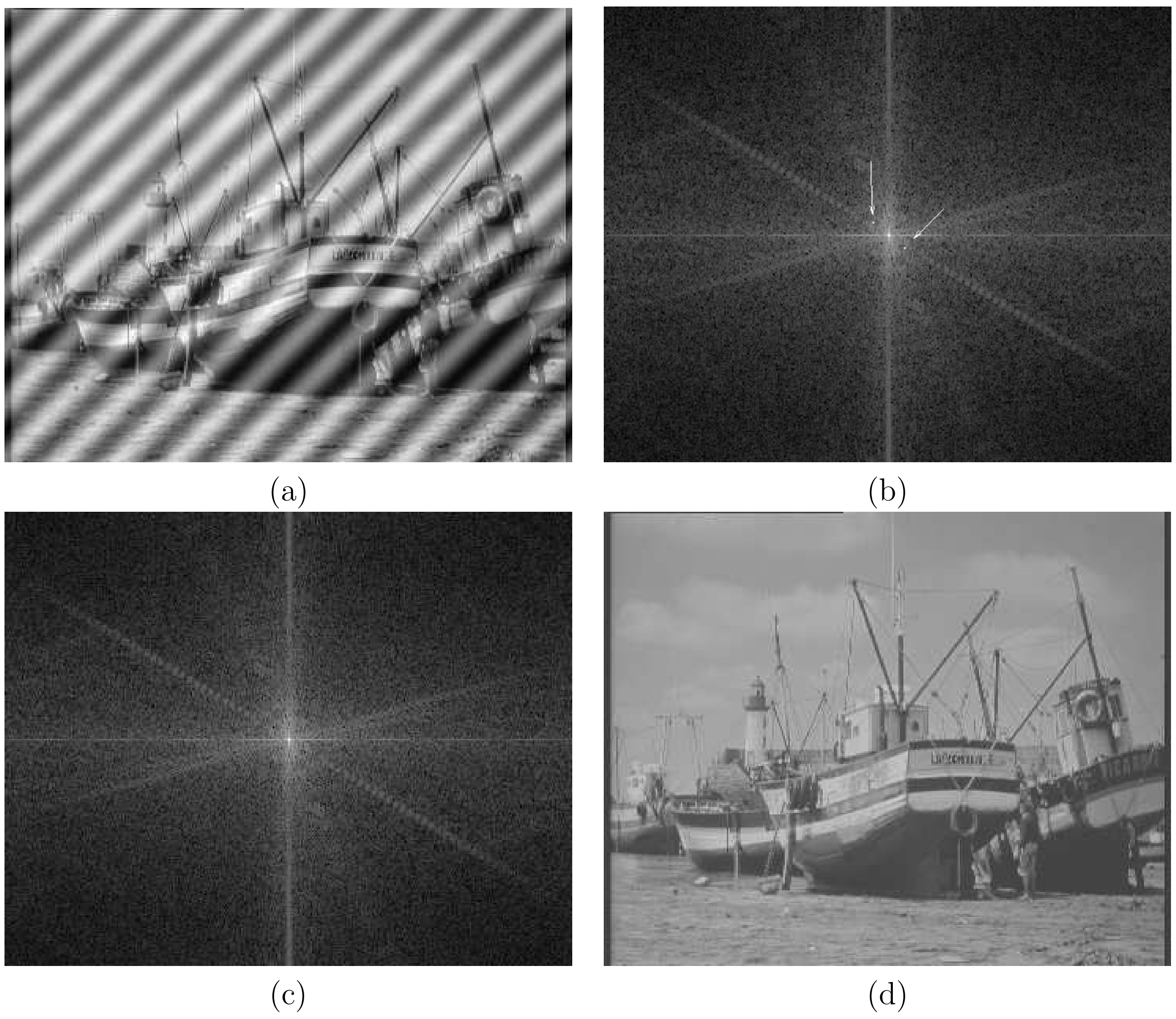


Figure 12: (a) Image contaminated with periodic noise; (b) DFT of (a), with the points corresponding to the noise highlighted; (c) DFT of (a) with the periodic noise removed; (d) Recovered image.

4 Image Restoration

In image restoration one is usually interested in recovering an image from a degraded version of it. It is essentially different from image enhancement, which is concerned with accentuation or extraction of image features. Moreover, in image restoration one usually has mathematical models of the degradation and a statistical description of the ensemble of images used. In this section we will study one type of restoration problem, namely, the recovery of images degraded by a linear system, contaminated with additive noise. A linear model for degradations is depicted in Figure 13. The matrix $u(m, n)$ is the original image; $h(m, n)$ is a linear system representing the degradation; $\eta(m, n)$ is an additive noise and $v(m, n)$ is the observed degraded image. We can write

$$v(m, n) = u(m, n) * *h(m, n) + \eta(m, n) \tag{18}$$

where "$**$" represents a two-dimensional convolution (see equation (16)).

An illustrative case is when $\eta(m, n) = 0$. In this situation, we can recover $u(m, n)$ from $v(m, n)$ by using a linear filter $g(m, n)$ such that $g(m, n) * *h(m, n) = \delta(m, n)$. In the frequency domain, this is equivalent to having

$$G(\omega_x, \omega_y) = \frac{1}{H(\omega_x, \omega_y)} \tag{19}$$

This procedure is called *inverse filtering.* The main problem with the inverse filter is that it is not defined in the cases that there is a pair (ω_1, ω_2) such that $H(\omega_1, \omega_2) = 0$. A solution to this problem is the *pseudo-inverse filter*, defined as

$$\overline{G}(\omega_x, \omega_y) = \begin{cases} \dfrac{1}{H(\omega_x, \omega_y)}, & H(\omega_x, \omega_y) \neq 0 \\ 0, & H(\omega_x, \omega_y) = 0 \end{cases} \tag{20}$$

As an example, we analyze now the restoration of images degraded by blur. For image blur caused by excessive film exposure time in the presence of camera motion in the horizontal direction, this degradation can be modeled, in the case of absence of additive noise, as

$$v(x, y) = \frac{1}{T} \int_0^T u(x - ct, y)dt \tag{21}$$

where the relative speed between the camera and the scene is c and the camera exposure

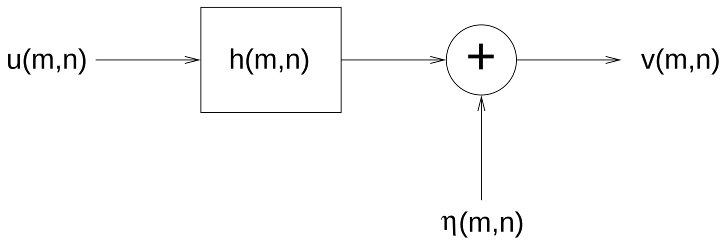


Figure 13: Image degradation model.

time is T. In the digital domain, this is equivalent to the following operation

$$v(m,n) = \frac{1}{L}\sum_{l=0}^{L-1} u(m-l,n) \tag{22}$$

Figure 14a shows the ZELDA image, of dimensions 360×288, blurred using equation (22) with $L = 48$. Applying the pseudo-inverse filter (see equation (20)) in the frequency domain (using a DFT), we obtain Figure 14b. We notice that the recovery can be quite good. It is important to note that, in this case, since the process is supposed to be noiseless, the degradation had to be performed using full machine precision. If we quantize the degraded image using 256 levels, as is usual in image processing applications, a small quantization noise is superimposed on the degraded image. As we have seen in Section 2, this quantization noise is invisible to the human eye. On the other hand, applying the pseudo-inverse filter to the quantized degraded image, we obtain the image in Figure 14c. We can clearly see that the restoration process has failed completely. The excessive noise present in the restored image is due to the fact that, since the quantization noise is relatively white, there is noise information even around the frequencies where $H(\omega_1,\omega_2)$ is very small, and, as a consequence, $G(\omega_1,\omega_2)$ is very large. This greatly amplifies the noise around these frequencies, thus producing an image like the one in Figure 14c. In fact, one would need a filter that is more or less equivalent to the pseudo-inverse in frequencies where the signal to noise ratio is high, but has little effect in regions where the signal to

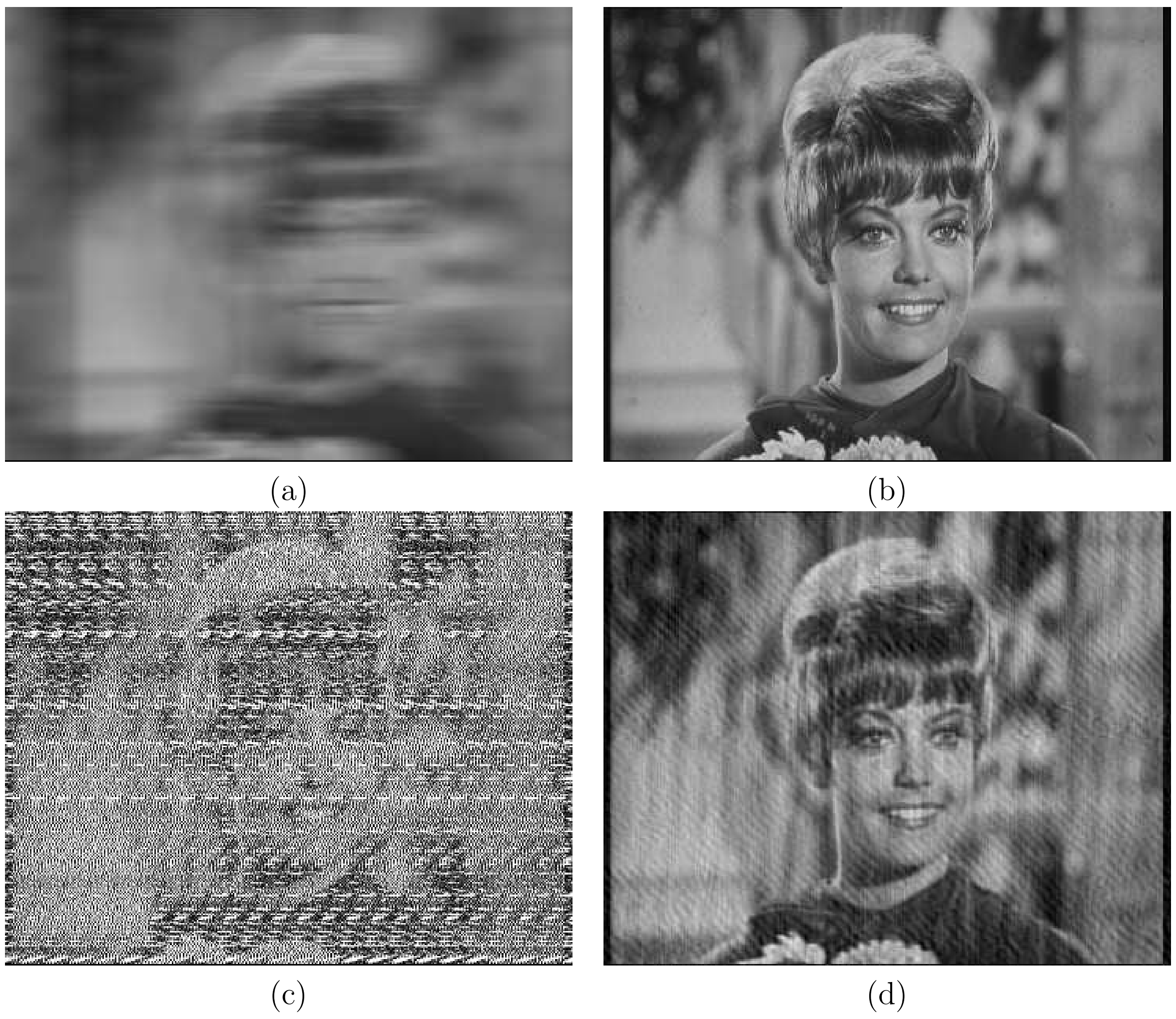


Figure 14: (a) Image degraded by blur; (b) Image restored using the pseudo inverse filter; (c) Image in (a), quantized with 256 levels, after application of the pseudo-inverse filter; (d) Image in (a), quantized with 256 levels, after application of the Wiener filter

noise ratio is low. One way to accomplish this is using the *Wiener filter*, which is optimum for a given image and noise statistics. We describe it in what follows.

Wiener filter

Given the image degradation model in Figure 13, suppose that $x(m,n)$ and $\eta(m,n)$ are zero mean, stationary random processes [26]. The image restoration problem can be formally stated as: Given $v(m,n)$, find the best estimate of $u(m,n)$, $\hat{u}(m,n)$, such that the mean squared estimation error $E[e^2(m,n)]$ is minimum, where

$$e(m,n) = u(m,n) - \hat{u}(m,n) \tag{23}$$

Such best estimate $\hat{u}(m,n)$ is known to be [26, 18]

$$\hat{u}(m,n) = E[u(m,n)|v(k,l), \forall(k,l)] \tag{24}$$

In our case, we are looking for linear estimates, that is, $\hat{u}(m,n)$ of the form

$$\hat{u}(m,n) = v(m,n) * *g(m,n) \tag{25}$$

Accordingly to the orthogonality principle [26], we have the best estimate when

$$E[e(m_1,n_1)v^*(m_2,n_2)] = 0, \quad \forall m_1,n_1,m_2,n_2 \tag{26}$$

where * represents the complex conjugation.

From equation (23), we have that the above equation is equivalent to

$$E[\{u(m_1,n_1) - \hat{u}(m_1,n_1)\}v^*(m_2,n_2)] = 0, \quad \forall m_1,n_1,m_2,n_2 \tag{27}$$

For stationary processes, equation (27) implies that

$$r_{uv}(m,n) = r_{\hat{u}v}(m,n) \tag{28}$$

where $r_{ab}(m,n)$ is the cross-correlation [26] between signals $a(m,n)$ and $b(m,n)$.

If $S_{ab}(\omega_x,\omega_y)$ is the cross power spectral density [26] between signals $a(m,n)$ and $b(m,n)$, the above equation implies that

$$S_{uv}(\omega_x,\omega_y) = S_{\hat{u}v}(\omega_x,\omega_y) \tag{29}$$

Since, from equation (25), $\hat{u}(m,n) = v(m,n) * *g(m,n)$, equation (29) is equivalent to

$$S_{\hat{u}v}(\omega_x,\omega_y) = G(\omega_x,\omega_y)S_{vv}(\omega_x,\omega_y) \tag{30}$$

From equation (29), this implies that

$$G(\omega_x, \omega_y) = \frac{S_{uv}(\omega_x, \omega_y)}{S_{vv}(\omega_x, \omega_y)} \tag{31}$$

Since, from Figure 13 and equation (18),

$$v(m, n) = u(m, n) * *h(m, n) + \eta(m, n)$$

and $u(m, n)$ and $v(m, n)$ are considered uncorrelated, we have that

$$S_{vv}(\omega_x, \omega_y) = S_{uu}(\omega_x, \omega_y)|H(\omega_x, \omega_y)|^2 + S_{\eta\eta}(\omega_x, \omega_y) \tag{32}$$

$$S_{uv}(\omega_x, \omega_y) = S_{uu}(\omega_x, \omega_y)H^*(\omega_x, \omega_y) \tag{33}$$

Then, from equation (31), the above equations imply that the optimum linear estimation filter is

$$G(\omega_x, \omega_y) = \frac{S_{uu}(\omega_x, \omega_y)H^*(\omega_x, \omega_y)}{S_{uu}(\omega_x, \omega_y)|H(\omega_x, \omega_y)|^2 + S_{\eta\eta}(\omega_x, \omega_y)} \tag{34}$$

This is known as the *Wiener filter*. Please note that the implementation of the Wiener filter requires, besides the knowledge of the degradation process $H(\omega_x, \omega_y)$, the knowledge of the power spectral densities of the ensemble of the original inputs, $S_{uu}(\omega_x, \omega_y)$ and of the noise, $S_{\eta\eta}(\omega_x, \omega_y)$.

Figure 14d shows the image recovered from the blurred and quantized image. In the implementation of the Wiener filter, it was supposed that the ensemble of the input could be modeled by a first order Gauss-Markov process (AR(1)) with correlation coefficient equal to 0.95. Since the image has a dynamic range of 256 and was quantized with quantization interval 1, the noise was supposed to be white, uniform with density 1/12 (see equation (15)). We note that the recovery is far better than the one obtained with the pseudo-inverse filter.

We can rewrite equation (34) as

$$G(\omega_x, \omega_y) = \frac{H^*(\omega_x, \omega_y)}{|H(\omega_x, \omega_y)|^2 + \dfrac{S_{\eta\eta}(\omega_x, \omega_y)}{S_{uu}(\omega_x, \omega_y)}} \tag{35}$$

A couple of interesting conclusions can be drawn from the above equation. The first one is that, in regions of the frequency plane where the signal to noise ratio is high, that is, $\frac{S_{\eta\eta}(\omega_x,\omega_y)}{S_{uu}(\omega_x,\omega_y)} \to 0$, then equation (35) is equivalent to equation (20), that is, the one of the pseudo-inverse filter. On the other hand, in regions of the frequency plane where the signal to noise ratio is low, then the Wiener filter depends on the power spectral densities of both the noise and the input.

In this section, we attempted to provide the reader with a general notion of the image restoration problem, which is still an active area of research [3]. For a deeper treatment of the subject, the reader is referred to [9, 18, 21, 28, 29].

5 Image Coding

One of the main difficulties when it comes to image processing applications resides in the large amount of data required to represent an image. For example, one frame of a standard definition digital television image has dimensions 720×480 pixels. Therefore, using 256 gray levels (8 bits) for each color component, it requires $3 \times 8 \times 720 \times 480 = 8,294,400$ bits to be represented. Considering that in digital video applications one uses 30 frames/second, one second of video needs more than 240M bits! Thus, in order for the processing, storage and transmission of images to be feasible, one needs to reduce somehow this number of bits. *Image compression*, or *coding*, is the discipline that deals with forms of achieving this. In order to better understand the image compression problem, consider the 256×256 image in Figure 15. If we need to represent this image as it is, one would need 8 bits/pixel, giving a total of $8 \times 256 \times 256 = 524,288$ bits. Alternatively, if we know that the image in Figure 15 is a circle, it would be enough to say that it represents a circle of radius 64, gray level 102, with its center at coordinates $x = 102$ and $y = 80$ and superimposed on a background of gray level 255. Since the radius, x-coordinate and y-coordinate are between

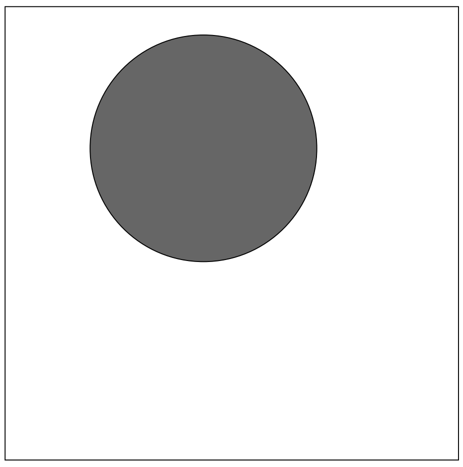


Figure 15: Illustration of the image compression problem.

0 and 255, one would need 8 bits for specifying each one; since both the gray level of the circle and of the background need 8 bits each, the whole image in Figure 15 would need just 40 bits to be specified. This is equivalent to a compression of more than 13,000:1!

However, the vast majority of images is not as simple as the one in Figure 15, and such amount of compression cannot be easily obtained. Fortunately, most useful images are redundant, that is, their pixels carry a lot of superfluous information. This can be better understood by resorting to the images in Figure 16.

By looking at Figure 16a, we note that, by taking a pixel, for example, at the girl's face, that the pixels around it have, on average, values similar to it. Certainly there are pixels where this does not hold, as for example, at the boundary of the girl's skin and hair. However, this is true of most of its pixels. Examining the images in Figures 16b and 16c, we see that the same reasoning applies to them. In other words, if we know something about one pixel, it is likely that we can guess, with a good probability of success, many things about the pixels around it. This implies that we do not need all their pixels to represent them, only the ones carrying information which cannot be guessed.

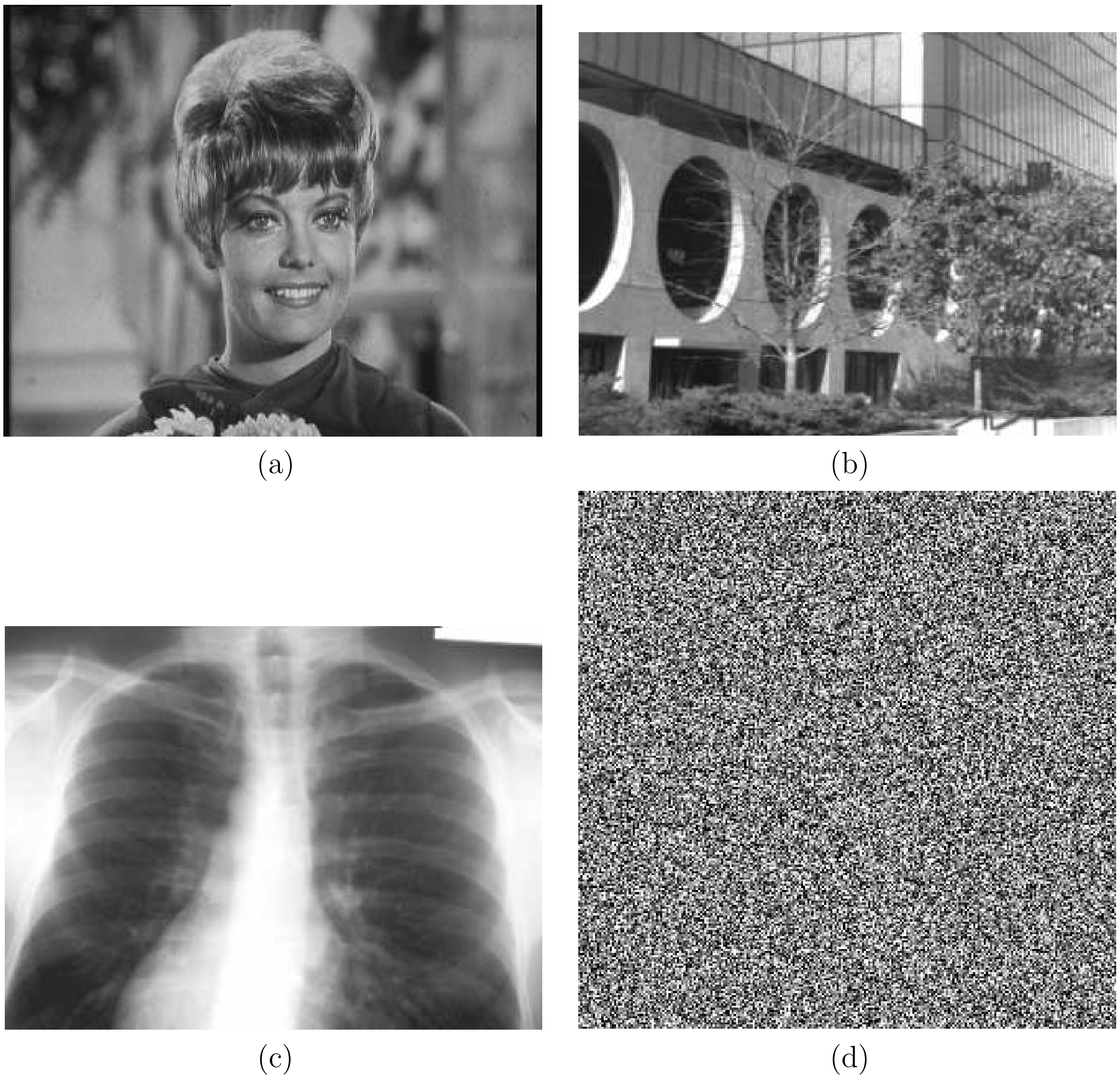


Figure 16: (a), (b), and (c): Redundant images; (d) A non-redundant image.

At this point, a question arises: are all images redundant? The answer is no, as one can see in Figure 16d. This image represents uniform distributed white noise between zero and 255. In it, to know something about a given pixel does not imply knowing anything about the pixels around it. However, such an image is not useful for carrying information, and is not of the kind frequently present in nature. Indeed, one can state that most natural images are redundant. Therefore, they are amenable to compression. How to exploit this redundancy to create a compact representation of an image is the main objective of the image compression techniques.

In general, an image compression method can be split into three basic steps, as depicted in Figure 17. They are transformation, quantization and coding. In the *transformation*

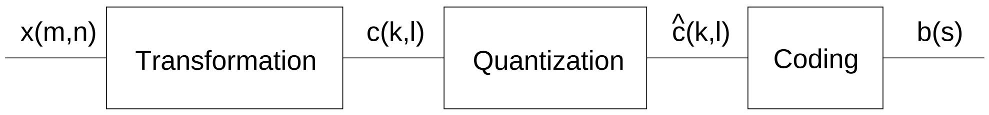


Figure 17: Three basic steps involved in image compression.

step, a mathematical transformation is applied to the image pixels $x(m, n)$, generating a set of coefficients $c(k, l)$ with less correlation between themselves. Its main objective is to exploit the correlation between the image pixels. In order for these coefficients $c(k, l)$ to be represented with a limited number of bits, they have to be mapped to a finite number of symbols, $\hat{c}(k, l)$. This mapping is commonly referred to as *quantization*. In Section 2 we have introduced some forms of quantization. After the coefficients are mapped to a finite number of symbols, these symbols $\hat{c}(k, l)$ must be mapped to a string of bits $b(s)$ in order to be either transmitted or stored. This operation is commonly referred to as *coding*. Actually, we have compression only when the number of bits in $b(s)$, $s = 0, 1, \ldots$ is smaller than the number of bits needed to represent the pixels $x(m, n)$ themselves. These three steps can be clearly illustrated in the next example, *Differential Pulse-Coded Modulation* (DPCM).

DPCM

When looking at the redundant images in Figures 16a to 16c, one notices, for example, that if we know the value of a pixel $x(m, n - 1)$, a good guess for the value of the pixel on its right, $\overline{x}(m,n)$, would be the value of the pixel $x(m,n-1)$. A way to reduce the redundancy on those images would be to transmit or encode only the “part” of the pixel that cannot be guessed. This is represented by the difference between the actual value of the pixel and the guess, that is, one would encode $c(m,n)$ as below

$$c(m,n) = x(m,n) - \overline{x}(m,n) = x(m,n) - x(m,n-1) \tag{36}$$

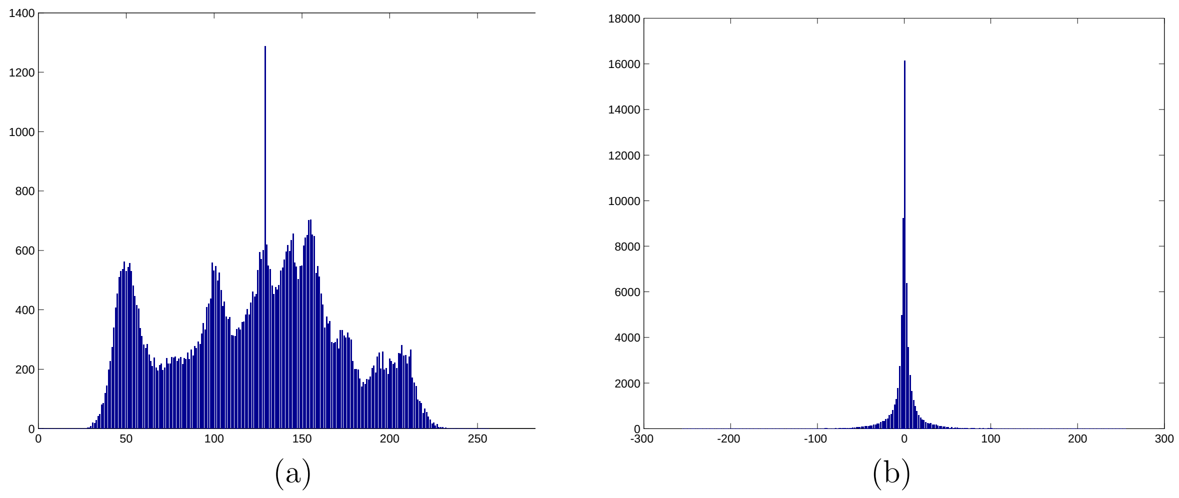


Figure 18: (a) Histogram of the image LENA; (b) Histogram of the coefficients $c(m,n)$ generated according to equation (36).

the histogram of the original image in Figure 18a has nothing special about it, that is, it has a more or less random shape. On the other hand, one can see that the histogram of the coefficients $c(m,n)$ has a well defined shape, tending to a Laplacian distribution. In it, we see that indeed the small differences $c(m,n)$ are much more likely than the smaller differences, what confirms the redundancy of the LENA image. A remarkable fact is that the histogram of the differences has a similar distribution for almost any type of natural images. Then, in fact, the *transformation* we performed, consisting of taking differences between adjacent pixels, has effectively reduced the redundancy of the original image. In addition, since the smaller values are much more probable than the larger ones, if we use a variable-length code to represent the coefficients, where the more likely values are represented with a smaller number of bits and the less likely values are represented with a larger number of bits, we can effectively obtain a reduction on the number of bits used. It is well known from information theory [7] that, by properly choosing a code, one can encode a source using a number of bits arbitrarily close to its *entropy*, defined as

$$H(S) = -\sum_{i=1}^{L} p_i \log_2 p_i \tag{37}$$

where the source S consists of L symbols $\{s_1, \ldots, s_L\}$, with symbol s_i having probability of occurrence p_i. Examples of such codes are *Huffman codes* [7] and *arithmetic codes* [4]. A discussion of them is beyond the scope of this article. For a good treatment of them, the reader is referred to [4, 7].

For example, the image LENA has the histogram in Figure 18a. It corresponds to an entropy of 7.40 bits/pixel, and thus one needs at least 8 bits to represent each pixel of the image LENA. On the other hand, the entropy of the coefficients, whose histogram is in Figure 18b is 3.18 bits/coefficient, and therefore one can find a Huffman code that just needs 4 bits to represent each of them. This is equivalent to a compression of 2:1. This compression has been achieved by the combination of the *transformation* (taking differences) and *coding* (Huffman codes) operation.

It is noteworthy that the compression obtained in the above example has been achieved without any loss, that is, the input image could be recovered, without error, from its compressed version. This is referred to as *lossless compression.* In some circumstances one is willing to give up the error-free recovery in order to obtain larger compression rates. This is referred to as *lossy compression.* In the case of images, lossy compression is used a great deal, specially when one can introduce distortions with just a low degree of visibility [37].

These higher compression rates can be obtained through *quantization*, as described in Section 2. In the example above, the differences $c(m, n)$ have a dynamic range from $0 - 255 = -255$ to $255 - 0 = 255$. Therefore, there are 512 such differences. By applying quantization as in Figure 9, the number of possible differences can be decreased, at the expense of some distortion being introduced. Figure 19 describes a practical way of achieving a lossy compression using DPCM. Notice that, in this case, due to quantization, the decoder cannot recover the original pixels without loss. Therefore, instead of $c(m, n)$ being computed as in equation (36), with $x(m, n-1)$ being the prediction for the pixel $x(m, n)$, the prediction for $x(m, n)$ is the "recoverable" value of $x(m, n-1)$ at the decoder. Note that, in order to achieve this, there must be a perfect copy of the decoder in the encoder. Details of this kind of coder/decoder can be found in [19].

For the LENA image, if one uses a 16-level quantizer of the same type as the one in Figure 9, then the histogram of the quantized differences $Q[c(m, n)]$ is as in Figure 20a. The bit rate necessary to represent them can be made arbitrarily close to the entropy by means of conveniently designed Huffman codes. In this case, the entropy is 1.12 bits/pixel. Therefore, the reconstructed image in Figure 20b is the result of a compression more than 7:1. As can be seen, despite the high compression ratio achieved, the quality of the reconstructed image is quite acceptable.

This example highlights the nature and effect of the three basic operations described in Figure 17: *transformation* for reducing the redundancy between the pixels; *quantization* for reducing the number of symbols to encode; *coding* to associate to the generated symbols, the smallest average number of bits possible.

Transform coding

In the DPCM encoder/decoder discussed above, the transformation consisted of taking

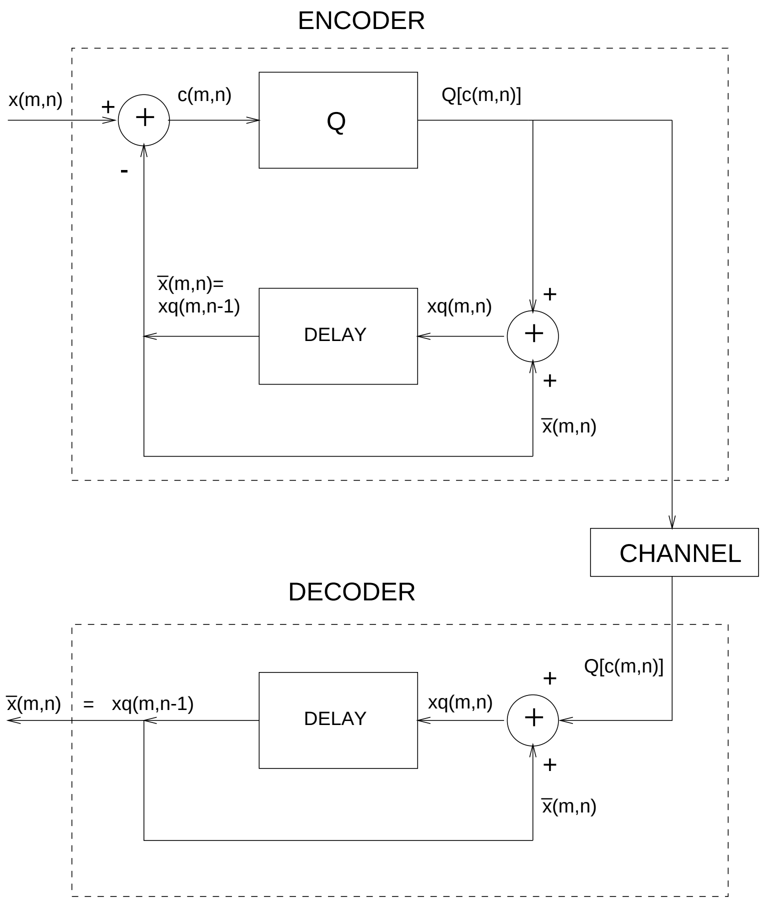
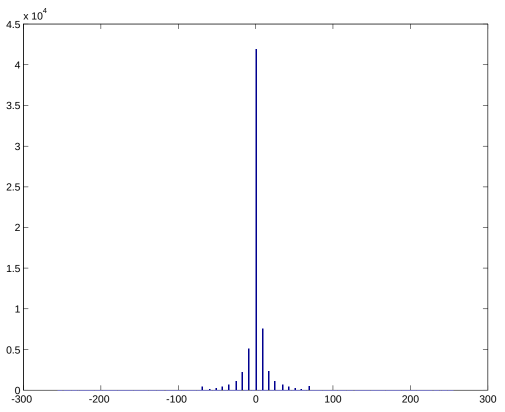


Figure 19: DPCM encoder/decoder.

the differences between a pixel and the adjacent pixel to its left. It indeed reduced the redundancy existent within the image. However, only the redundancy between two adjacent pixels have been exploited. By looking at images like the ones in Figure 16a, 16b and 16c, we notice that a pixel has a high correlation with many pixels surrounding it. This implies that not only the redundancy between adjacent pixels can be exploited, but also the redundancy in an area of the image. By exploiting this, one expects to obtain even higher compression ratios. *Transform coding* is the generic name given to compression methods in which this type of redundancy reduction is carried out by means of a linear transform [6].

By far, the most widely used linear transform used in compression schemes up to date



(a)

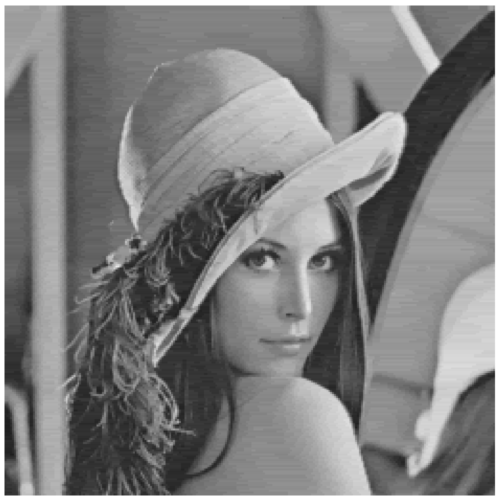

(b)

Figure 20: (a) Histogram of the quantized differences $Q[c(m,n)]$ in Figure 19, for the case of the LENA image; (b) Recovered image $\overline{x}(m,n)$.

is the *Discrete Cosine Transform* (DCT). It is part of the old JPEG standard [27], as well as most modern video compression standards, like H.261 [17], H.263 [12], MPEG1 [13], MPEG2 [14] and MPEG4 [16]. The DCT of a length-N signal $x(n)$ consists of the coefficients $c(k)$ such that [1]

$$c(k) = \alpha(k) \sum_{n=0}^{N-1} x(n) \cos\left[\frac{\pi(2n+1)k}{2N}\right] \tag{38}$$

where $\alpha(0) = \sqrt{\frac{1}{N}},\ \alpha(k) = \sqrt{\frac{2}{N}},\ 1 \leq k \leq N-1.$

Given the DCT coefficients, the original signal can be recovered using

$$x(n) = \sum_{k=0}^{N-1} c(k)\alpha(k) \cos\left[\frac{\pi(2n+1)k}{2N}\right] \tag{39}$$

For images, the DCT can be calculated by first computing the one-dimensional DCT of all the rows of the image and then computing the one-dimensional DCT of all the columns of the result.

The DCT is very effective in reducing the redundancy of real images. It tends to concentrate energy in very few transform coefficients. In other words, the coefficients with most of the energy are in general the ones with low values of k in equation (38). Since, in this equation, k is proportional to the frequency of the cosine function, usually one refers to them as low frequency coefficients. In order to illustrate this, in Figure 21 it is shown the 256×256 DCT of the image LENA 256×256. Actually, the logarithm of the absolute value of the coefficients has been plotted. Black corresponds to zero, and the whither the pixel, the higher its value.

Figure 21: 256×256 DCT of the 256×256 LENA image.

We can see clearly that the energy is highly concentrated in the low frequency coefficients. Therefore, it is natural to think of a lossy image compression system in which only the coefficients having higher energies are encoded. In order to get a feeling of this, one can look at Figure 22. There, the LENA image has been divided into 8×8 blocks and the DCT of each block was computed. In Figures 22a to 22d we can see the image recovered, respectively, using 1, 3, 15 and all 64 coefficients of each block. We note, for example, that the difference between the original image and the one recovered using 15 coefficients is almost imperceptible.

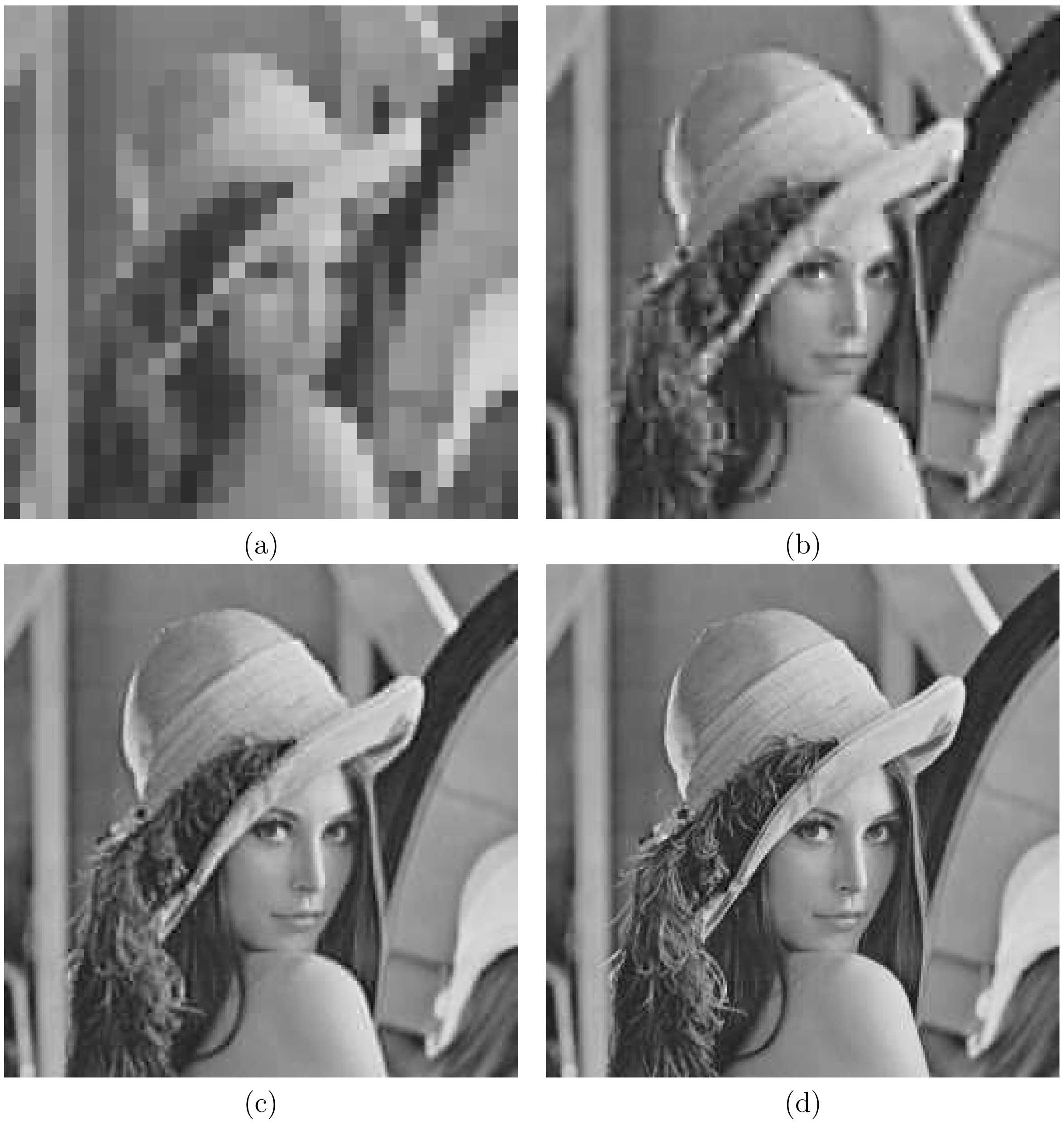


Figure 22: LENA image recovered using, from a 8×8 DCT: (a) 1 coefficient; (b) 3 coefficients; (c) 15 coefficients; (d) 64 coefficients.

Since the DCT highly concentrates the energy in few coefficients, we have that, after quantization, a large number of the DCT coefficients are zero. Therefore, a coding method that avoids explicitly encoding the zeros can be very efficient. In practice, this is achieved by scanning the quantized DCT coefficients as in Figure 23 prior to encoding. Then,

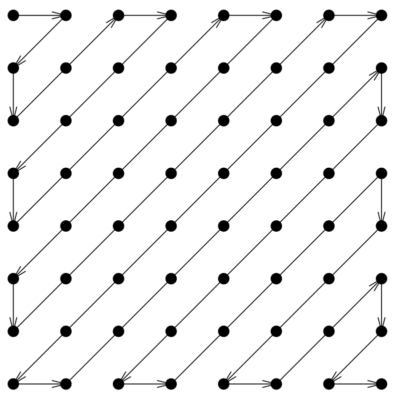


Figure 23: Scanning order of the quantized DCT coefficients.

instead of encoding each coefficient, one encodes the pairs (`run`,`level`); `level` indicates the quantized value of the non-zero coefficient, while `run` indicates the number of zeros that comes before them. The encoding is performed using an entropy coder, like Huffman or arithmetic coder. High compression rates can be achieved using such schemes. For example, the JPEG standard [27] is essentially based on the above scheme. The same applies to the H.261, H.263, MPEG1, MPEG2 and MPEG4 standards [12, 13, 14, 16, 17]. Figure 24 shows the LENA 256 $\times$ 256 image encoded using JPEG with 0.5 bits/pixel, a compression ratio of 16:1.

One of the main drawbacks of DCT-based image compression methods, is the blocking effect that occurs at low rates (see Figure 22). One way to solve this is to use a *wavelet transform* in the transformation stage.

Essentially, a wavelet transform is an octave band decomposition, in which each band is subsampled according to its bandwidth [23, 36]. A way of achieving this is to first divide a signal into low and highpass bands, with each band being subsampled by two. Then, only

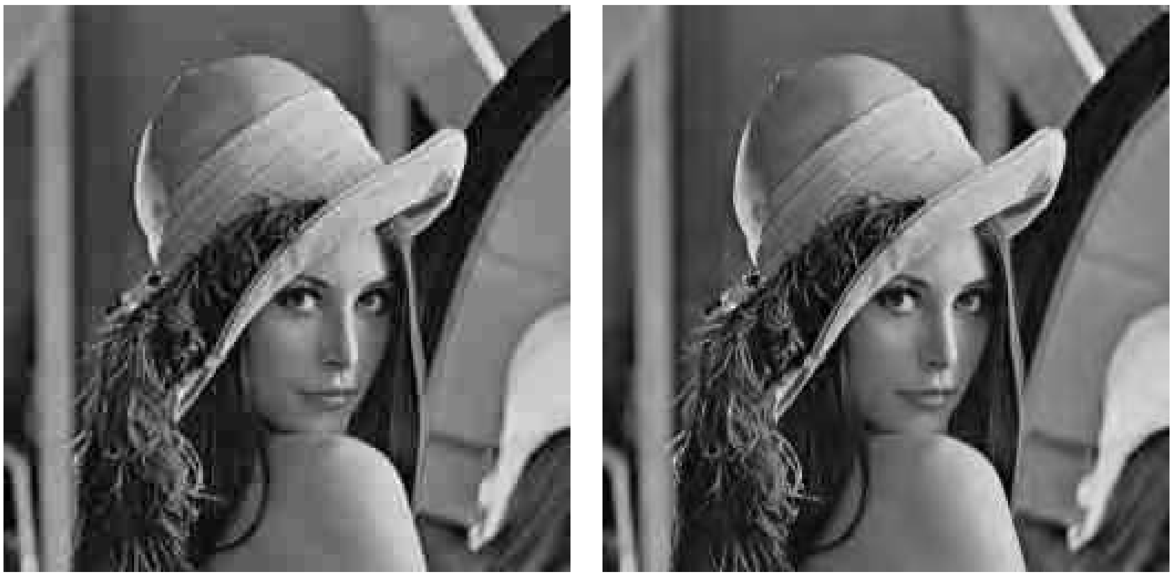


Figure 24: LENA 256 × 256 image with a compression ratio of 16:1: (a) using the JPEG standard; (b) using wavelet transforms.

the low pass channel is again low and high pass filtered and each band is subsampled by two. This process is recursively repeated until a predetermined number of stages is reached. For an image, the same process is performed for both its rows and columns. Figure 25 shows an image and its wavelet transform. As can be seen from Figure 25b, the wavelet transform contains mostly coefficients with small magnitudes; in addition, its bands are directional, that is, besides the low-frequency band, one can see horizontal, vertical and diagonal bands. From an image compression point of view, it is important to notice that the coefficients with small value, which are put to zero after quantization, tend to be clustered. This facilitates their efficient coding, thereby producing high compression ratios. Image coding methods based on the wavelet transform have the advantage of the absence of blocking effects in low rates. Figure 24b shows LENA 256 × 256 image compressed at a rate of 16:1 using a wavelet based encoder. On can notice the superior image quality compared

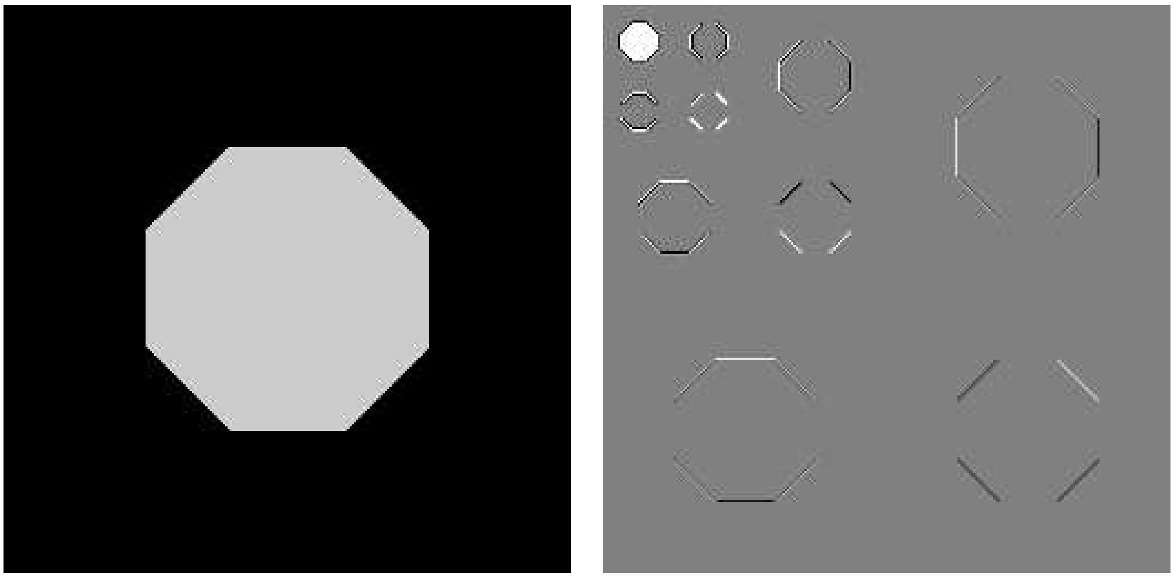

(a) (b)

Figure 25: (a) octagon image; (b) its wavelet transform (white corresponds to positive values, black to negative values and gray to zero values).

to the DCT-coded one in Figure 24a. Indeed, the JPEG2000 standard [15] is based on the wavelet transform. References to state-of-the-art wavelet image encoding methods can be found in [2, 20, 32, 34, 35].

6 Image Analysis

The discipline of image analysis is a very vast one. It deals with the automatic extraction of information from an image or sequence of images. It is also an essential part of computer vision systems. In this section we just provide a brief overview of its main issues. For a more detailed treatment, the reader is referred to some excellent texts on image processing and computer vision, as [5, 8, 9, 10, 11, 18, 22, 28, 30, 31].

In order for an image to be analyzed, first its main *features* have to be isolated. *Spatial* features, like edges, central moments and entropy, are used a great deal. However, *frequency* *domain* features, obtained in the Fourier transform domain may also be useful. Since, for analysis purposes, a scene must be segmented in objects, there must be ways of extracting the objects and representing them. Edge extraction systems are one of the first steps for isolating the different image objects. They are usually based on some form of gradient-based approach, where the edges are associated to large gray level variations. Once the edges are extracted, one can use contour following algorithms to identify the different objects. *Texture* is also a useful feature to be used.

Once the features are extracted, one uses image *segmentation* algorithms to separate the images into different regions. The simplest are the ones based on amplitude thresholding, in which the features used for region determination are the amplitudes of the pixels themselves; each region is defined by a range of amplitudes. If the features used are the edges, one can use *boundary representation* algorithms to define the regions in an image. One can also use *clustering* algorithms in order to define a region as the set of pixels having similar features. *Quad-trees* are a popular approach for defining regions. *Template* or *texture* matching algorithms can be used to determine which pixels represent a same shape or pattern, thereby defining regions based on them. These belong to the class of *pattern recognition* algorithms, that have, by themselves, a large number of applications. In the case of image sequences, *motion detection* can be used to define the regions as the pixels of the image having the same kind of movement. For each region, one can also perform *measurements*, such as *perimeter, area, Euler number*, etc. *Mathematical morphology* [33] is a discipline commonly employed for performing such measurements.

Having the different image regions defined, one must group them into objects, that is, one must label the regions so that all the regions belonging to an object have the same label, and regions belonging to different objects have different labels. In order to achieve this, one can use some kind of clustering algorithm.

Once the objects are identified they must be classified in different categories. The classification can be based, for example, on their shapes. Then, image understanding techniques can be used to identify high level relations among the different objects.

The image analysis tools mentioned above can be exemplified by Figure 26, the HARDWARE image. It shows one bolt and two nuts. In it, the features to be extracted are, for

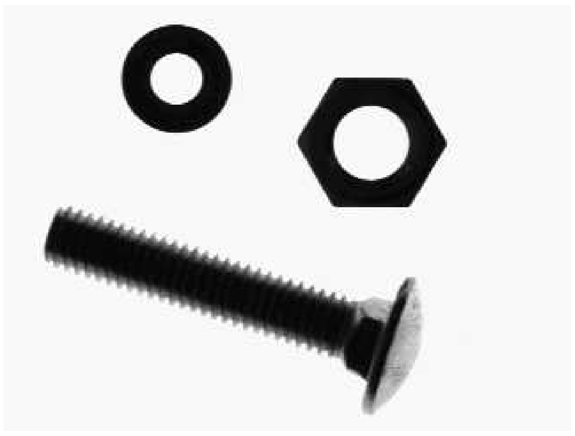


Figure 26: Hardware image.

example, the contours of the nuts and the bolt. The regions can be defined as the pixels inside the different contours; alternatively they can be represented as the pixels having gray level within predetermined ranges. Depending on the application, we may know the shape of one object (the bolt, for example), and perform pattern matching to determine the position of the bolt on the image. One could also use mathematical morphology to compute the Euler number to count the "number of holes" of each object, and consider the nuts as the ones having just one hole. Then, the inner perimeter of the nuts can be determined in order to check whether they comply to manufacturing specifications. By closely observing the bolt, one can notice that it is composed of several different regions (for example, its head clearly belongs to a region different from its body). Therefore, after the segmentation algorithm, one should perform region labelling in order to determine which regions should be united to compose the bolt. Once having established that the image is composed of three objects, image classification and understanding techniques should be used in order to figure out what kind of objects they are. In this case, one must determine that there are two nuts and one bolt from the data extracted in previous steps. Finally, a robot could use this information, extracted automatically, to screw the bolt into the round nut.

As mentioned previously, this section provides only a very brief overview of this vast subject, but the open literature provides a vast treatment of it [5, 8, 9, 10, 11, 18, 22, 28, 30, 31].

7 Summary

This article presented an introduction to image processing techniques. First, the issues of image sampling and quantization were dealt with. Then, image enhancement and restoration techniques were analyzed. Image compression methods were described, and the article was finished by a very brief description of image analysis techniques.

References

[1] N. Ahmed, T. Natarajan, and K. R. Rao. Discrete cosine transform. *IEEE Transactions on Computers*, C-23:90–93, January 1974.

[2] Jim Andrew. A simple and efficient hierarchical image coder. In *1997 IEEE International Conference on Image Processing*, Santa Barbara, CA, October 1997.

[3] Mark R. Banham and Aggelos K. Katsaggelos. Digital image restoration. *IEEE Signal Processing Magazine*, 14(2):24–41, March 1997.

[4] T. C. Bell, J. G. Cleary, and I. H. Witten. *Text Compression.* Prentice Hall, Englewood Cliffs, NJ, 1990.

[5] Kenneth R. Castleman. *Digital Image Processing.* Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1979.

[6] R. J. Clarke. *Transform Coding of Images.* Academic Press, London, 1985.

[7] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory.* Wiley, New York, NY, 1991.

[8] Olivier Faugeras. *Three-Dimensional Computer Vision - A Geometric Viewpoint.* The MIT Press, Cambridge, Massachusetts, 1993.

[9] Rafael C. Gonzalez and Paul Wintz. *Digital Image Processing.* Addison Wesley Publishing Company, London, 1977.

[10] Robert M. Haralick and Linda G. Shapiro. *Computer and Robot Vision*, volume 1. Addison Wesley Publishing Company, Reading, Massachusetts, 1992.

[11] Robert M. Haralick and Linda G. Shapiro. *Computer and Robot Vision*, volume 2. Addison Wesley Publishing Company, Reading, Massachusetts, 1993.

[12] INTERNATIONAL TELECOMUNICATION UNION. Video codec for low bitrate communication, May 1996.

[13] ISO/IEC JTC1/CD 11172. Coding of moving pictures and associated audio for digital storage media up to 1.5mbit/s, 1992.

[14] ISO/IEC JTC1/CD 13818. Generic coding of moving pictures and associated audio, 1994.

[15] ISO/IEC JTC1/SC29/WG1 (ITU/T SG28). JPEG2000 verification model 5.3, 1999.

[16] ISO/IEC JTC1/SC29/WG11. MPEG-4 video verification model version 8.0, July 1997.

[17] ITU-T. Recommendation H.261, Video codec for audio visual services at p$\times$64kbit/s, 1990.

[18] Anil K. Jain. *Fundamentals of Digital Image Processing.* Prentice Hall International, Englewood Cliffs, NJ, 1989.

[19] N. S. Jayant and P. Noll. *Digital Coding of Waveforms.* Prentice Hall, Englewood Cliffs, NJ, 1984.

[20] Tse-Hua Lan and Ahmed H. Tewfik. Multigrid embedding (MGE) image coding. In *Proceedings of the 1999 International Conference on Image Processing*, Kobe, 1999.

[21] Jae S. Lim. *Two-dimensional Signal and Image Processing.* Prentice Hall, Englewood Cliffs, New Jersey, 1990.

[22] Adrian Low. *Introductory Computer Vision and Image Processing.* McGraw-Hill Book Company, Berkshire, England, 1991.

[23] S. G. Mallat. *A Wavelet Tour of Signal Processing.* Academic Press, San Diego, CA, 1998.

[24] Russel M. Mersereau and Dan E. Dudgeon. *Multidimensional Digital Signal Processing.* Prentice-Hall Signal Processing Series. Prentice-Hall, Englewood Cliffs, NJ, 1984.

[25] A. V Oppenheim and Ronald. W. Schaffer. *Discrete-Time Signal Processing.* Prentice Hall, second edition, 1989.

[26] Athanasios Papoulis. *Probability, Random Variables and Stochastic Processes.* McGraw-Hill, second edition, 1984.

[27] William B. Pennebaker and Joan L. Mitchell. *JPEG Still Image Data Compression Standard.* Van Nostrand Reinhold, New York, 1993.

[28] W. K. Pratt. *Digital Image Processing.* Wiley, New York, 1978.

[29] Azriel Rosenfeld and Avinash C. Kak. *Digital Picture Processing*, volume 1. Academic Press, New York, second edition, 1982.

[30] Azriel Rosenfeld and Avinash C. Kak. *Digital Picture Processing*, volume 2. Academic Press, New York, second edition, 1982.

[31] John C. Russ. *The Image Processing Handbook.* CRC Press, Boca Raton, second edition, 1995.

[32] Amir Said and William A. Pearlman. A new, fast and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*, 6(3):243–250, June 1996.

[33] J. Serra. *Image Analysis and Mathematical Morphology.* Academic Press, 1982.

[34] Jerome M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 41(12):3445–3462, December 1993.

[35] David Taubman. High performance scalable image compression with EBCOT. In *1999 IEEE International Conference on Image Processing*, Kobe, October 1999.

[36] Martin Vetterli and Jelena Kovačević. *Wavelets and Subband Coding.* Prentice Hall PTR, Englewood Cliffs, New Jersey, 1995.

[37] Andrew B. Watson, editor. *Digital Images and Human Vision.* The MIT Press, Cambridge, MA, 1993.