MEDICAL IMAGE PROCESSING

Instructor : Prof. D. Cavouras
Electronic Engineer, B.Sc., M.Sc., Ph.D.
(cavouras@teiath.gr)

Table of contents

I. INTRODUCTION.

1.1. The Computerized Imaging System.

Figure (1) shows a block diagram of the basic components of a typical imaging system whose operation is controlled by a dedicated computer. The operation of such a system may be divided into four principal categories: acquisition, digitization, processing, and display. For the first operation, the data acquisition device may be the scintillation camera, the gantry of a CT or MR unit, the probe of the ultrasound unit, the image intensifier of the D.S.A., to mention the most important computerized imaging systems. These are the transducers that transform the various types of radiation to analogue electric currents prior to entering the host computer.

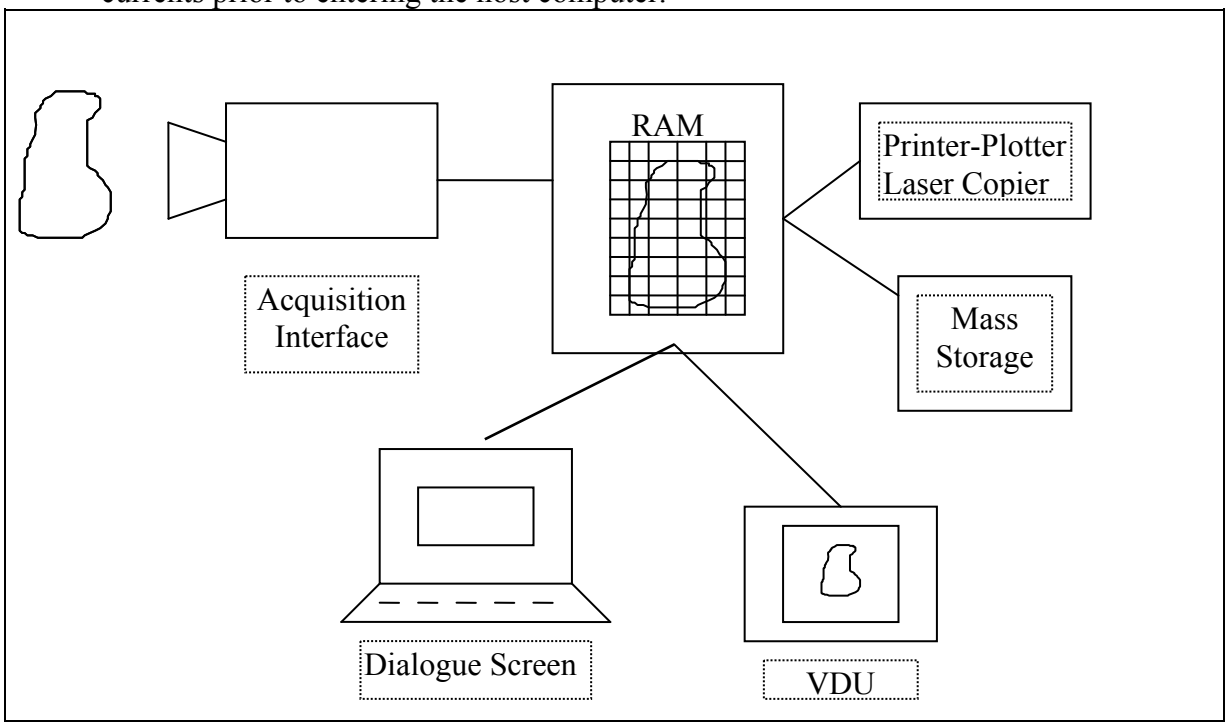


Figure 1

The second operation, the digitization process, in effect transforms the analogue signals into digital form, suitable for input to the computer system. This function is performed by the analogue to digital converters (ADC) and it is a part of acquisition interface which connects the acquisition device to the host computer. The design of both the acquisition device and the interface varies amongst the various imaging modalities, but they share a common purpose; that of transforming the oncoming radiation from (or through) the patient's body into a two dimensional digital signal, the so-called digital image, and storing it in the computer memory.

A major part of the digital image formation process plays the third operation, the data processing function, which is accomplished by the computer itself and the specialized hardware built to speed up processing time, such as array processors. The main parts of the computer system are shown in Figure (2).

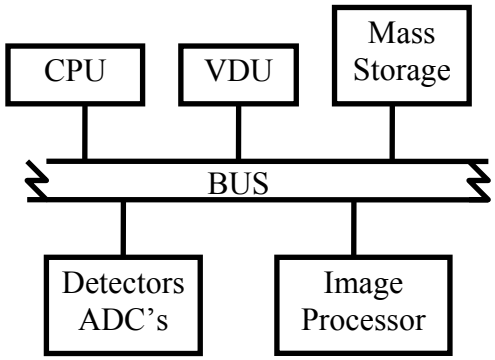


Figure 2

The result of the acquisition-digitization-processing functions is the digital image (used here in its discrete form $x(n_1,n_2)$ by analogy to the discrete signal x(n)), which in fact refers to a (discrete) matrix whose row n2 and column n1 indices identify a point in the image. The corresponding matrix element value $x(n_1,n_2)$ is the result of the interaction of the radiation with tissue and it is displayed on a monitor by a brightness level or grey level. That point is usually called pixel (picture element).

1.2. Image Display Process.

As shown in Figure (3) the digital image is stored in the RAM memory of the CPU as a two dimensional array x(n1,n2) or image matrix, whose dimensions may vary from 32x32 up to 1024x1024 depending on the modality used. The picture displayed on the monitor (see Figure(3)) consists of bright points, whose brightness is proportional to the value held in a corresponding position (byte or word) in the image matrix of the CPU, which, in turn, is proportional to the radiation intensity that strikes the imaging system's detector.

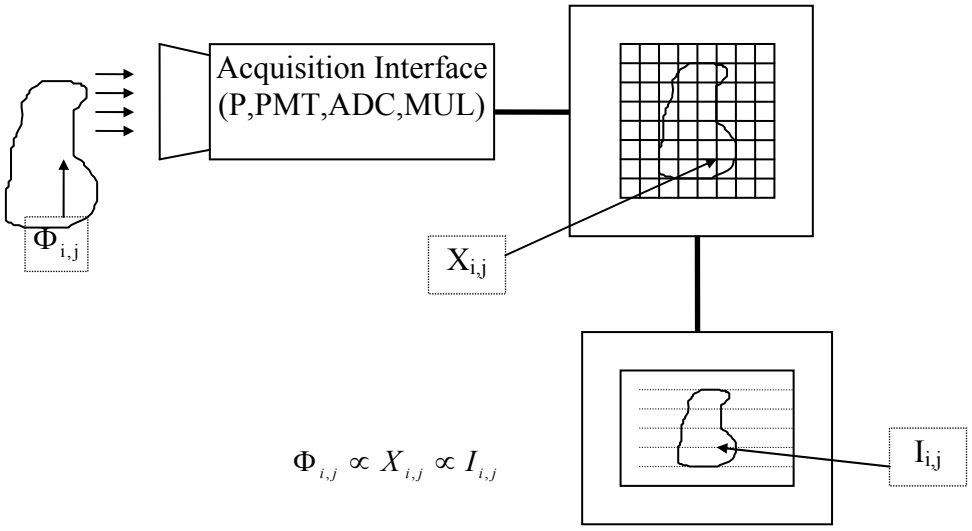


Figure (3)

The visual display unit (VDU). As shown in Figure (4) it consists of the controller and the display screen. **The VDU controller** consists of a RAM memory and a micro-processor unit. The RAM is used to store the picture to be displayed, so that the imaging system's CPU is turned over to other tasks, such as receiving new data etc.

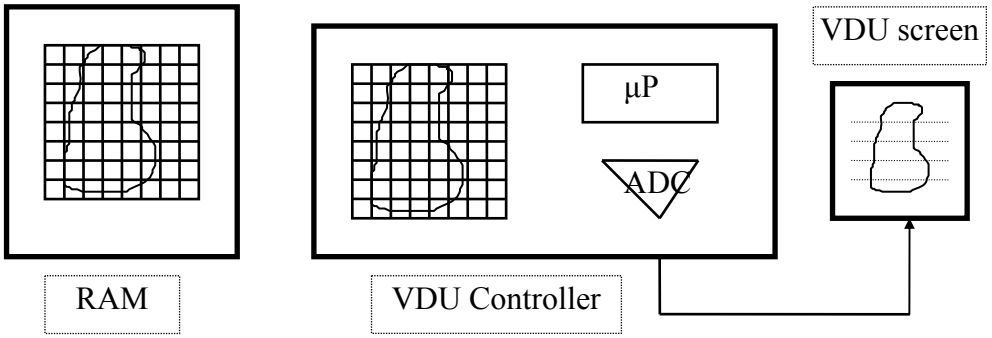


Figure 4

The microprocessor is dedicated to reading each matrix row and from the value stored in each row position (e.g. a byte) it calculates a dc voltage, which is then applied, through the DAC, across the electron gun of the VDU.
The VDU display screen. It consists of the electron gun which every 16 nsecs or less sweeps the whole of the VDU screen in parallel consecutive lines. At each instant position it shoots against the phosphor layer of the VDU screen an electron beam, whose electrons attain kinetic energies proportional to the numbers stored in the corresponding bytes of the image matrix. The effect is the emission for about 16 nsecs of visible light of brightness proportional to the kinetic energy of the electron beam . That bright spot is a picture or image element and it is the building block of the picture (e.g. 128x128 pixels constituting a picture). In this way, it is possible to display a picture stored in the RAM memory of the controller and to keep it there by (refreshing) sweeping the screen every 16 nsecs or less. **The pixel brightness levels** depend on the particular VDU in use, but for most medical imaging systems the brightness levels were, up to recently, restricted to 16 levels of brightness or grey levels that form the grey scale, usually displayed alongside most digital medical images. There are two major reasons for the limited number of available grey-levels on the screens of most medical systems. The first reason is the cost and the second is that it has been proved that the trained optical system of the radiologist can distinguish only 16 (for others 80) grey-levels in a picture. However, for most medical images (eg. CT images), the range of values (densities) in their image matrix is by far larger (about 3000 for CT images) than the available 16 screen greylevels. Any attempt to depict all image values on the screen at the same time will result in the loss of image contrast, giving a diagnostically poor picture. For this reason, image enhancement techniques called 'grey-scale modification techniques', described in chapter (2), are employed in order to display part of the range of image-values (quantization levels) at a time.

1.3. The Quality of the Medical Image.

The quality of the medical image is as good as is the clarity of the specific information sought for in the image by the observing physician. However, there is a need to define objective image quality features in order to assess the quality of the image produced by an imaging system or by a processing technique. The quality of the medical image depends on and is assessed by three parameters: sharpness, contrast, and noise.

Image sharpness, concerns mainly the ability to distinguish image detail. Image detail deterioration or blurring is mainly due to the imaging system's impulse response, the so called Point Spread Function (PSF). Image sharpness is assessed by the image spatial resolution. The latter is defined as the ability of the imaging system to distinguish (i.e. clearly display) two small high contrast dots close to each other. Quantitatively, the spatial resolution is determined by the smallest distance between two distinguishable high contrast dots or by the magnitude of the FWHM (Full Width at Half Maximum) or by the LSF (the number of distinguishable lines per cm) or by the MTF (Modulation Transfer Function).

Figure (5) gives the meaning of the PSF, while it shows the distance FWHM, which is the limit of the distance that the two point sources should differ in order to be distinguishable. Another way of quantifying Image Sharpness is the MTF, which in effect is the Fourier Transform of the PSF shown in Figure 6. It defines the cut-off frequency fco at the 5% of its maximum magnitude.

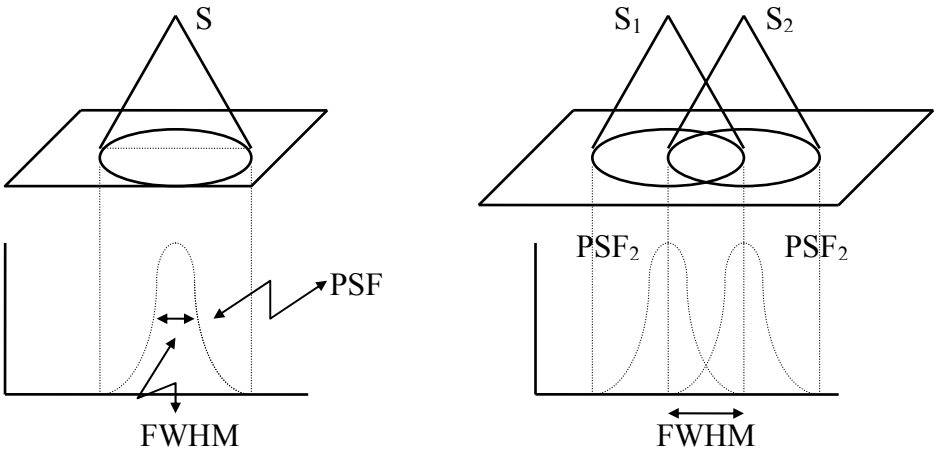


Figure 5

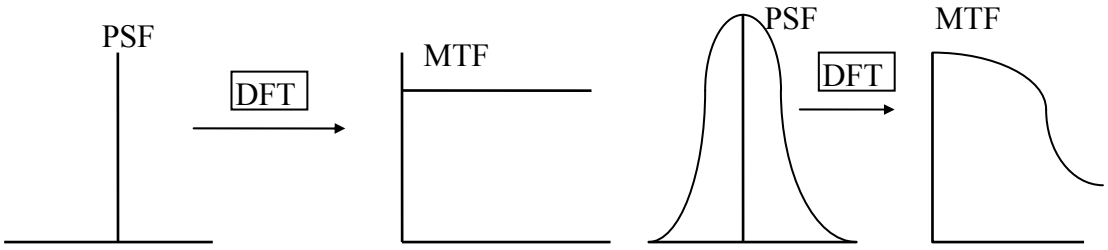


Figure 6

Image contrast, concerns the ability to distinguish image detail of low contrast with its surrounding background. In other words, it is the ability of the imaging system to sense small variations in radiation intensity at its detector end and to display that intensity variation. A source of image contrast degradation is the presence of noise. For the assessment of image contrast the term contrast resolution is used and it is defined as the smallest distinguishable intensity difference between a small image area (of specific shape and size) and its background. Image contrast can be quantified by equation (1)

$$\%\text{contrast} = \frac{I_{area} - I_{background}}{I_{background}} x100 \qquad (1)$$

Image noise, is of statistical nature and signal dependent but, without great loss of accuracy, it can be considered additive and white. The noise contribution to each pixel of the medical image is not known but an assessment of the overall noise contribution to the image can be obtained from:

i/ the standard deviation of the pixel intensity in an image area, where the signal is relatively constant. Equation (2) gives the formula used for calculating the standard deviation

$$SD = \left[\sum_{i=1}^{N} \frac{(x_i - \mu)^2}{N - 1} \right]^{\frac{1}{2}} \qquad (2)$$

where, N: number of pixels involved

μ :mean intensity value

x_i:pixel value

ii/ The noise power spectrum, which can be approximately assessed from the image high frequencies, where the signal to noise ratio is small. If the image noise is considered white, then across the spatial frequency spectrum of the image the noise power spectrum is constant.

Finally, the overall image quality may be approximately evaluated by equation (3)

$$\text{Image\_Quality} = K\frac{(\text{Sharpness})^2\,(\text{contrast})^2}{\text{Noise\_Power\_Spectrum}} \qquad (3)$$

Formula (3) can be used in the evaluation of various medical imaging systems using a suitable phantom.

1.4. The Discrete Fourier Transform for image processing.

Definition. The 1-d DFT is defined in (1)

$$Fx(k)= \frac{1}{N}\sum_{n=0}^{N-1} x(n)e^{-j\frac{2\pi}{N}kn} \qquad (1)$$

where, x(n) is the discrete periodic signal in the time/spatial domain, N is the period of x(n) or the length of the sampled signal, and n,k are the time/space and frequency variables.

The Inverse DFT or IDFT is defined in (2)

$$x(n)= \sum_{k=0}^{N-1} Fx(k)e^{j\frac{2\pi}{N}kn} \qquad (2)$$

The discrete Fourier pair defined in (1) and (2) may also be written as in (3)

$$Fx(k)= \sum_{n=0}^{N-1} x(n)W^{-kn}$$

(3a)

$$x(n)= \frac{1}{N}\sum_{k=0}^{N-1} Fx(k)W^{kn} \qquad (3b)$$

where $W = e^{j\frac{2\pi}{N}} = \cos\left(\frac{2\pi}{N}\right) + j\sin\left(\frac{2\pi}{N}\right)$ from Euler's relations.

Since Fx(k) is a complex, then we have

$$Fx(k)= \left|Fx(k)\right|e^{j\varphi x(k)} = Rx(k)+ jIx(k) \qquad (4)$$

where |

$$Fx(k)= \left(\left[Rx(k)\right]^2 + \left[Ix(k)\right]^2\right)^{\frac{1}{2}} \quad \text{... amplitude}$$

$$\phi x(k)= \tan^{-1}\frac{Ix(k)}{Rx(k)} \quad \text{... phase}$$

$$Px(k)= \left|Fx(k)\right|^2 \text{ ... power}$$

relations (4) are used to generate the amplitude, phase, and power (frequency) spectra of the discrete signal x(n).

The 2-dimensional DFT. For a N1xN2 image matrix, the 2-DFT pair is defined in (5), by analogy to (1) and (2):

$$Fx(k_1,k_2)=\frac{1}{N_1}\frac{1}{N_2}\sum_{n_2=0}^{N_2-1}\left[\sum_{n_1=0}^{N_1-1}x(n_1,n_2)W^{-k_1n_1}\right]W^{-k_2n_2} \qquad (5a)$$

$$x(n_1,n_2)=\sum_{k_2=0}^{N_2-1}\left[\sum_{k_1=0}^{N_1-1}Fx(k_1,k_2)W^{k_1n_1}\right]W^{k_2n_2} \qquad (5b)$$

From (5a) we get

$$Fx(k_1,k_2)=\frac{1}{N_2}\sum_{n_2=0}^{N_2-1}\left[\frac{1}{N_1}\sum_{n_1=0}^{N_1-1}x(n_1,n_2)W^{-k_1n_1}\right]W^{-k_2n_2}$$

or

$$Fx(k_1,k_2)=\frac{1}{N_2}\sum_{n_2=0}^{N_2-1}\left[Fx'(k_1n_2)\right]W^{-k_2n_2} \qquad (6a)$$

where

$$Fx'(k_1,n_2)=\frac{1}{N_1}\sum_{n_1=0}^{N_1-1}x(n_1,n_2)W^{-k_1n_1} \qquad (6b)$$

In (6b) n_2 is considered constant and n1 is variable. It is obvious that (6b) is the 1-dimensional DFT of image column n2. It is, therefore, concluded that the first step to calculate the 2-dimensional DFT of an image N_1xN_2 is to form a new N_1xN_2 complex matrix of the 1-dimensional DFT of the columns of the image matrix. In relation (6a) k_1 is considered constant and n_2 is the variable. Also, (6a) is the 1-d DFT of the complex 2-d signal $Fx'(k_1,n_2)$, which is the row k_1 of the new complex image-matrix form, found in the previous step by means of relation (6b). Thus, the second and final step of calculating the 2-d DFT of an image matrix N1xN2 is to take the 1-d DFT of the rows of the complex image matrix found in step 1 and store the results in the final complex matrix N_1xN_2. A similar procedure is followed for the computation of the 2-d IDFT. The Fourier Transform of an image $x(n_1,n_2)$ is usually computed using the Fast Fourier Transform (FFT).

1.4.1. Properties of the DFT

Shift theorem

1-d case :

if 1-d DFT[x(n)] = Fx(k) then 1-d DFT[x(n-h)] = W^{-kh}Fx(k) (1)

also

if 1-d IDFT[Fx(k)] = x(n) then 1-d IDFT[Fx(k-h)] = W^{kh}x(n) (2)

2-d case :

if 2-d DFT[$x(n_1,n_2]$ = $Fx(k_1,k_2)$ then

$$2d\_DFT[x(n_1-h_1,n_2-h_2)] = W^{-(k_1h_1+k_2h_2)}\, Fx(k_1,k_2) \quad (3)$$

also

if 2d_ IDFT[$Fx(k_1,k_2)$] = $x(n_1,n_2)$ then

$$2d\_IDFT[Fx(k_1-h_1,k_2-h_2)] = W^{(n_1h_1+n_2h_2)}\, x(n_1,n_2) \quad (4)$$

Now, if $n_1=n_2=N/2$

$W^{(n1h1+n2h2)}\, x(n1,n2)=...=(-1)^{n1+n2}x(n1,n2) =$

$$= 2-d \;\; IDFT\left[Fx\left(k_1 - \frac{N}{2}, k_2 - \frac{N}{2} \right) \right] \quad (5)$$

Thus, if we multiply the image matrix $x(n_1,n_2)$ by the factor $(-1)^{n_1+n_2}$ we can shift the origin in the frequency domain to the point (N/2,N/2). This is often useful when we attempt to display the amplitude or power spectrum of an image.

Periodicity:

1-d case : In the definition of the DFT (relation 1) the discrete signal x(n) was considered periodic with period equal to the length of the signal, i.e. x(n)=x(n+N), where n:0,1,2,...,N-1 and N is the period. The DFT of x(n), that is the Fx(k) where k:0,1,2,...,N-1, is a periodic complex signal with period N, equal to the period N of the signal x(n). Also, one period of the signal x(n) or of the complex signal Fx(k) is enough to completely specify the DFT and the IDFT respectively.

2-d case : Similar considerations hold for the 2-d case, that is:

$$Fx(k_1,k_2)=Fx(k_1+N,k_2)=Fx(k_1,k_2+N)=Fx(k_1+N,k_2+N) \quad (6)$$

Complex Conjugate Theorem.

1-d case : For the DFT the complex conjugate theorem states:

$$Fx\left(\frac{N}{2} + L \right) = F_x^*\left(\frac{N}{2} - L \right) \quad (7)$$

where, L=0,1,2,..., $\frac{N}{2}$ -1 and $F_x^*\left(\frac{N}{2} - L \right)$ is the complex conjugate. In other words, we only need to calculate half the DFT of a signal x(n), since the other half is the complex conjugate of the first N/2 values of the Fx(k). Taking under consideration the fact that the magnitudes of the Fx(k) and its complex conjugate Fx* (k) equal to each other

$$\left|Fx\left(\frac{N}{2}+L\right)\right| = \left|F_x^*\left(\frac{N}{2}-L\right)\right| \qquad (8)$$

we can then demonstrate the symmetric property of the Fx(k) given in (8). Usually, the way to demonstrate the Frequency Spectrum of a signal is to shift it by N/2 in the frequency domain, i.e. taking the DFT of $(-1)^n x(n)$ as explained in the shift theorem paragraph.

2-d case: Similar to 1-d case, we get the 2-d complex conjugate theorem:

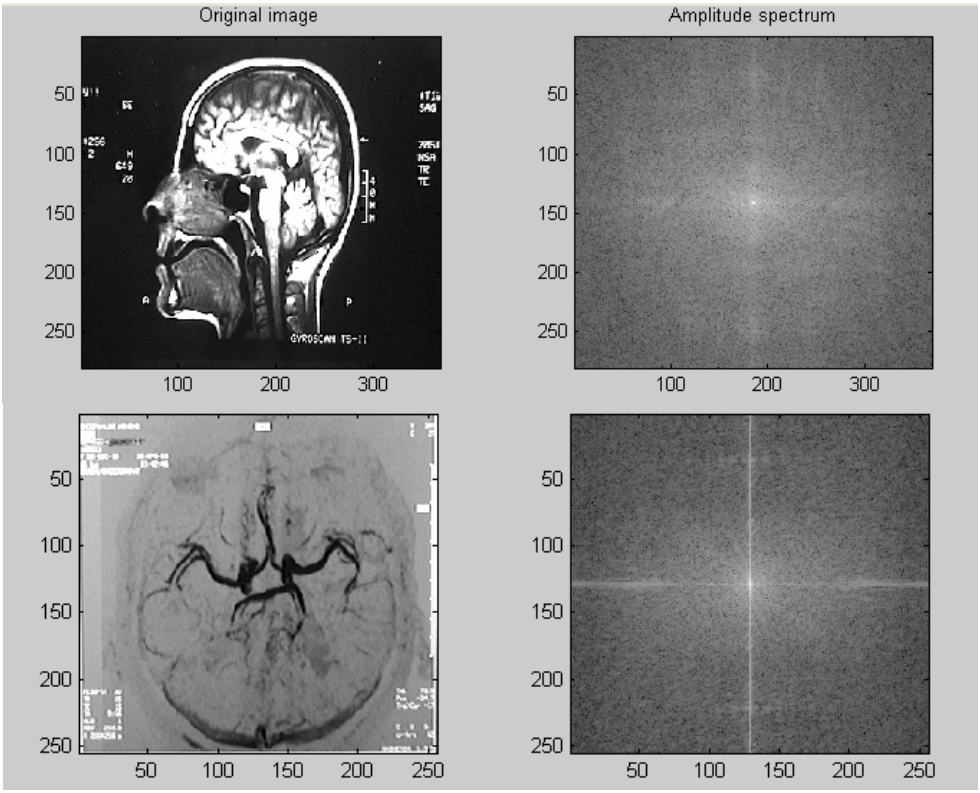
$$Fx\left(\frac{N}{2}+L_1,\frac{N}{2}+L_2\right) = F_x^*\left(\frac{N}{2}-L_1,\frac{N}{2}-L_2\right) \qquad (9)$$

where, $L_1,L_2=0,1,2,\ldots,\frac{N}{2}-1$

Also, for displaying the Fourier Spectrum of a picture we use relation (10):

$$\left|Fx(k1,k2)\right| = \left|2d-DFT\left[(-1)^{n1+n2}x(n1,n2)\right]\right| \qquad (10)$$

Figure (7) shows the picture and its Fourier Spectrum displaced at (N/2,N/2).



Σχήμα 7

Convolution/Correlation Theorem.

1-d case : The discrete **convolution** theorem between two periodic signals $x_1(n)$ and $x_2(n)$ is defined as in (11)

$$z(n)= x_1(n)* x_2(n)= \frac{1}{N}\sum_{m=0}^{N-1} x_1(m)x_2(m-n) \qquad (11)$$

where '*' denotes the convolution operation.

Similarly the discrete **correlation** between two periodic signals is defined as

$$z'(n)= x_1(n)\otimes x_2(n)= \frac{1}{N}\sum_{m=0}^{N-1} x_1(m)x_2(m+n) \qquad (12)$$

where '$\otimes$' denotes the correlation operation.

It can be proved that taking the DFT of 11 and 12 then

$$DFT[z(n)] = DFT[x_1(n)*x_2(n)] = Fx_1(k)Fx_2(k) = Fz(k) \qquad (13)$$

$$DFT[z'(n)] = DFT[x_1(n)\otimes x_2(n)] = Fx_1^*(k)Fx_2(k) = Fz'(k) \qquad (14)$$

where, the term $Fx_1^*(k)$ denotes the complex conjugate of Fx1(k). Equations (13) and (14) indicate that we can obtain the convolution/correlation of two signals by taking the inverse DFT of their product in the frequency domain (in the case of the correlation the complex conjugate of the DFT of the first signal is used). Equations (13) and (14) are referred to as the convolution/correlation theorems.

Wrap-around error. It is an error associated with the convolution of two discrete signals $x_1(n)$ and $x_2(n)$ of length N (period). Their convolution is also a periodic signal of length N. It has been shown that unless we expand with zeroes the signals to a length of M>=2N-1, the periods of the convolution will overlap giving rise to false components in the overlapping areas. However, in the case of medical images this error is not serious and it can be overlooked with a good approximation, if we take under consideration the increase in the number of calculations associated with doubling the size of the image matrices.

2-d case : Analogous to equations (13) and (14) are the 2-d versions of the convolution/correlation theorems:

$$z(n_1,n_2)= x_1(n_1,n_2)* x_2(n_1,n_2)= \frac{1}{N}\frac{1}{N}\sum_{m=0}^{N-1}\sum_{l=0}^{N-1} x_1(m,l)x_2(m-n_1,l-n_2) \qquad (15a)$$

$$z(n_1,n_2)=2d\_IDFT\left[Fx_1(k_1,k_2)Fx_2(k_1,k_2)\right] \qquad (15b).$$

$$z'(n_1,n_2)= x_1(n_1,n_2)\otimes x_2(n_1,n_2)= \frac{1}{N}\frac{1}{N}\sum_{m=0}^{N-1}\sum_{l=0}^{N-1} x_1(m,l)x_2(m+n_1,l+n_2) \qquad (16a)$$

$$z'(n_1,n_2)=2d\_IDFT\left[Fx_1^*(k_1,k_2)Fx_2(k_1,k_2)\right] \qquad (16b).$$

II. IMAGE ENHANCEMENT.

The purpose of the image enhancement techniques is to suitably modify the image so that the resulting image will be of superior quality to the eye of the observing physician. There are several image enhancement techniques and their employment depends on the specific application problem and the observer himself. They may be distinguished into 2 major categories: i/ Grey scale manipulation for increasing image contrast and ii/ Image matrix manipulation for decreasing image noise and blur. Furthermore, image enhancement by image matrix modification can be achieved in either the spatial or the frequency domain, since the convolution process can be carried out in either domain. For this reason, in this text, image enhancement is examined in both domains.

First, we examine the grey scale manipulation techniques.

2.1. Grey Scale Manipulation Techniques.

These are image enhancement techniques which, although simple and easy to implement, they provide impressive results. They attempt to deal with the problem that in most medical images the range of values of the image matrix (quantization levels) is by far larger than the available range of intensity or grey levels. Any attempt to display the whole of the image matrix range will result in the loss of image contrast. There are two types of grey scale manipulation techniques: a/the windowing techniques and b/ the histogram modification techniques, both widely used in computerized Medical Imaging Systems.

2.1.1. Windowing Techniques.

In the windowing techniques only part of the range of the image matrix values is displayed with available greyscale at a time. The term "window" refers to the section of the image value range that is displayed, which of cource can "slide" across the whole image value range, displaying different parts at a time. The "windowing" principle is shown in Figure (1).

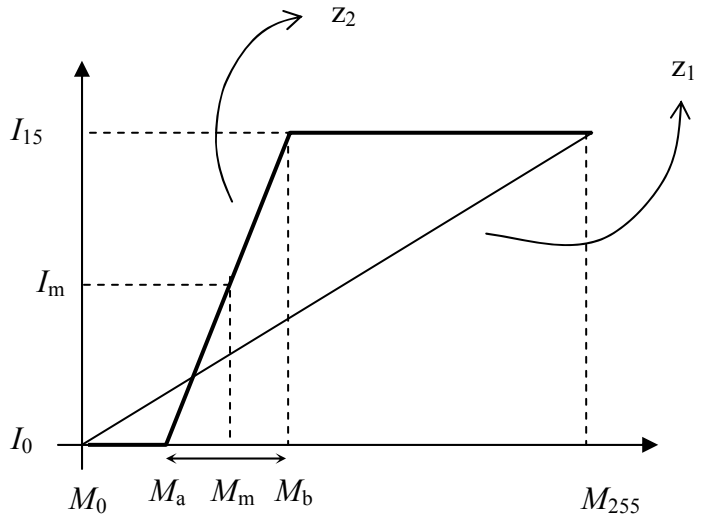


Figure 1(a)

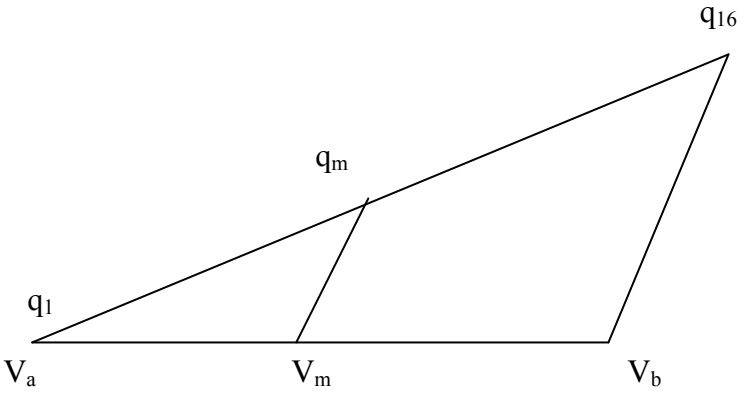


Figure 1(b)

Let the image (quantization) values be 256 (2^8) from V_1 to V_{256} and let the VDU grey levels be 16 (2^4) from g_1 to g_{16}, represented by the horizontal and vertical axes of Figure (1) respectively. One way to display the image (z_1 display method) is to assign the whole range of 256 values to the 16 greylevels. By this method, each one of the 16 grey-levels represents 16 consecutive (256/16) image values on the screen providing a low contrast image. The other way (z_2 displaying method) is to choose a range of values $[V_a,V_b]$, such as 64, and display that range using 16 greylevels. In the latter case, there will be 4 consecutive (instead of 16) image values assigned to each greylevel, thus part of the image is displayed highly contrasted.

The algorithm that calculates the greylevel (q_m) that each value (V_m) of the selected sub-range $[V_a,V_b]$ will be assigned to, is given below. From the similar triangles in Figure (1b) it is easily deducted that:

$$\frac{q_{16}-q_1}{V_b-V_a}=\frac{q_m-q_1}{V_m-V_a} \qquad \text{or}$$

$$q_m=\frac{q_{16}-q_1}{V_b-V_a}\left(V_m-V_a\right)+q_1 \qquad \text{for } V_a \le V_m \le V_b$$

$$\text{or } = q_1 \qquad \text{for } V_m<V_a \qquad (1)$$

$$\text{or } = q_{16} \qquad \text{for } V_m>V_b$$

Remarks:

1/$[V_a,V_b]$ is called window and V_b-V_a is the window width, usually chosen by the user interactively as the first parameter for displaying a medical image.

2/ The mean value $(V_b+V_a)/2$ or window level or level is the second parameter chosen by the user interactively and it determines the position (value) of the image-value sub-range at the left and right of which the window will be equally stretched.

Figure (2) shows the same image displayed at a window level.

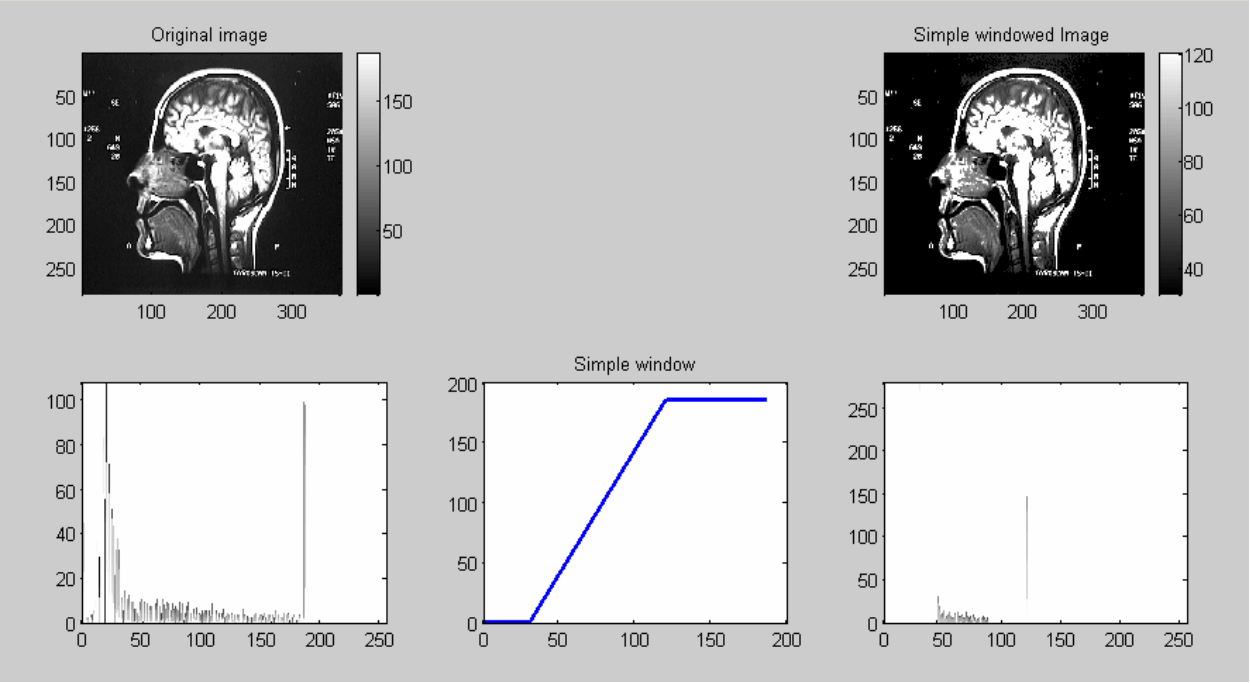


Figure 2.

3/Following similar reasoning, in some cases we may have two windows at different levels as shown in Figure (3a). This is particular useful if the system's VDU has a large number of greylevels (eg. 64 or 256). In that case, various parts of an image, such as the lungs and the spine of a CT slice, can be displayed at the same time and with a good contrast.

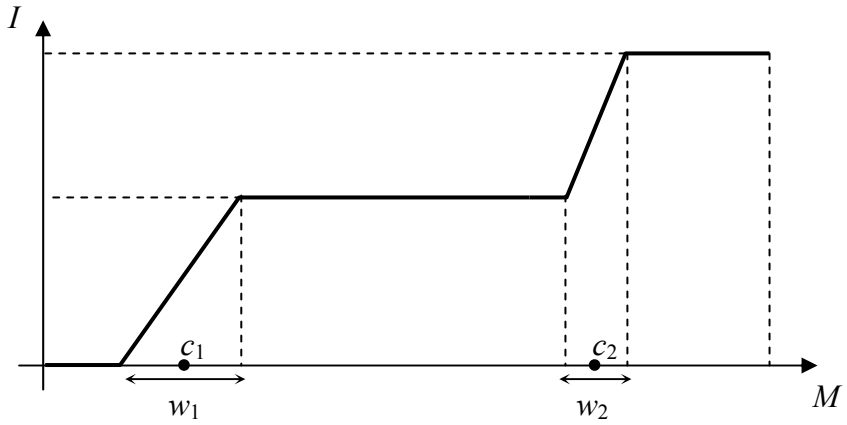


Figure 3(a): Double window

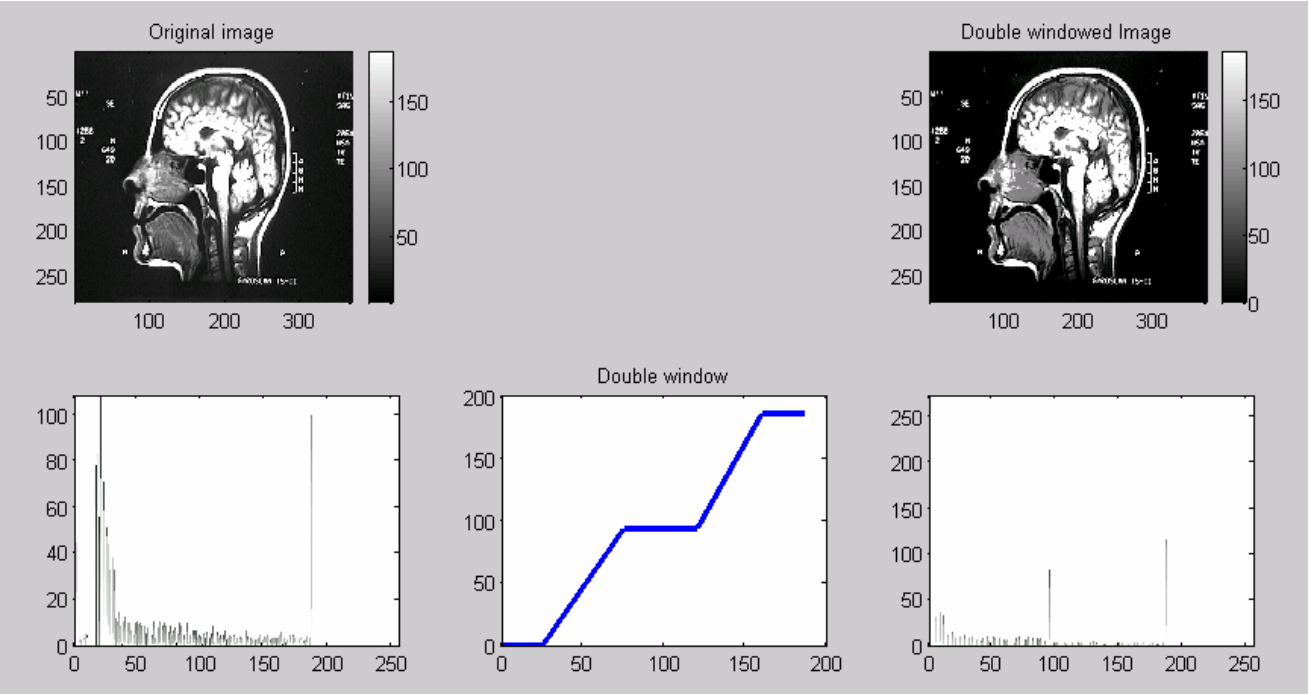


Figure 3(b): Double window and histograms.

4/ The way image values may be assigned to greylevels does not always have to be linear but it may follow other patterns as in Figure (3c).

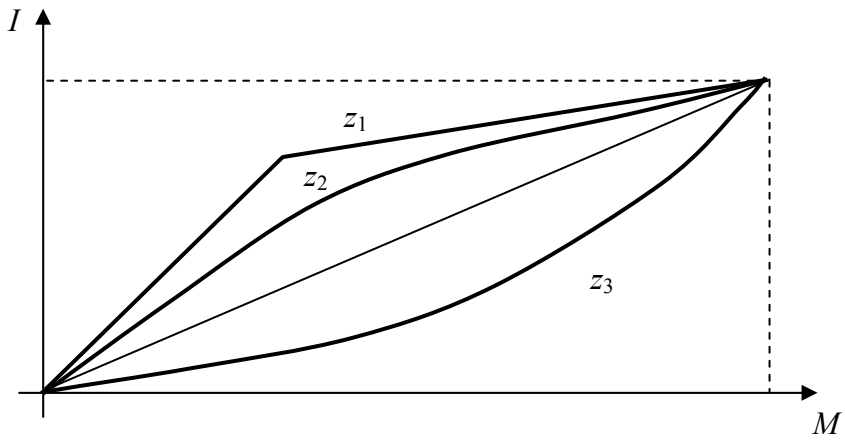


Figure 3c: Non-linear functions of windowing.

2.1.2. Histogram Modification Techniques.

The histogram is a graph describing the number of pixels assigned to each greylevel of a digital image. Each displayed medical image has a unique histogram and the histogram modification techniques attempt to enhance image contrast by altering the image histogram. The new histogram may be that of a known picture or a histogram with equal number of pixels per greylevel. The latter case, which is the most popular in medical systems, is called the histogram equalization procedure or technique.

Histogram equalization Algorithm

Let h_i be the histogram of an image, digitized to 256 levels, as shown in Table 1, where we have assumed that the VDU screen can provide 8 distinct greylevels and that the image under consideration is chosen, for the sake of simplicity, to be part of a larger image matrix. Let n_i be the new equalized image histogram, where there is an equal number of pixels assigned to each greylevel.

Table 1: Histogram equalization example.

	00	32	64	96	128	160	192	225 : Look up table
	31	63	95	127	159	191	224	256 : of diplay routine.
i:	0	1	2	3	4	5	6	7 : Greylevels
h_i:	2	4	12	14	14	12	4	2 : Original Histogram
n_i:	8	8	8	8	8	8	8	8 : Equalization Hist.

Algorithm description: As a general principal we form n_i by borrowing pixels from h_i as follows:

1/ $n_0=h_0+h_1+2h_2$, where $2h_2$ denotes that we choose 2 pixels from the twelve pixels of h_2 by one of the following principles:

a/ we choose the pixels whose corresponding matrix value is nearest to the border value 63 of h_1 (see Table 1).

b/ we choose 2 pixels at random.

All chosen pixels of h_0, h_1, and $2h_2$ are now assigned to greylevel 0, thus, in effect those image pixels are now squashed or compressed under the same greylevel.

2. $n_1=8h_2$, that is 8 of the remaining 10 pixels of h_2 are used to form n_1 and the last 2 pixels will be used for the formation of n_2.

3/ $n_2=2h_2+6h_3$ and so on...

We see that the 12 points of h_2 have been spread or stretched amongst 3 greylevels of the equalized histogram, i.e. n_0, n_1, n_2. This means that we can now distinguish those points on the screen, i.e. the image contrast has been improved. Figure (4) shows the result of applying histogram equalization techniques.

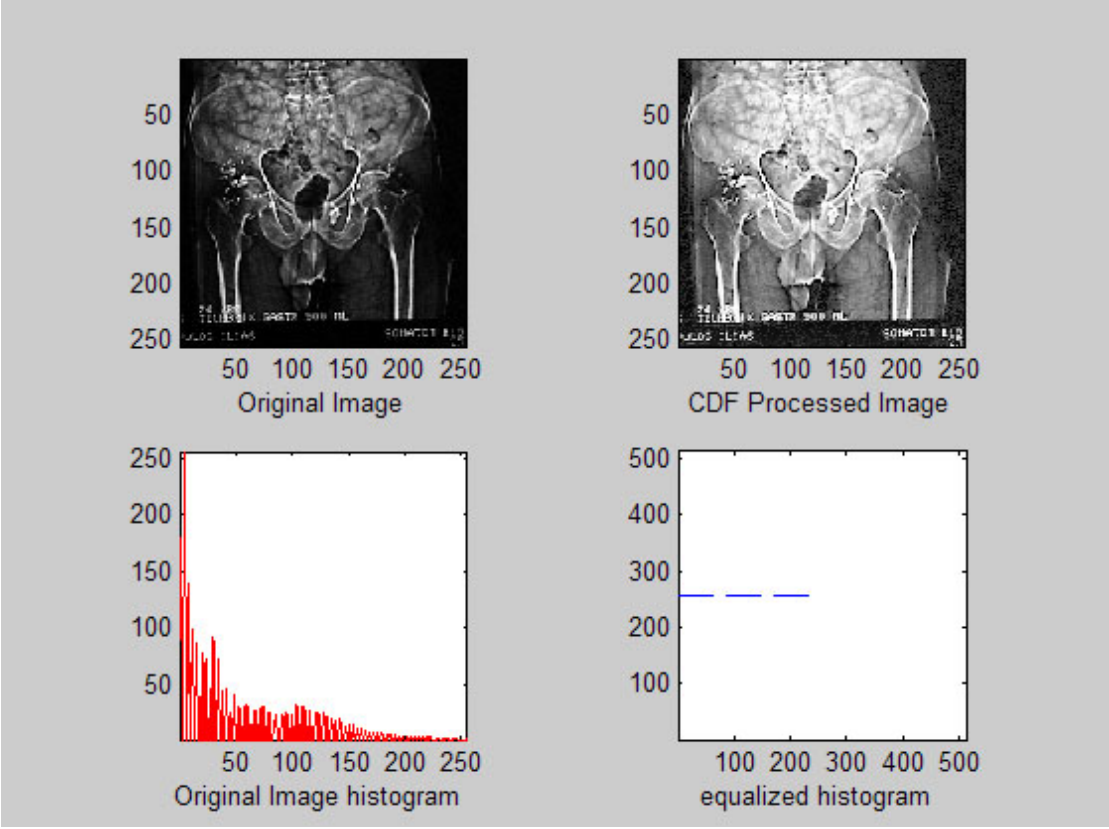


Figure 4

CDF-based histogram modification

CDF: Cumulative Distribution Function

$$\sum_{k=0}^{i} h(k) = \sum_{k=0}^{i} g(k), i = 0,2,3,..,7$$

Grey levels	h(i)	CDFh(i)	q(i)	CDFq(i)	
0	1(0)	1	16	16	Σ
1	7(0)	8	16	32	Σ
2	21(1)	29	16	48	Δ
3	35(3)	64	16	64	Δ
4	35(5)	99	16	80	Δ
5	21(6)	120	16	96	Δ
6	7(7)	127	16	112	Σ
7	1(7)	128	16	128	Σ

$t = (int)\left(\frac{CDFh(i)}{16} - 1\right)$ for approximate but fast calculation. Result is shown in Fig 5.

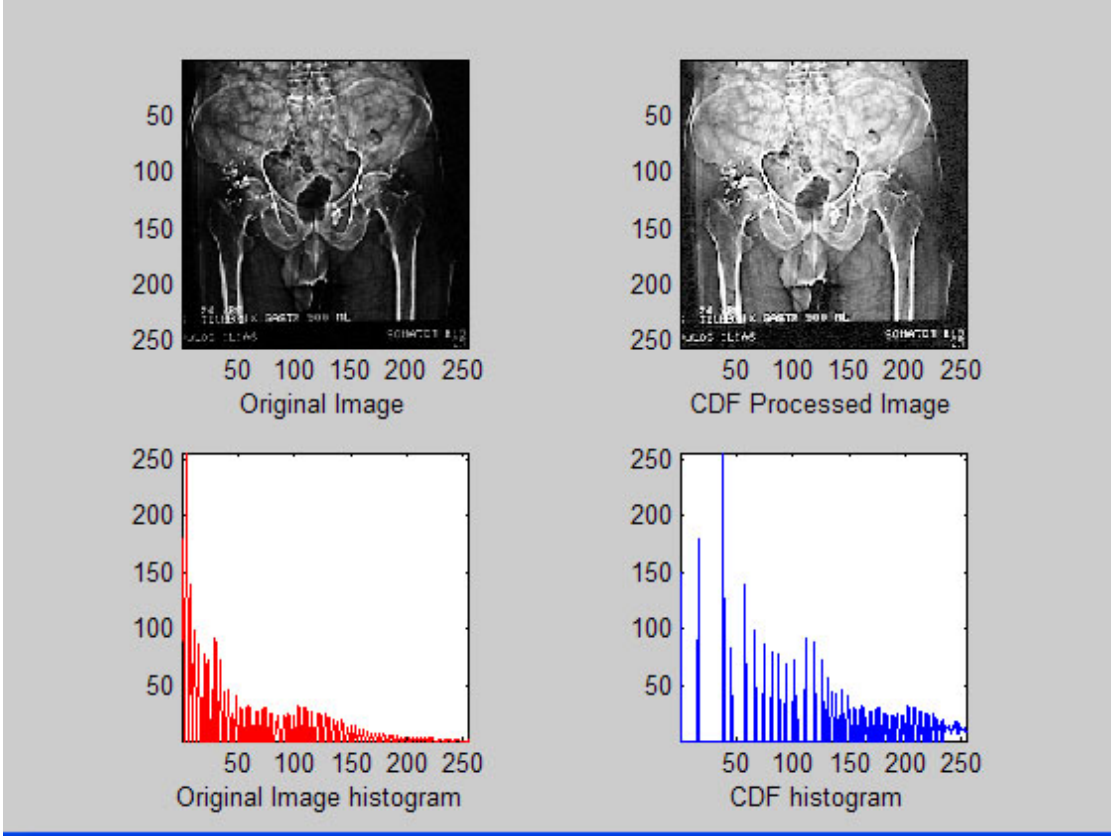


Figure 5

2.2. Time Domain Image Matrix Manipulation Techniques.

These techniques attempt to reduce one of the sources of image degradation, described by the general image degradation model and given by equation (1).

$$x^*(n_1,n_2) = x(n_1,n_2) * h(n_1,n_2) + d(n_1,n_2) \quad (1)$$

1/ The convolution term $x(n_1,n_2)*h(n_1,n_2)$ usually suppresses the medium to high frequency image content with result the blurring of the image boundaries and edges and considerable loss of image detail ($h(n_1,n_2)$ is the same as the Point Spread Function or PSF of the system).

2/The noise degradation $d(n_1,n_2)$ is of statistical nature and to a good approximation it is considered as additive and signal independent (although signal dependency may sometimes be severe). Noise resides mostly in the high frequency spectrum of the medical image and when its magnitude is larger than the difference in intensity of two neighboring areas (e.g. metastatic area against normal tissue in liver) then distinguishing those two areas is difficult.

2.2.1. Image Smoothing.

Image smoothing concerns image processing techniques used for suppressing noise, which for various reasons (e.g. sampling, transmission, statistical nature of radiation) resides in the image. Usually, medical image noise is statistical and occupies mainly the image high frequency band. Thus, techniques for image smoothing are in effect high frequencies suppression techniques or low pass filtering techniques. The obvious repercussion is the suppression of useful high frequency image information, resulting in image boundary blurring and image detail degradation. Thus, caution should be exercised on the amount of image smoothing.

Local Averaging.

1-dimensional case…: Figure (1) presents the signal degradation process for signal x(n) through system h(n) and with additive noise d(n). Output signal y(n) is described by equation (1):

$$y(n)=x(n) * h(n)+d(n)=x'(n)+d(n) \quad (1)$$

where $x'(n)=x(n)*h(n)$

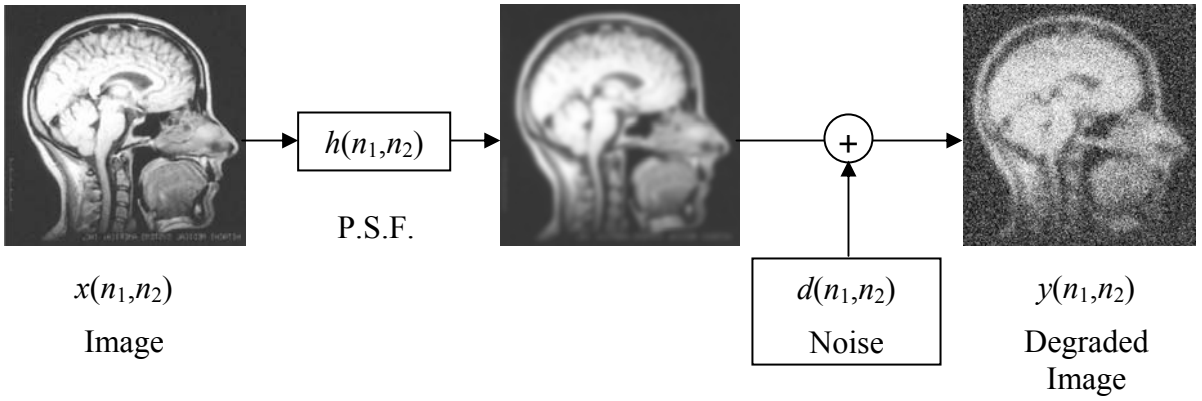


Figure 1

Let hf(n) be the digital filter impulse response then the degraded signal y(n) is filtered as shown below.

If $h_f(n)=\frac{1}{N}[1,1,1,...,1]$ (N terms) then the filtered output y'(n) is given by:

$y'(n)=y(n) * h_f(n)=[x'(n)+d(n)] * h_f(n)=$ (2)

$$= \sum_{k=0}^{N-1}[x'(n-k)+d(n-k)]h_f(k) \quad (3)$$

Now for N=2, $h_f(n)=\frac{1}{2}[1,1]$ and from (3)

$$y'(n)=\frac{1}{2}(x'(n)+x'(n-1))+\frac{1}{2}(d(n)+d(n-1)) \quad (4)$$

Similarly for N=M

$$y'(n)=\frac{1}{M}(x'(n)+x'(n-1)+...+x'(n-M))+\frac{1}{2}(d(n)+d(n-1)+...+d(n-M)) \quad (5)$$

Now, if noise is considered white then we would expect its mean value to tend to zero, thus the larger the M in (5) the closer to zero is the second term of (5). Therefore, when we want to filter out the noise from signal x(n) we replace each signal point by the average of M neighboring points. Caution should be taken as to the number of M points used, since a large M would smooth severely the image, suppressing the high frequencies signal content. Usually, equation (5) is used as in (6)

$$y'(n)=\frac{1}{M}\sum_{k=-\frac{M-1}{2}}^{k=\frac{M-1}{2}}x(k) \quad (6)$$

where M is odd, i.e. M=3,5,7 etc.

Sometimes the filter is called moving average since it is applied locally by moving across the signal.

Frequency response of 2-point averaging.

It is interesting to examine the frequency response of the local averaging filter to prove that it is a low pass filter. For this, we examine the simplest 2-point averaging described by (7):

$$y(n)= \frac{1}{2}\big(x(n)+ x(n-1)\big) + \frac{1}{2}\big(d(n)+ d(n-1)\big) \tag{7}$$

where n:0,1,2,...,N-1.

if the noise term in (7) is considered negligible then

$$y(n)= \frac{1}{2}\big(x(n)+ x(n-1)\big) \tag{8}$$

Taking the DFT of (8)

$$Fy(k)= DFT\left(\frac{x(n)}{2}\right) + DFT\left(\frac{x(n-1)}{2}\right) = \frac{Fx(k)}{2} + W^{-k}\,\frac{Fx(k)}{2} =$$

$$= \frac{1}{2}\left(1+ e^{-j\left(\frac{2\pi}{N}\right)k}\right) Fx(k)$$

which means that

$$Fy(k) = Fh(k)\, Fx(k) \tag{9}$$

$$\text{where, } Fh(k)= \frac{1}{2}\left(1+ e^{-j\left(\frac{2\pi}{N}\right)k}\right) \tag{10}$$

using trigonometric relations (11)

$$\cos(\varphi)=2\cos^2\left(\frac{\varphi}{2}\right) -1 \ , \ \ \sin(\phi)=2\sin\left(\frac{\varphi}{2}\right)\cos\left(\frac{\varphi}{2}\right) \tag{11}$$

$$\text{We get} \quad Fh(k)= \cos\left(\frac{\pi k}{N}\right) e^{-j\left(\frac{\pi k}{N}\right)} \tag{12}$$

from where it is obvious that

$$\big|Fh(k)\big| = \left|\cos\left(\frac{\pi k}{N}\right)\right| \text{ and } \big|\phi(k)\big| = -\frac{\pi k}{N} \tag{13}$$

which is a low-pass filter.

2-dimensional case or Local Averaging Filter.

Let image degradation equation be described by (14)

$$y(n_1,n_2)= x(n_1,n_2)*h(n_1,n_2)+ d(n_1,n_2) \qquad (14)$$

Let, also, the filtering procedure described by (15):

$$y'(n_1,n_2)= y(n_1,n_2)*h_f(n_1,n_2) \qquad (15)$$

Now, let $h_f(n_1,n_2)$ be given by (16)

$$h_f(n_1,n_2)= \frac{1}{M}\begin{matrix} 1 & 1 & \ldots. & 1 \\ 1 & 1 & \ldots. & 1 \\ \ldots & \ldots & \ldots & \ldots \\ 1 & 1 & \ldots. & 1 \\ 1 & 1 & \ldots. & 1 \end{matrix} \qquad (16)$$

then equation (15) can be written as in (17)

$$y'(n_1,n_2)= y(n_1,n_2)*h_f(n_1,n_2)=$$

$$= \sum_{l=-\frac{M-1}{2}}^{l=\frac{M-1}{2}} \sum_{k=-\frac{M-1}{2}}^{k=\frac{M-1}{2}} x(n_1-l,n_2-k)h_f(l,k) \qquad (17)$$

Let M=3 then equation (17) says that a 3x3 matrix (or mask) will move over the entire image matrix, locally performing the 2-d convolution of (17). This is shown in Figure (3).

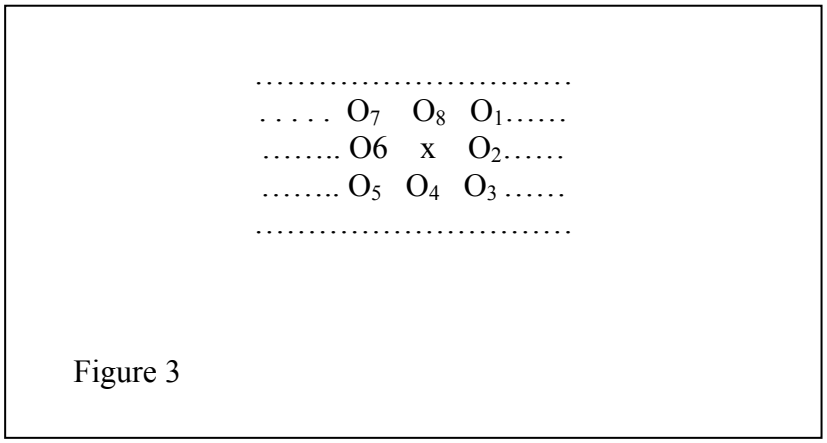


Figure 3

For clarification reasons, points on the image matrix have been labeled as $O_1,O_2,..,O_8$ and x is the point whose $y(n_1,n_2)$ is to be computed. Then the convolution operation of (17) is simply described by (18)

$$y = \frac{1}{9}\left(\sum_{i=1}^{8} Oi + x\right) \tag{18}$$

$$h_f(n_1,n_2) = \frac{1}{9}\begin{vmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{vmatrix}$$

Now, there are various filtering masks that combine various points around x in Figure (3) and with different weights. The most popular masks are described in (19)

$$SM_1 = \frac{1}{9}\begin{vmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{vmatrix} \quad \text{Smoothing Mask 1}$$

$$SM_2 = \frac{1}{5}\begin{vmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{vmatrix} \quad \text{Smoothing Mask 2}$$

$$SM_3 = \frac{1}{10}\begin{vmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{vmatrix} \quad \text{Smoothing Mask 3} \tag{19}$$

$$SM_4 = \frac{1}{9}\begin{vmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{vmatrix} \quad \text{Smoothing Mask 4}$$

Figure (5) shows the smoothing effect of mask SM_3 on the original image of Figure (4).

Figure 4 **Figure 5**

Selective Local Averaging.

As mentioned before, local averaging may reduce image noise, but at the same time it reduces high frequency image content, i.e. blurring of boundaries, edges, and of image detail. To prevent this from happening, conditional local averaging may be introduced. The idea behind the conditional or **selective local averaging** is that image points are left unchanged if some condition is satisfied. This methodology is clarified by the following examples.

Image noise reduction without boundary or edge blurring.

For this, some sort of an edge or boundary detector is applied and local averaging is performed when such features are not present. Let, local averaging be performed by SM1 mask and equation (17). First, we form the edge-detector condition, which is tested for each point x of the image matrix (see Figure (3)):

$$\text{if} \quad x' = \frac{1}{8}\sum_{i=1}^{8} Oi \quad \text{and} \quad |x - x'| \geq T \ \text{ then an edge is present} \qquad (20)$$

where T is a threshold representing a mean magnitude of intensity change close to the image edges/boundaries. Thus, the Selective Local Averaging Algorithm is as in (21):

$$y = \frac{1}{9}\left(\sum_{i=1}^{8} Oi + x\right) \qquad (21)$$

και

$$y = \begin{cases} y & if \quad |x - x'| < T \\ x & elsewhere \end{cases} \qquad (22)$$

In this way, image suppression and image blurring is avoided near the boundaries and for this it is assumed that the noise magnitude is less than T, a mean edge magnitude of the image. Further refinement of this algorithm may be achieved by taking directional local average, that is involving image points that lie in the direction along the edge. In this way we avoid averaging across the edges. Then the algorithm is modified accordingly:

a/Let smoothing mask

$$SM_1 = \frac{1}{9}\begin{vmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{vmatrix} = \frac{1}{\sum_{i=1}^{9} m_i}\begin{vmatrix} m_7 & m_8 & m_1 \\ m_6 & m_9 & m_2 \\ m_5 & m_4 & m_3 \end{vmatrix} \quad (23)$$

b/ $\begin{cases} m_i = m_1 & \text{if} \quad |x - Oi| < T \\ m_i = m_0 & \text{if} \quad |x - Oi| \geq T \end{cases}$

c/ Perform local averaging with locally modified SM1 mask. For example, if SM1 mask points O_8, O_1, O_2 are found to lie across a boundary line then: the SM1 is modified locally as

$$SM_{variable} = \frac{1}{6}\begin{vmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{vmatrix}$$

and it is used as h(m,l) in equation (17). This refined algorithm suppresses image noise while keeping image edges and boundaries sharp, but it is computationally involved.

Median Filters.

The idea behind this smoothing technique is that we replace each image value x by the median of the values surrounding that point, the value of the point itself included. Thus for 3x3 neighborhood the value x is replaced by the fifth (median) largest value. This implies that at each image matrix point a value sorting algorithm has to be applied, which increases the algorithms computational demands. However, it is not necessary to sort the values for each 3x3 point neighborhood. Once they have been sorted for the first 3x3 sub-matrix, then we can handle successive neighborhoods by deleting those at the trailing edge and merging in those in the leading edge. Figure (6) shows the result of the application of the median filter.

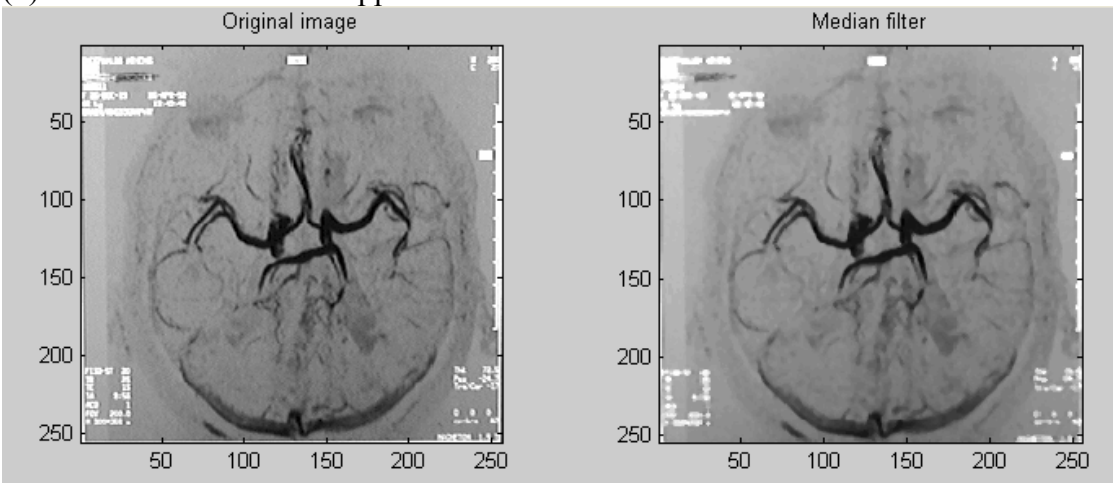


Figure 6

2.2.2. Image Sharpening

Image sharpening refers to image enhancement techniques employed to reduce blurring caused by the Point Spread Function (PSF) of the image formation process. The equation describing the image degradation process is:

$$y(n_1,n_2)= x(n_1,n_2)*h(n_1,n_2)+ d(n_1,n_2) \quad (1)$$

where $x(n_1,n_2)$ is the original image, $y(n_1,n_2)$ is the degraded image, $h(n_1,n_2)$ is the impulse response or the PSF of the image formation system, and $d(n_1,n_2)$ is the image noise. The result of image blurring is the loss of image detail and boundary clarity, that is the suppression of medium to high frequency image content. It is obvious that image deblurring or sharpening is a high frequency 'strengthening' operation or high pass filtering. However, high pass filtering should be exercised with caution since image noise resides in the high frequencies of the image spectrum, where the noise magnitude may be comparable to image signal. Thus, image sharpening may be applied after noise has been reduced or to selectively emphasize image frequencies in the high frequencies spectrum, where signal to noise ratio is low. For this, in this chapter it will be assumed that the image degradation process is described by (2):

$$y(n_1,n_2)= x(n_1,n_2)*h(n_1,n_2) \quad (2)$$

where it assumed that image noise has been suppressed and its contribution is negligible.

Differencing filter

1-d case : Figure (2) shows the signal degradation process. For the 1-d case and in accordance with equation (2) we get (3)

$$y(n)= x(n)*h(n) \quad (3)$$

Let hf(n) be the impulse response of the high pass filter, then the output is given by (4):

$$y'(n)= y(n)*h_f(n) \quad (4)$$

or by substituting (3) in (4)

$$y'(n)=\left[x(n)*h(n)\right]*h_f(n)= x(n)*\left[h(n)*h_f(n)\right] \quad (5)$$

Obviously, one would attempt to design a filter hf(n) such that

$$h(n)*h_f(n)= \delta(n) \quad (6)$$

where $\delta(n)$ is the delta function. Let h(n), the signal degrading function be of the form of (7), which has been shown (see section 2.1) to have a low passing effect on the original signal x(n). Thus, if h(n) is given by:

$$h(n)= \frac{1}{M}\{1,1,1,...,1\}, \text{ M terms} \quad (7)$$

then from (6) and (7)

$$\delta(n)= \sum_{k=0}^{M-1} h(n-k)h_f(k) \tag{8}$$

if M=2 then

$$\delta(n)= \frac{1}{2}\sum_{k=0}^{1} h(n-k)h_f(k)= \frac{1}{2}\left[h(n-0)h_f(0)+ h(n-1)h_f(1)\right] \tag{9}$$

for n=0 and from (9)

$$\delta(0)=1= \frac{1}{2}\left[h(0)h_f(0)+ h(-1)h_f(1)\right]$$

or

$$h_f(0)=2 \tag{10}$$

where h(-1)=0 since the signal degrading function is considered causal. Similarly, for n=1 and from (9) and (10)

$$h_f(1)= -2 \tag{11}$$

Thus, the required filter hf(n) to satisfy equation (6) is given by (12):

$$hf(n)=2\{1,-1\} \tag{12}$$

and using equation (4)

$$y'(n)= y(n)* h_f(n)= \sum_{k=0}^{1} y(n-k)h_f(k)=2\left(y(n)- y(n-1)\right) \tag{13}$$

and in general:

$$h_f(n)= A\{1,-1\} \tag{14}$$

described by the operation on signal y(n) as in (15)

$$y'(n)= A\left\{y(n)- y(n-1)\right\} \tag{15}$$

Frequency Response Of Differencing Filter. To obtain the frequency response of the 1-d differencing filter we take the DFT of equation (15), written here in a more convenient form:

$$y(n) = A\{x(n)-x(n-1)\} \tag{16}$$

where x(n) is the degraded signal and y(n) is the filtered output signal. The DFT of (16) is:

$$Fy(k)= A\left\{Fx(k)- Fx(k)W^{-k}\right\} = A\left\{1- W^{-k}\right\}Fx(k) \tag{17}$$

$$\text{or } Fy(k)= Fh(k)Fx(k) \tag{18}$$

where,

$$Fh(k)= A\left\{1- W^{-k}\right\} \tag{19}$$

and where, obviously, relation (19) is the frequency response of the Filter. From (19) and using the trigonometric relations:

$$\cos(\varphi)= \cos^2\left(\frac{\varphi}{2}\right) - \sin^2\left(\frac{\varphi}{2}\right) =2\cos^2\left(\frac{\varphi}{2}\right) -1 \tag{20}$$

$$\sin(\phi)=2\sin\left(\frac{\phi}{2}\right)\cos\left(\frac{\phi}{2}\right)$$

we get

$$Fh(k)=2A\sin\left(\frac{\phi}{2}\right)e^{\frac{j(\pi-\varphi)}{2}} \tag{21}$$

where, $\varphi = \frac{2\pi}{N} k$

thus, from (21)

$$|Fh(k)| = 2A \sin\left(\frac{\varphi}{2}\right) \qquad (22)$$

$$\vartheta(k) = \left(\frac{\pi - \varphi}{2}\right)$$

It is obvious from (22) and Figure 7 that the differencing filter is a high-pass filter. It is apparent from equations (18) and (22) that the effect of the differencing filter will be to cut-off completely image frequency components close to zero. This will result in the loss of most low frequency image information, such as large areas of smooth tissue, and in the emphasizing of the high frequency content, such as edges and boundaries. For this reason, this filtering technique is mainly an edge enhancement technique.

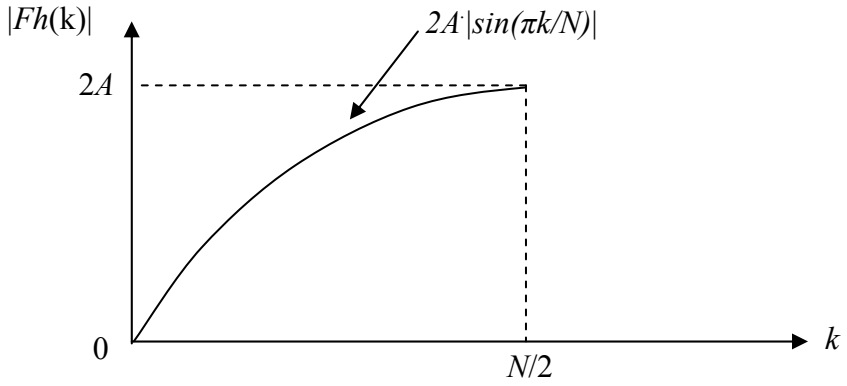


Figure 7

2-d:Laplacian Filter

For convenience, the 1-d differencing filter used for developing the 2-dimensional filter is given by relation (23):

$$y(n) = [x(n+1)-x(n)] \qquad (23)$$

Let a 5-point image sub-matrix consist of O_1, O_2, O_3, O_4, and x the central point, whose value is to be changed. If f_i is the difference of point x with point O_i and in the direction from x towards O_i, then according to (23):

$$\begin{aligned} \Delta_1 &= O_1 - x \\ \Delta_2 &= O_2 - x \\ \Delta_3 &= O_3 - x \\ \Delta_4 &= O_4 - x \end{aligned} \qquad (24)$$

Now, a simple estimation of y would be to add all four differences given by (24):

$$y = \Delta_1+\Delta_2+\Delta_3+\Delta_4 = (O_1-x)+ (O_2-x)+ (O_3-x)+((O_4-x)) =$$
$$= O_1+ O_2+ O_3+ O_4 - 4x \qquad (25)$$

Similarly, if we re-order (25)

$$y= (O_1-x)-(x-O_3)+(O_4-x)-(x-O_2) = \Delta_h^2 + \Delta_v^2 \qquad (26)$$

where, $\Delta_h^2 + \Delta_v^2$ is the horizontal and vertical second differences or the Laplacian Operators in the horizontal and vertical direction. This is the reason that the filter described by (25) will be also referred to as **the Laplacian filter.** In more formal description, relation 25 can be re-written as in (27):

$$y(n_1,n_2)= \sum_{m=-1}^{1}\sum_{l=-1}^{1} x(n_1-m,n_2-l)h(m,l) \qquad (27)$$

where,

$$h(m,l)= \begin{vmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{vmatrix} \quad LM_1 \text{ (Laplacian Mask 1)}$$

which will be referred to as Laplacian Mask 1 or LM1.

Now, if more image points are involved we can design more Laplacian masks. Several popular Laplacian masks are given below:

$$LM_2 = \begin{vmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{vmatrix} \quad LM_2 \text{ (Laplacian Mask 2)}$$

$$LM_3 = \begin{vmatrix} 1 & 2 & 1 \\ 2 & -12 & 2 \\ 1 & 2 & 1 \end{vmatrix} \quad LM_3 \text{ (Laplacian Mask 3)} \qquad (28)$$

$$LM_4 = \begin{vmatrix} -1 & 2 & -1 \\ 2 & -4 & 2 \\ -1 & 2 & -1 \end{vmatrix} \quad LM_4 \text{ (Laplacian Mask 4)}$$

In the resulting images negative values are dealt with as in equation (29)

$$y(n_1,n_2) = \begin{cases} y(n_1,n_2) & for \quad y(n_1,n_2)>0 \\ 0 & for \quad y(n_1,n_2)\le 0 \end{cases} \qquad (29)$$

Figure (8) shows the result of applying LM1 mask. Again, the edge-enhancement effect and the loss of low frequency information are obvious. These techniques are mainly used for shape description, in image analysis algorithms, and for boundary detection.

Figure 8

High emphasis filtering. Edge enhancement may be of value in many cases of image processing but equally important is image sharpening, that is retaining the low frequency image information and at the same time sharpening the edges and the overall image detail. This is achieved by the so-called high emphasis filter, which is best demonstrated by the following example:

1-d case :

Let x(n) be a signal consisting of a ramp (2 4 6 8 10) and a step (edge:.. 10 10 15 15 ...) as shown below:

Table 1: Sharpening filters example

Signal	0	2	4	6	8	10	10	10	10	10	15	15
Diff.	2	2	2	2	2	0	0	0	0	5	0	…
Lapl	…	0	0	0	0	-2	0	0	0	5	-5	…
High E	0	2	4	6	8	12	10	10	10	5	20	…
Space	0	1	2	3	4	5	6	7	8	9	10	11

where,

Diff concerns the differencing filter: $\Delta x(n)=x(n+1)-x(n)$ (30)

Lapl concerns the 1-d Laplacian filter: $\Delta^2 x(n)=x(n+1)+x(n-1)-2x(n)$ (31)

High E concerns the 1-d High Emphasis filter: $y(n)=x(n)-\Delta^2 x(n)$ (32)

First, for the sake of demonstration, we apply the difference filter (relation 30) and as it can be observed (Differ row) sharp changes occur near the edges (time intervals [4,5], [9,10], [14,15]), while the valuable information, such as ramp detail, is destroyed revealing the characteristic effect the differencing filter has on the signal as described in the previous section.

Second, we apply the Laplacian filter (relation 31) on the original signal and we observe (Laplac row) that all signal information vanishes (becomes 0) except at the edges, revealing the edge enhancement character of the filter.

Third, the need to derive a filter which preserves low frequency information and at the same time enhances edges is satisfied by the high emphasis filter (relation 32), which simply is defined as the difference between the original signal and the Laplacian filter (High E row). The example used shows that for the high emphasis filter low frequency information is retained (time intervals [5,9] and [10,14]), signal detail is preserved (given by the ramp in the time interval [0-4]) and edges are emphasized (time intervals [4,5], [9,10], [14,15]) .

2-d High Emphasis Filter.

Referring to equations (25) and (32) the 2-d high emphasis filter is given by:

$$y= x-(O_1+ O_2+ O_3+ O_4 - 4x)= 5x-(O_1+ O_2+ O_3+ O_4) \quad (33)$$

which is formally described by (34) in accordance with

$$y(n_1,n_2)= \sum_{m=-1}^{1} \sum_{l=-1}^{1} x(n_1 - m,n_2 - l)h(m,l)$$

where,

$$h(m,l)= \begin{vmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{vmatrix} \quad HEM_1 \text{ (High Emphasis Mask 1)}$$

and where h(m,l) will be referred as High Emphasis Mask 1 or HEM_1, corresponding to the Laplacian mask LM1. Similarly, the HEM masks described in (35) are the ones corresponding to the LM masks given in (28):

$$HEM_2 = \begin{vmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{vmatrix} \quad HEM_2 \text{ (High Emphasis Mask 2)}$$

$$HEM_3 = \begin{vmatrix} -1 & -2 & -1 \\ -2 & 13 & -2 \\ -1 & -2 & -1 \end{vmatrix} \quad HEM_3 \text{ (High Emphasis Mask 3)} \quad (35)$$

$$HEM_4 = \begin{vmatrix} 1 & -2 & 1 \\ -2 & -5 & -2 \\ 1 & -2 & 1 \end{vmatrix} \quad HEM_4 \text{ (High Emphasis Mask 4)}$$

Again, negative values in the resulting image are dealt with by relation (29). Figure (9) shows the result of applying HEM1 mask.

F

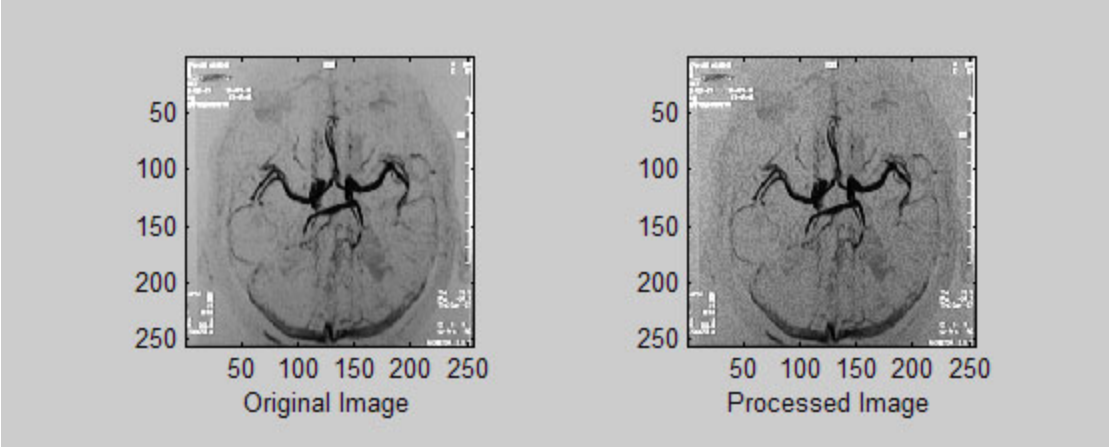

50
100
150
200
250
50 100 150 200 250
Original Image
50
100
150
200
250
50 100 150 200 250
Processed Image

Figure (9)

2.3. Frequency Domain Image Enhancement.

Image enhancement in the frequency domain, whether low-pass, high-pass, band-reject or band-pass is performed as described by equation (1):

if	2-d DFT[$x(n_1,n_2)$] = $Fx(k_1,k_2)$ is the image DFT	(a)	
and	2-d DFT[$h_f(n_1,n_2)$]= $Fh_f(k_1,k_2)$ the filter DFT	(b)	(1)
then	$y(n_1,n_2)$ = 2-d DFT[$Fx(k_1,k_2)\ Fh_f(k_1,k_2)$]	(c)	

is the enhanced image.

Equation (1c) gives the convolution procedure, which in the frequency domain is a simple multiplication of the corresponding DFT's of the image and filter.

The filters described in this section are real functions in the frequency domain, the so-called zero-phase-shift filters. They equally affect the real and imaginary parts of the image transform $Fx(k_1,k_2)$ and they do not alter the phase of the transform. Several examples of popular filter transfer functions are presented in this section in their 1-dimensional version, since their 2-dimensional version is easily obtained using (2):

$$\begin{aligned} &\text{if } \sqrt{k_1^2+k_2^2} \le N \\ &\text{then } Fh_f(k_1,k_2) = Fh_f\left(\sqrt{k_1^2+k_2^2}\right) \\ &\text{else } Fh_f(k_1,k_2) = Fh_f(N) \end{aligned} \tag{2}$$

where N is the dimension of the square image matrix.

2.3.1. Ideal Filters. Equations (3) to (6) give the ideal filters which can be implemented only by computer:

1/ Low-pass ideal

$$Fh^{LP}(k)= \begin{cases} 1, & \text{for} \quad k \le f_{co} \\ 0, & \text{for} \quad k > f_{co} \end{cases} \tag{3}$$

2/ High-pass ideal

$$Fh^{HP}(k)= \begin{cases} 0, & \text{for} \quad k \le f_{co} \\ 1, & \text{for} \quad k > f_{co} \end{cases} \tag{4}$$

3/ Band-reject ideal

$$Fh^{HP}(k)= \begin{cases} 1, & \text{for} \quad k \le d - \dfrac{w}{2} \\ 0, & \text{for} \quad d - \dfrac{w}{2} < k < d + \dfrac{w}{2} \\ 1, & \text{for} \quad k > d + \dfrac{w}{2} \end{cases} \tag{5}$$

4/ Band-pass ideal

$$Fh^{HP}(k)= \begin{cases} 0, & \text{for} \quad k \le d - \dfrac{w}{2} \\ 1, & \text{for} \quad d - \dfrac{w}{2} < k < d + \dfrac{w}{2} \\ 0, & \text{for} \quad k > d + \dfrac{w}{2} \end{cases} \tag{6}$$

or Band-pass ideal $Fh^{BP}(k) = 1 - Fh^{BR}(k)$

where, where fco is the cut-off frequency.

2.3.2. Butterworth Filters.

These filters are described by equations (7) to (10):

1/Low-pass:

$$Fh^{LP}(k) = \frac{1}{1+0.414\left(\frac{k}{f_{co}}\right)^{2n}} \quad (7)$$

where n is the degree of the filter.

2/High-pass:

$$Fh^{HP}(k) = \frac{1}{1+0.414\left(\frac{f_{co}}{k}\right)^{2n}} \quad (8)$$

3/Band-reject:

$$Fh^{BR}(k) = \frac{1}{1+0.414\left(\frac{f_{co}}{k-d}\right)^{2n}} \quad (9)$$

where d is the 'band-reject' bandwidth

4/Band-pass:

$$Fh^{BP}(k) = 1 - Fh^{BR}(k) \quad (10)$$

2.3.3. Exponential Filters.

These filters are similar to the Butterworth filters and they are described by equations (11) to (14):

1/Low-pass:

$$Fh^{LP}(k) = e^{-0.347\left(\frac{k}{f_{co}}\right)^{n}} \quad (11)$$

2/High-pass:

$$Fh^{HP}(k) = e^{-0.347\left(\frac{f_{co}}{k}\right)^{n}} \quad (12)$$

3/Band-Reject:

$$Fh^{BR}(k) = e^{-0.347\left(\frac{f_{co}}{k-d}\right)^{n}} \quad (13)$$

4/Band-pass:

$$Fh^{BP}(k) = 1 - Fh^{BR}(k) \quad (14)$$

Figure (10) shows the effect of a Butterworth low-pass and their respective frequency spectra.

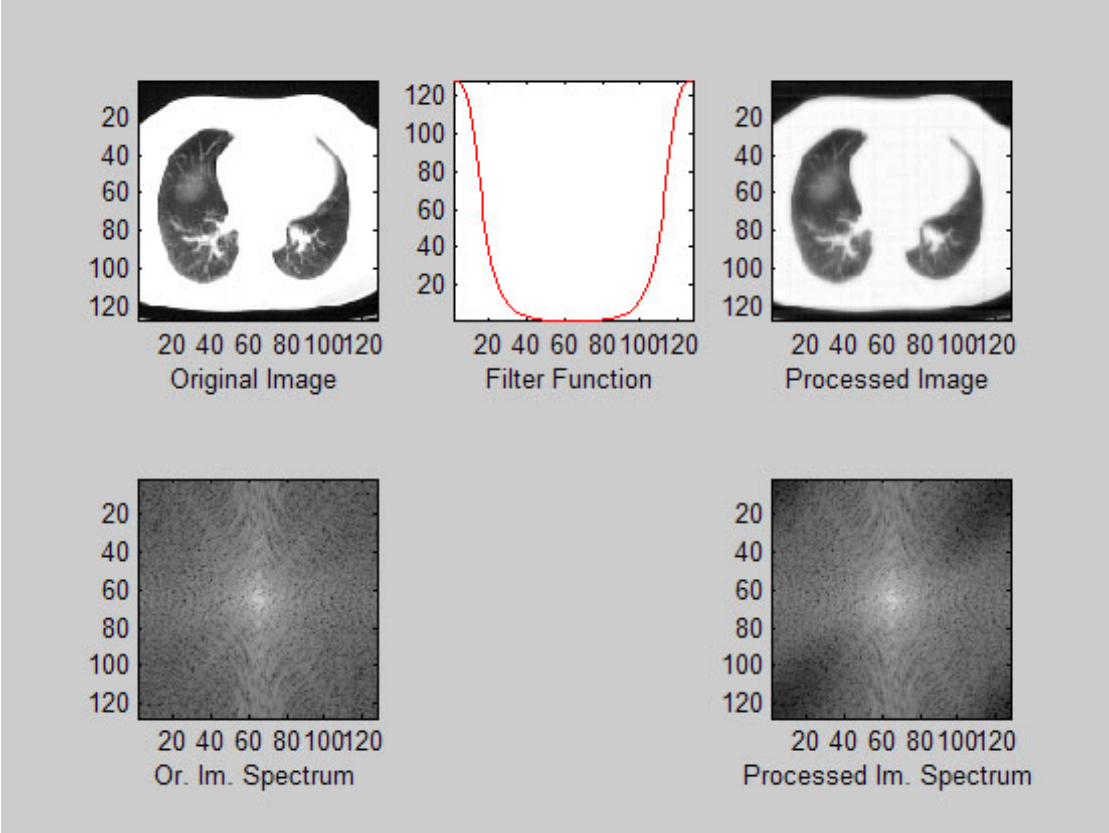


Figure 10: Smoothing by Butterworth Low-Pass

Note: Caution should be taken when applying high-pass filters, since in the way described in this section they tend to suppress most low frequency information. Thus, filters must be applied in a modified form such as the high-pass version used which is described by equation (15) :

$$Fh^{HP}(k) = Fh^{HP}(k+d) \qquad (15)$$

where d is a spatial frequency shift giving the filter response. See Figures 11 and 12 for corresponding effect.

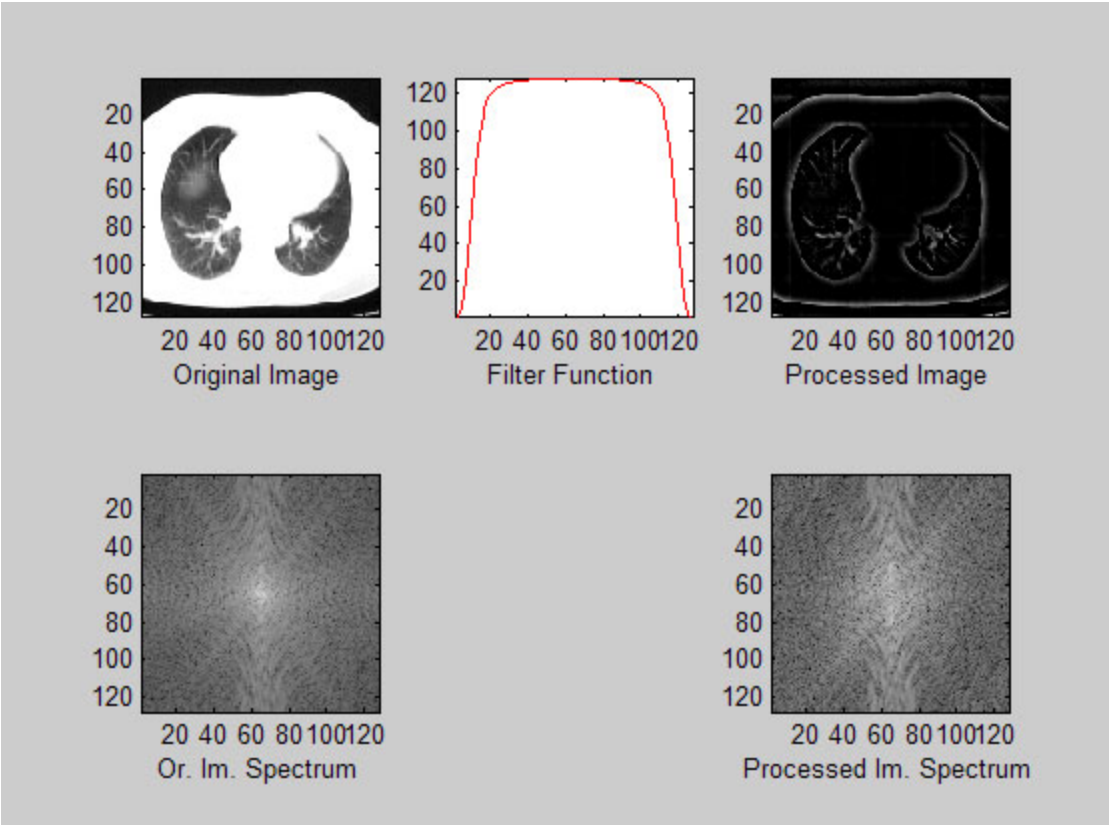


Figure 11: edge enhancement

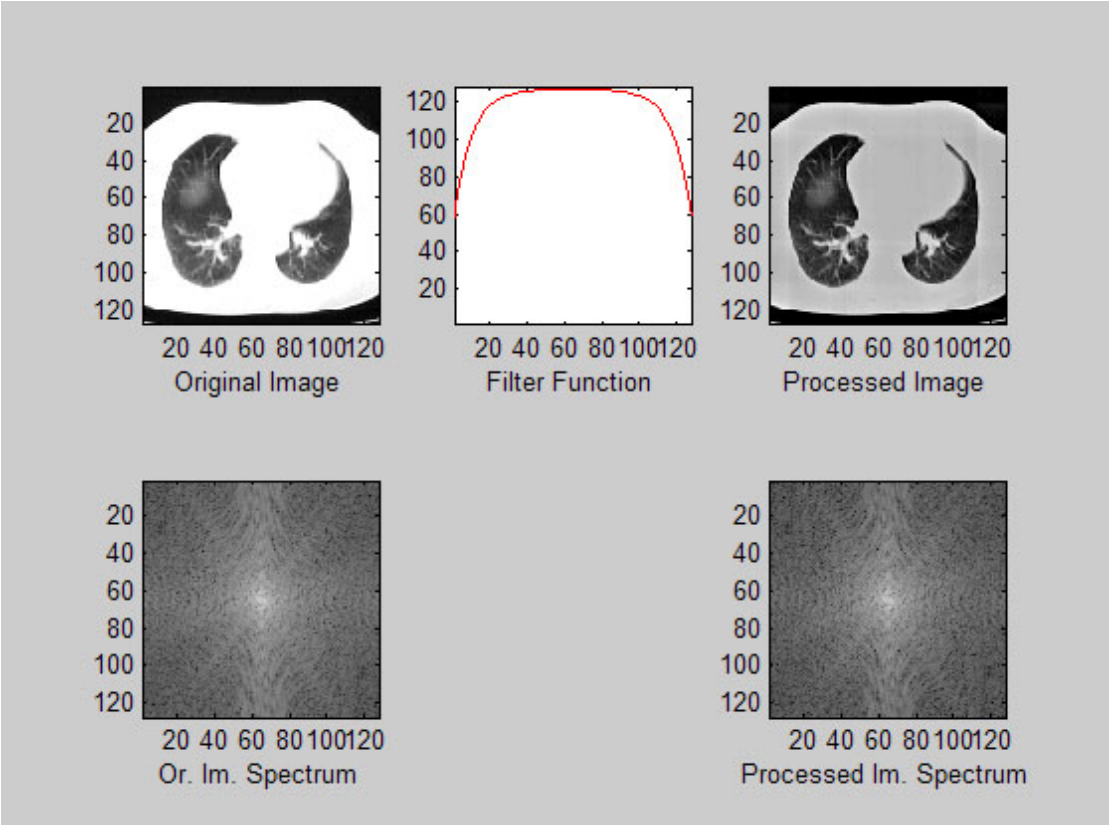


Figure 12: Butterworth High Pass: The effect of frequency shift.

2.4. Time against frequency domain image enhancement.

It can be easily proved that the spatial domain filtering, using 3x3 masks to convolve with the image, is much faster than filtering the image in the frequency domain. This can be easily assessed by studying equations (16) and (17):

spatial domain filtering:

$$y(n_1,n_2)= \sum_{l=-1}^{1} \sum_{k=-1}^{1} x(n_1 - l,n_2 - k)h_f(n_1,n_2) \qquad (16)$$

frequency domain filtering:

$$y(n_1,n_2) = \text{2d-IDFT } \{ Fx(k_1,k_2)Fh_f(k_1,k_2) \} \qquad (17)$$

However, for more precise space domain filtering, more image points have to be involved. In other words, the dimensions of the filtering mask have to be increased to 5x5 or 7x7 or 9x9 or 11x11 etc. Usually, for masks larger than 11x11 it has been shown that frequency domain filtering is faster, taking under perspective the available computer hardware (RAM memory, array processors etc.). In commercial medical imaging systems like CT-scanners, gamma-cameras, DSA etc. image enhancement is achieved mainly by a number of 3x3 masks provided by the manufacturer's software.

III. IMAGE RESTORATION

Image restoration attempts to recover the original image from the degraded by mathematically modeling the degradation process and applying the inverse process on the degraded image. All filters described in this chapter are treated for 1-dimensional case and then they are extended to two dimensions.

3.1. Inverse Filter.

1-d case :

Let the signal degradation process be described by relation (1):

$y(n) = x(n)*h(n)+d(n)$ (1)

where, x(n): the original signal h(n): the system impulse response d(n): the noise, considered here additive

Taking the DFT of (1)

$Fy(k) = Fx(k)Fh(k)+Fd(k)$ (2)

Now, Fh(k) (and similarly h(n)) can be modelled by exciting the system by a delta function (in the case of an imaging system it is a point source) and measuring the output. Safe-guarding against zeroes relation (3) is obvious:

$$Fx(k)= \frac{Fy(k)}{Fh(k)} - \frac{Fd(k)}{Fh(k)} \quad (3)$$

In the case that the noise contribution is small the gradient Fd(k)/Fh(k) can be ignored and thus

$$Fx(k)\cong \frac{Fy(k)}{Fh(k)} = Fy(k)\left[\frac{1}{Fh(k)}\right] \quad (4)$$

and if Fw(k)=[1/Fh(k)] then an estimation of the restored signal can be obtained by (5):

$$x(n)\approx IDFT\{Fy(k)Fw(k)\} \quad (5)$$

However, the noise contribution is often not negligible, especially in the high frequencies where Fh(k) is small and Fd(k) is relatively large (since the noise resides in high frequencies), thus the following pseudo-inverse filter is usually employed:

$$Fx(k)= \begin{cases} \frac{Fy(k)}{Fh(k)} & \text{if} \quad Fh(k)> C \\ \frac{Fy(k)}{C} & \text{if} \quad Fh(k)\le C \end{cases} \quad (6)$$

and $x(n)\ \chi\ IDFT\{Fx(k)\}$, where C: a small constant.

2-d case :

By analogy to (6) relation (7) is obvious:

$$Fx(k_1,k_2)= \begin{cases} \frac{Fy(k_1,k_2)}{Fh(k_1,k_2)} & \text{if} \quad Fh(k_1,k_2)> C \\ \frac{Fy(k_1,k_2)}{C} & \text{if} \quad Fh(k_1,k_2)\le C \end{cases} \quad (7)$$

Usually, $h(n_1,n_2)$ is given a gaussian shape (e.g. the PSF of the gamma-camera is of gaussian shape) of the form given in (8):

$$h(n_1,n_2)= C_1 e^{-\frac{\sqrt{n_1^2+n_2^2}}{C_2}} \tag{8}$$

where C_1 and C_2 are constants.

3.2. Wiener Filter.

The Wiener filter attempts to overcome the problem of "noise contribution" of the inverse filter.

1-d Wiener Filter

If Fx'(k) is an approximate solution to equation (2) then we are looking for a filter function Fw(k) for which

$$Fx'(k)= \frac{Fw(k)Fy(k)}{Fh(k)} \tag{1}$$

where Fy(k): is the DFT of y(n), the degraded image

Fh(k): is the spatial frequency response of the degradation process so that

$$e^2 = \sum_{k=0}^{N-1} \left|Fx'(k)- Fx(k)\right|^2 : \text{minimum} \tag{2}$$

where Fx(k) is the DFT of the original non-degraded signal.

From (1), (2) and image degradation process:

$$e^2 = \sum_{k=0}^{N-1} \left| \frac{Fw(k)\left(Fx(k)Fh(k)+ Fd(k)\right)}{Fh(k)} - Fx(k) \right|^2 =....$$

$$= \sum_{k=0}^{N-1} \frac{1}{Fh(k)} \left\{ \left|Fx(k)Fh(k)\right|^2 \left|Fw(k)-1\right|^2 + \left|Fw(k)Fd(k)\right|^2 \right\} \tag{3}$$

where we have considered that Ó $|Fd(k)Fx(k)| = 0$ since the noise and the signal do not correlate. Differentiating (3) with respect to |Fw(k)| we get

$$\frac{\partial e^2}{\partial \left|Fw(k)\right|} =0 \quad \text{or}$$

$$Fw(k)= \frac{\left|Fx(k)Fh(k)\right|^2}{\left|Fx(k)Fh(k)\right|^2 + \left|Fd(k)\right|^2} \tag{4}$$

Now, if Fw(k) is zero-phased then |Fw(k)|=Fw(k), i.e. real valued in the frequency domain (and positive from (4)) and from (1)

$$Fx'(k)=\frac{|Fx(k)Fh(k)|^2}{|Fx(k)Fh(k)|^2+|Fd(k)|^2}\frac{Fy(k)}{Fh(k)} \quad (5)$$

and safeguarding against zeroes in

$$Fx'(k)=\left[\frac{|Fh(k)|^2}{|Fh(k)|^2+\frac{|Fd(k)|^2}{|Fx(k)|^2}}\right]\frac{Fy(k)}{Fh(k)} \quad (6)$$

It is obvious from (6) that the quotient $|Fd(k)|^2 / |Fx(k)|^2$ is the ratio of the power spectra of the noise and the original signal or the "noise to signal power spectra density ratio." In the low frequencies the noise to signal ratio is almost zero, thus (6) reduces to the inverse filter. In the high frequencies the signal to noise ratio attains measurable values and the term inside the brackets in (6) becomes less than 1, thus 1/Fh(k) becomes smaller, giving an optimum restoration (in the mean square sense) in the presence of noise. In the case that the noise statistics are not known, the quotient $|Fd(k)|^2 / |Fx(k)|^2$ is considered constant giving (7)

$$Fx'(k)=\left[\frac{|Fh(k)|^2}{|Fh(k)|^2+C}\right]\frac{Fy(k)}{Fh(k)} \quad (7)$$

2-d Wiener Filter.

By analogy to (7) the 2-dimensional version of the Wiener Filter is obvious in (8)

$$Fx'(k_1,k_2)=\left[\frac{|Fh(k_1,k_2)|^2}{|Fh(k_1,k_2)|^2+C}\right]\frac{Fy(k_1,k_2)}{Fh(k_1,k_2)} \quad (8)$$

and

$x'(n_1,n_2)$ = 2-d IDFT $\{Fx'(k_1,k_2)\}$

3.3. Power spectrum equalization or Homomorphic Filter

In this restoration method an approximate solution Fx'(k) to the image degradation equation is sought such that its power spectrum $\Sigma |Fx'(k)|^2$ equals that of the original non-degraded signal $|Fx(k)|^2$.

1-d homomorphic filter.

As in the case of the Wiener filter we are looking for a filter function Fw(k) such that

$$Fx'(k)=\frac{Fw(k)Fy(k)}{Fh(k)} \quad (1)$$

where Fy(k): the DFT of the degraded signal, Fh(k): the frequency response (transfer function) of the degradation process.

With the assumption that

$$\sum_{k=0}^{N-1}\left|Fx'(k)\right|^2 = \sum_{k=0}^{N-1}\left|Fx(k)\right|^2 \quad (2)$$

From (1) and the image degradation equation and considering that the noise and the signal do not correlate, the left-hand term of (2) can be manipulated to give (3):

$$\sum_{k=0}^{N-1}\left|Fx'(k)\right|^2 = \sum_{k=0}^{N-1}\left|\frac{Fw(k)Fy(k)}{Fh(k)}\right|^2 =...=$$

$$= \sum_{k=0}^{N-1}\left|Fw(k)\right|^2 \frac{\left|Fx(k)\right|^2\left|Fh(k)\right|^2 + \left|Fd(k)\right|^2}{\left|Fh(k)\right|^2} \quad (3)$$

thus from (2) we get

$$\sum_{k=0}^{N-1}\left|Fx(k)\right|^2 = \sum_{k=0}^{N-1}\left|Fw(k)\right|^2 \frac{\left|Fx(k)\right|^2\left|Fh(k)\right|^2 + \left|Fd(k)\right|^2}{\left|Fh(k)\right|^2} \quad (4)$$

where solving for |Fw(k)| we get (5) |

$$Fw(k)= \sqrt{\frac{\left|Fx(k)Fh(k)\right|^2}{\left|Fx(k)Fh(k)\right|^2 + \left|Fd(k)\right|^2}} \quad (5)$$

and considering Fw(k) zero-phased we get (6)

$$Fx'(k)= \sqrt{\left[\frac{\left|Fx(k)Fh(k)\right|^2}{\left|Fx(k)Fh(k)\right|^2 + \left|Fd(k)\right|^2}\right]}\frac{Fy(k)}{Fh(k)} \quad (6)$$

and safeguarding against zeroes in |Fx(k)|

$$Fx'(k)= \sqrt{\left[\frac{\left|Fh(k)\right|^2}{\left|Fh(k)\right|^2 + \frac{\left|Fd(k)\right|^2}{\left|Fx(k)\right|^2}}\right]}\frac{Fy(k)}{Fh(k)} \quad (7)$$

As in the case of the Weiner filter in case that the noise statistics are not known, the quotient $\left|Fd(k)\right|^2 / \left|Fx(k)\right|^2$ is considered constant giving (8)

$$Fx'(k)= \sqrt{\left[\frac{\left|Fh(k)\right|^2}{\left|Fh(k)\right|^2 + C}\right]}\frac{Fy(k)}{Fh(k)} \quad (8)$$

It is obvious from (7) that:

1/ the term inside the brackets is similar to that of the Wiener filter only that it has been raised to the power 1/2.

2/ In the low frequencies, where the noise to signal ratio is small, the filter equals the inverse filter.

3/ In the high frequencies, where the noise contribution is significant, the term in the brackets becomes less than 1, and since it is raised to the power 0.5, its value becomes larger than the respective value of the Inverse Filter. Thus the term 1/Fh(k) when multiplied by the square root of the term in the brackets attains lower values than those of the Wiener Filter. In effect, high frequency signal suppression is less.

4/ In the middle part of the frequency spectrum the value of the filter lies between those of the Wiener and the Inverse Filter.

2-dimensional Homomorphic Filter.

By analogy to equation (8) the two dimensional homomorphic filter is described in (9)

$$Fx'(k_1,k_2)=\sqrt{\frac{|Fh(k_1,k_2)|^2}{|Fh(k_1,k_2)|^2+C}}\,\frac{Fy(k_1,k_2)}{Fh(k_1,k_2)} \qquad (9)$$

and $x'(n_1,n_2)$ = 2-d IDFT $\{Fx'(k_1,k_2)\}$

Figure (1) shows the application of the three restoration filters.

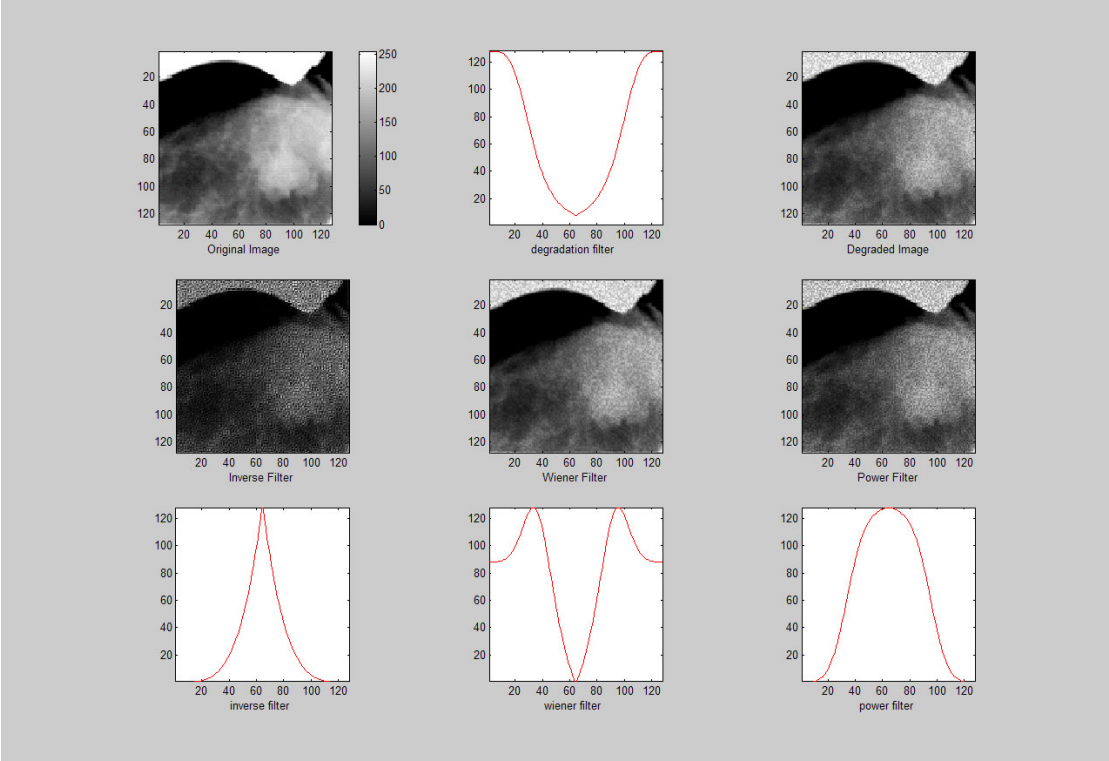


Figure 1: Restoration Filters

3.4. Generalized Wiener Filters.

The three restoration filters described in the previous paragraphs may be considered as special versions of a more generalized wiener filter described by (1)

$$Fx'(k_1,k_2)=\left[\frac{|Fh(k_1,k_2)|^2}{|Fh(k_1,k_2)|^2+\beta C}\right]^{1-\alpha}\frac{Fy(k_1,k_2)}{Fh(k_1,k_2)} \qquad (1)$$

Types of Wiener filters:

1/ for α =1 and $\beta = 0$, is the inverse filter .

2/ for α =0 and $\beta = 1$, is the wiener filter

3/ for α =0.5 and $\beta = 1$, is the power spectrum filter.

4/ various combinations of α =1 and $\beta = 0$ are possible.

IV.TOMOGRAPHIC IMAGE RECONSTRUCTION ALGORITHMS

4.1.The Fourier Reconstruction Algorithm.

Let the object in Figure (1a) be a transverse thin slice through the human body and let an x-ray beam scan the image-slice moving along the n1 axis. At the detector end we will have:

$$\left[M_2\right]_{nk} = \left[M_1 \exp\left(- \sum_{n_2=-\frac{N}{2}}^{n_2=+\frac{N}{2}} x(n_1,n_2) \right) \right]_{nk} \qquad (1)$$

where, nk: instant position of the x-ray camera.
M_1: the number of photons entering the image-slice along the nk matrix column.
M_2: the number of photons leaving the image-slice at the detector end.

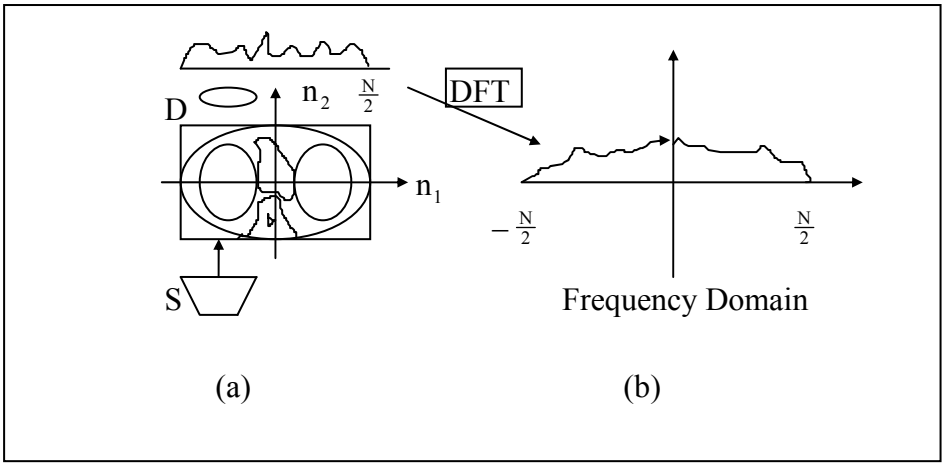


Figure 1

The number of photons detected is given by

$$\left[M_2\right]_{nk} = \left[M_1 \exp\left(- \sum_{n_2=-\frac{N}{2}}^{n_2=+\frac{N}{2}} x(n_1,n_2) \right) \right]_{nk} \qquad \textbf{(1)}$$

Rearranging (1), relation (2) is obvious

$$\left[\sum_{n_2=-\frac{N}{2}}^{n_2=+\frac{N}{2}} x(n_1,n_2)\right]_{nk} = \ln\left[\frac{M_1}{M_2}\right]_{nk} = p(nk) \tag{2}$$

where, p(nk) is the projection formed with the x-ray source positioned at nk.

Also, the DFT of the image-slice x(n1,n2) is given by

$$Fx(k_1,k_2) = \sum_{n_1=-\frac{N}{2}}^{\frac{N}{2}} \sum_{n_2=-\frac{N}{2}}^{\frac{N}{2}} x(n_1,n_2) W^{-(k_1 n_1, k_2 n_2)} \tag{3}$$

and rearranging (3)

$$Fx(k_1,k_2) = \sum_{n_1=-\frac{N}{2}}^{\frac{N}{2}} \left\{ \sum_{n_2=-\frac{N}{2}}^{\frac{N}{2}} x(n_1,n_2) W^{-(k_1 n_1)} \right\} W^{-(k_2 n_2)} \tag{4}$$

Let $k_2=0$ then from (4)

$$Fx(k_1,0) = \sum_{n_1=-\frac{N}{2}}^{\frac{N}{2}} \left\{ \sum_{n_2=-\frac{N}{2}}^{\frac{N}{2}} x(n_1,n_2) W^{-(k_1 n_1)} \right\}$$

or

$$Fx(k_1,0) = \sum_{n_1=-\frac{N}{2}}^{\frac{N}{2}} \left\{ \sum_{n_2=-\frac{N}{2}}^{\frac{N}{2}} x(n_1,n_2) \right\} W^{-(k_1 n_1)} \tag{5}$$

combining relations (2) and (5)

$$Fx(k_1,0) = \sum_{n_1=-\frac{N}{2}}^{\frac{N}{2}} \{p(n_1)\} W^{-(k_1 n_1)} \tag{6}$$

which is the 1-d DFT of p(n1), i.e.

$$Fx(k_1,0) = Fp(k_1,0) \tag{7}$$

Equation (7) says that the 1-d DFT of the image-slice projection formed with the x-ray source-detector system moving along the n_1 axis, as shown in Figure (1a), equals to the k_1 row of the 2-dimensional DFT of the image-slice shown in Figure (1b).

Relations (6) and (7) can be generalized for any slice projection sustaining an angle - with the n1 axis, thus

$$Fx(r,\varphi)= \sum_{t=-\frac{N}{2}}^{\frac{N}{2}} p(t,\varphi)W^{-rt} \qquad (8a)$$

or

$$Fx(r,\varphi)= Fp(r,\varphi) \qquad (8b)$$

where we have used polar coordinates and r is the axis in the frequency domain sustaining angle ϕ with the k_1 axis, shown in Figure (2), Fp(r, ϕ) is the 1-d DFT of the projection p(t, ϕ) along t at angle ϕ with the horizontal axis n_1. Relations (8) are referred to as the **Fourier Slice Theorem** or the **Projection Slice Theorem**, because they relate the 1-d DFT of a projection of an image-slice with the 2-d DFT of the slice itself. It follows that by altering angle ϕ between 0 and 180 degrees, we can form the 2-dimensional spatial frequency domain of the image-slice and then by taking the 2-d IDFT we can form the original image slice $x(n_1,n_2)$.

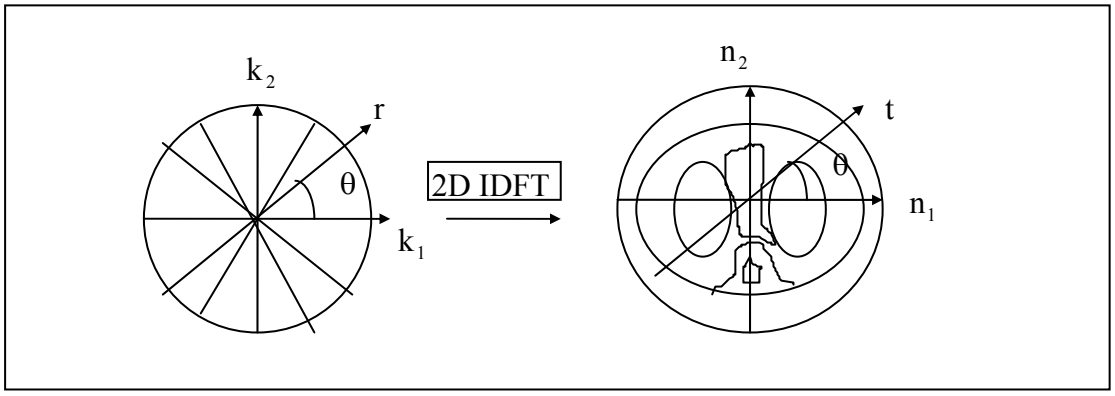


Figure 2.

The reconstruction procedure just described suffers from a serious source of error, which degrades severely the reconstructed image. As it can be observed from Figure (2) the spatial frequency domain is formed in such a way that there are more values at low frequencies than at high. Now, in the attempt to compute the Cartesian frequency domain $Fx(k_1,k_2)$ from the polar Fx(r, ϕ), the interpolation procedure will introduce a higher order of approximation errors in the high frequencies than in the low, because, in the former, polar points are further apart. This introduces a high frequency suppression (smoothing) effect, which is analogous to dividing the $Fx(k_1,k_2)$ by $\sqrt{k_1^2+k_2^2}$, the distance from the origin. To compensate for this we have to multiply $Fx(k_1,k_2)$ by that factor, i.e.

$$Fx(k_1,k_2)\sqrt{k_1^2+k_2^2} = T\left[Fx(r,\varphi)\right] \qquad (9)$$

where T[Fx(r, ϕ)] denotes the transformation procedure from polar to cartesian coordinates through linear interpolation.

Thus, the obvious solution to the tomographic image reconstruction problem is given in (10)

$$x(n_1,n_2)=\frac{1}{NN}\sum_{k_2=-\frac{N}{2}}^{\frac{N}{2}}\sum_{k_1=-\frac{N}{2}}^{\frac{N}{2}}Fx(k_1,k_2)\sqrt{k_1^2+k_2^2}\,W^{(k_1n_1,k_2n_2)} \qquad (10)$$

or

$$x(n_1,n_2)=2-d\_IDFT\left\{Fx(k_1,k_2)\sqrt{k_1^2+k_2^2}\right\}$$

This solution is called the Fourier Reconstruction Algorithm. Although there have been several examples of successful implementation of this algorithm, it has not been implemented in medical systems, for the following reasons:

1/ Polar to Cartesian coordinates transformation is performed in the frequency domain, which is time consuming, computer memory demanding, and erroneous.

2/ The 2-d DFT procedure in (10), even when fast FFT algorithms are employed, is time consuming, computer memory demanding (complex arrays), and erroneous, due to the wrap-around error associated with the periodic nature of the DFT in the convolution. Compensating for the latter with zero-padding and windowing, increases the computer memory and the computation time demands.

4.2 Convolution/Filter Back-Projection Algorithm.

It is the most popular image reconstruction algorithm used by most imaging modalities (e.g. CT, SPECT) and it is characterized by its accuracy and speed of reconstruction. It is derived by simple manipulation of equation (10) so that frequency domain linear interpolation and use of 2-d DFT's are avoided.

Rewriting the right hand side of equation (10) in polar form

$$x(n_1,n_2)=\frac{1}{NN}\sum_{\varphi=-0}^{\pi}\sum_{r=-\frac{N}{2}}^{\frac{N}{2}}Fx(r,\varphi)|r|W^{rk} \qquad (11)$$

where, $t=n1\cos\phi + n2\sin\phi$ as seen in Figure (2).

Now, from relations (8)

$$Fx(r,\phi)=\sum_{t=-\frac{N}{2}}^{\frac{N}{2}}p(t,\phi)W^{-rt}=Fp(r,\theta)\ \text{(DFT)} \qquad (12\alpha)$$

also,

$$p(t,\varphi)=\frac{1}{N}\sum_{r=-\frac{N}{2}}^{\frac{N}{2}}Fp(r,\phi)W^{rt}\ \text{(IDFT)} \qquad (12\beta)$$

thus, using relations (12) in (11)

$$x(n_1,n_2)=\frac{1}{N}\sum_{\varphi=-0}^{\pi}\frac{1}{N}\sum_{r=-\frac{N}{2}}^{\frac{N}{2}}Fp(r,\varphi)|r|W^{rk}$$

or

$$x(n_1,n_2)=\frac{1}{N}\sum_{\varphi=-0}^{\pi}p'(t,\phi) \qquad (13\alpha)$$

where, 13.

$$p'(t,\varphi)=\frac{1}{N}\sum_{r=-\frac{N}{2}}^{\frac{N}{2}}Fp(r,\varphi)|r|W^{rk} \qquad (13\beta)$$

Formulae 13(a) and 13(b) say that from each projection p(t, ϕ), we first calculate a filtered projection p'(t, ϕ) by means of 13(b) and then use 13(a) to obtain the image-slice $x(n_1,n_2)$.

Remarks:

1/ Relation 13(b) describes the filtering operation in the frequency domain with a filter function |r|, which is a ramp in the frequency domain. To avoid abrupt frequency changes at the high frequency end of the filter, a windowing function is usually employed in the frequency domain, and to avoid the wrap-around error effect zero-padding is also used. Thus, equation 13(b) is employed in the following form

$$p'(t,\theta)=1-d\_DFT\left(Fpe(r,\varphi)|r|w_f(r)\right) \qquad (14)$$

where, Fpe(r, ϕ) is a 2N 1-d array formed from the Fp(r, ϕ) with the last N complex values filled with zeroes and wf(r) is a window array of 2N length. Equation 13(b) can also be performed in the spatial domain, using the convolution formula below:

$$p'(t,\varphi)=\frac{1}{N}\sum_{t=-\frac{N}{2}}^{\frac{N}{2}}p(t,\varphi)h(t-n)=p(t,\varphi)*h(t) \qquad (15)$$

where

h(t) = 1-d IDFT { |r| wf(r) } is computed once only.

Equation (15) and 13(a) can be designed in hardware and provide fast implementation of the algorithm.

2/ Equation 13(a) may be interpreted as a back-projection operation, that is projecting back all the filtered projections and summing all the values assigned to each pixel x(n1,n2) through linear interpolation.

Accordingly, if equation (14) is combined with 13(a) then the algorithm is called **Filter Back Projection** and if (15) is used then it is called **Convolution Back Projection.**

3/ The window filtering functions used in (14) and (15) are the following:

Rectangular: wf(r) = 1 (otherwise Ram-Lak)

Hanning: wf(r) = 0.5(1-cos(2π r/N))

Hamming: wf(r) = 0.54 - 0.46cos(2π r/N)

Blackman: wf(r) = 0.42 - 0.5cos(2π r/N)+0.08cos(4π r/N)

Shepp-Logan: wf(r) = sin(π r/N) / (π r/N)

Cosine: wf(r) = cos(π r/N)

Figure 3 shows the reconstruction using the Shepp-Logan filter

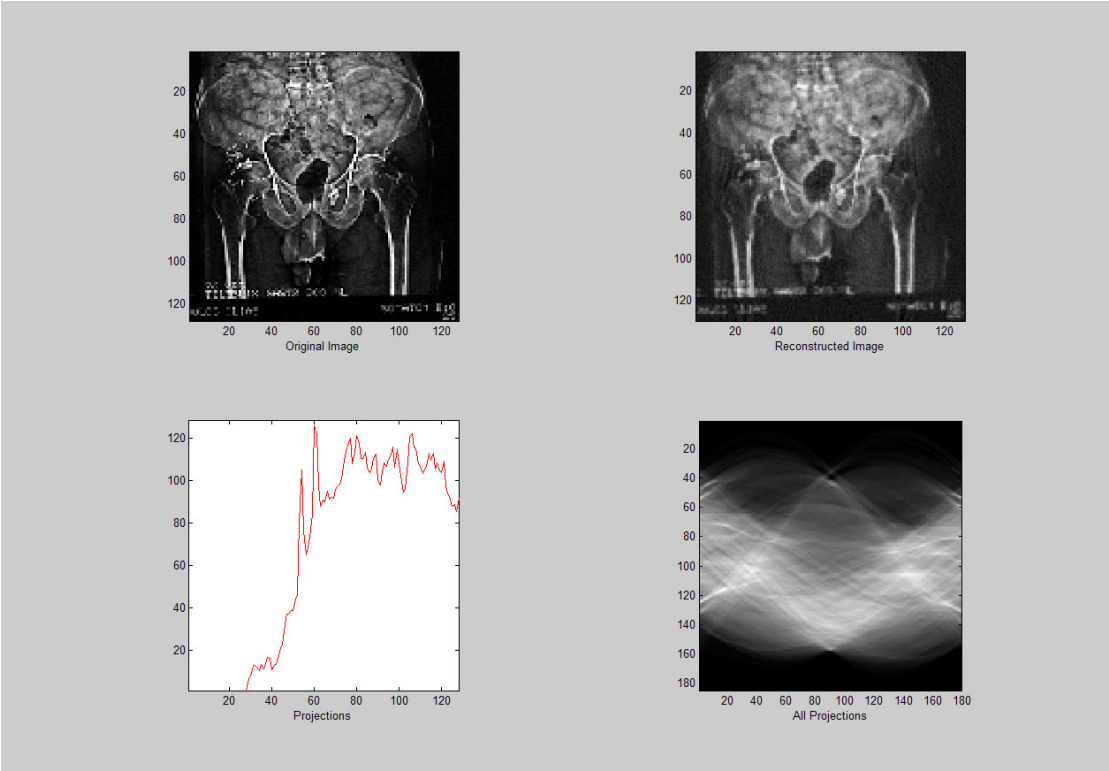


Figure 3

REFERENCES.

1/ Rogers D.F. "Procedural elements for computer graphics"McGraw-Hill (1985).
2/ Gonzalez R.C., Wintz P. "Digital image processing". Addison-Wesley (1977).
3/ Pratt W.K. "Digital image processing ". Wiley(1978).
4/ Rosenfeld A. and Kak A.C. "Digital picture processing" 2nd Edition. Academic Press (1982).
5/ Jain A.K. "Fundamentals of digital image processing". Prentice Hall (1989).

COMPUTER PROGRAMS

%Program_1

```
% Read & plot & window *.dat image
function []=Program_1();
clc;echo off;close all;
A=[ 30,31,12, 9,
        17,12,25,10,
        12, 8,17, 9,
        31,12,26,22];
A=double(A);B=A;
disp('original image matrix');disp(A);

image_depth=31;tones=8;
B=My_plot(A,tones);% B holds grey-tone values only
value=1;
switch value
        case 1
                WW=20;WL=20;
                B=My_simple_window(A,image_depth,tones,WW,WL);
        case 2
                gray_val=5;im_val=21;
                B=My_broken_window(A,image_depth,tones,gray_val,im_val);%Have to construct it

        case 3
                ww1=10;ww2=10;wl1=10;wl2=25;
                B=My_double_window(A,image_depth,tones,ww1,ww2,wl1,wl2);%Have to construct it
end
disp(round(B));
%=======================================================
function [C]=My_plot(A,tones);
x=size(A,1);y=size(A,2);
```

```
for i=1:x
	for j=1:y
		ival=A(i,j);
			tone_ival=(tones-1)*(double(ival)-0)/(31-0);
		C(i,j)=tone_ival;
	end;
end;
disp(round(C));
%=============================================================
function [C]=My_simple_window(A,image_depth,tones,WW,WL);
% WL=window level
% WW=window level
x=size(A,1);y=size(A,2);
We=(2.0*WL+WW)/2.0;
if(We>image_depth) We=image_depth;end;
Ws=We-WW;
if(Ws<0) Ws=0;end;
for i=1:x,
	for j=1:y,
		ival=A(i,j);
		if (ival<=Ws) tone_ival=0; end;
		if (ival>=We) tone_ival=tones-1;end;

		if ( ival>=Ws & ival<=We)
			tone_ival=(tones-1)*(double(ival)-Ws)/(We-Ws);
		end;
		C(i,j)=tone_ival;
	end;
end;
%=============================================================================
%Lab Assignment:
%1/type main program,  simple_window function, and compare results with ones calculated
%2/use routine in graphics program Program_1_gr.m
%Homework: repeat steps 1 & 2 for broken_window and double_window (have to construct your own routines) .
```

%Graphics version of Program_1

```
%Program 1 :Read image and process by simple window, broken window, double
%window
function []=Program_1_gr();
clc;echo off;close all;
A=imread('..\Images 128x128\Pelvis.BMP');%Pelvis.bmp HEAD6.BMP
A=double(A);
x=size(A,1);y=size(A,2);
sprintf('x= %f  y=%f',x,y);
max_A=max(max(A));min_A=min(min(A));A=(A-min_A)*(255/(max_A-min_A));%back to 0-255
B=A;
image_depth=255;tones=256;
tic
value =1;
switch value
        case 1
                %Put code for simple window here
                WW=256;WL=20;
                B=My_simple_window(A,image_depth,tones,WW,WL);
                %End of code for simple window here
        case 2
                %Put code for broken window here
                gray_val=150;im_val=60;
                B=My_broken_window(A,image_depth,tones,gray_val,im_val);%Have to construct it
                %End of code for simple window here
        case 3
                %Put code for double window here
                ww1=50;ww2=50;wl1=50;wl2=220;
                B=My_double_window(A,image_depth,tones,ww1,ww2,wl1,wl2);%Have to construct it
end
%========= PLOT IMAGES ====================

colormap('gray');
switch value
        case 1
```

```
                subplot(1,2,1);imagesc(A);xlabel('Original Image');%colorbar;
                axis equal;axis([1 size(A,2) 1 size(A,1)]);
                subplot(1,2,2);imagesc(B);xlabel('Single-Window Processed Image');
                axis equal;axis([1 size(B,2) 1 size(B,1)]);
        case 2
                subplot(1,2,1);imagesc(A);xlabel('Original Image');%colorbar;
                axis equal;axis([1 size(A,2) 1 size(A,1)]);
                subplot(1,2,2);imagesc(B);xlabel('Broken-Window Processed Image');
                axis equal;axis([1 size(B,2) 1 size(B,1)]);
        case 3
                subplot(1,2,1);imagesc(A);xlabel('Original Image');%colorbar;
                axis equal;axis([1 size(A,2) 1 size(A,1)]);
                subplot(1,2,2);imagesc(B);xlabel('Double-Window Processed Image');
                axis equal;axis([1 size(B,2) 1 size(B,1)]);
end

toc
%=========================================================
function [C]=My_simple_window(A,image_depth,tones,WW,WL);
C=A;
% WL=Window level
% WW=Window width

%Put your code below
```

%Program 2

```
% Read & plot & histogram equalization *.dat image
function []=Program_2();
clc;echo off;close all;
A=[ 30,31,12, 9,
        17,12,25,10,
        12, 8,17, 9,
        31,12,26,22];
A=double(A);B=A;
disp(A);
image_depth=31;tones=8;
B=My_plot(A,tones);
h_A=My_hist_A(A,tones);disp(h_A);
value=2;
switch value
        case 1
                B=CDF(A,tones);%Have to construct it
        case 2
                B=My_hequal(A,tones);
end
disp(round(B));
h_B=My_hist_A(B,tones);disp(h_B);

%=============================================
function [C]=My_plot(A,tones);
x=size(A,1);y=size(A,2);
for i=1:x
        for j=1:y
                ival=A(i,j);
                        tone_ival=(tones-1)*(double(ival)-0)/(31-0);
                C(i,j)=tone_ival;
        end;
end;
```

```
disp(round(C));
%==============================================
function [m]= My_hequal(im,tones)
tic
m=im;
x=size(im,1);y=size(im,2);
max_im=max(max(im));
hh=zeros(tones,1);qq=zeros(tones,1);

for i=1:x,
        for j= 1:y ,
                m(i,j)= -1;
        end;
end;
neq = x*y/tones;
%{form image histogram}
for i= 1 :x ,
        for j= 1: y,
                iz = (tones-1)*im(i,j)/max_im;
                im(i,j)=round(iz);
                hh(round(iz+1)) = hh(round(iz+1)) + 1;
        end;
end;

% %{------------ HISTOGRAM EQUALIZATION ----------------}
k=0;
for it = 1 : tones ,
        itone = it;
        i = 0;
        isum = 0;
        while (i <= tones-1 & isum < neq)
                i=i+1;
                isum = isum + hh(i);
                k = i;
                hh(i) = 0;
```

```
        end; %{ of while}
        idif1 = isum - neq;
        hh(k) = idif1;
        for ii = 1 : x ,
                for jj = 1 : y,
                        iz = im(ii,jj);
                        if (qq(it) < neq)
                                if (m(ii,jj) == -1 & iz <= k)
                                        m(ii,jj) = itone-1;
                                        qq(it) = qq(it) + 1;
                                end;% {of if}
                        end; %{ of if}
                end; %{of jj}
        end; %{of ii}
        if (qq(it) ~= neq),
                if (qq(it) < neq)
                        for ii = 1 : x,
                                for jj = 1 : y,
                                        iz = im(ii,jj);
                                        if (m(ii,jj) == -1 & iz == k)
                                                m(ii,jj) = itone-1;
                                                qq(it) = qq(it) + 1;
                                        end; %{ of if}
                                end; %{of jj}
                        end; %{of ii}
                end; %{of if}
        end; %{of if}
end; %{of it}
toc
%=============================================================
function [h]=My_hist_A(A,tones)
x=size(A,1);y=size(A,2);
h=zeros(tones,1);

maxi=max(max(A));if (maxi<=0) maxi=1;end;
```

```
for i=1:x,
  for j=1:y,
            p=(tones-1)*A(i,j)/maxi;
            h(round(p+1))=h(round(p+1))+1;
  end;
end;
%=============================================================================
%Lab Assignment:
%1/type main program, function hequal and compare results with ones calculated
%2/use routine in graphics program Program_2_gr.m

%3/Homework: repeat steps 1 & 2 for CDF (have to construct your own routine CDF).
```

%Graphics version of Program 2

```
%Program 1:Read image and process by CDF
function []=Program_2_gr();
clc;echo off;close all;
A=imread('..\Images 128x128\Pelvis.BMP');%Pelvis.bmp HEAD6.BMP
A=double(A);
x=size(A,1);y=size(A,2);
sprintf('x= %f  y=%f',x,y)
max_A=max(max(A));min_A=min(min(A));A=(A-min_A)*(255/(max_A-min_A));%back to 0-255
B=A;
image_depth=255;tones=256;
%==========CALL FUNCTIONS=================
value=2;
switch value
        case 1
                B=My_CDF(A,tones);%Have to construct it
        case 2
                B=My_hequal(A,tones);
end

max_B=max(max(B));min_B=min(min(B));B=(B-min_B)*(255/(max_B-min_B));%back to 0-255
%========= PLOT IMAGES ====================
colormap('gray');
subplot(2,2,1);imagesc(A);xlabel('Original Image');
axis equal;axis([1 size(A,2) 1 size(A,1)]);

subplot(2,2,2);imagesc(B);xlabel('CDF Processed Image');
axis equal;axis([1 size(B,2) 1 size(B,1)]);
h=My_hist_A(A,tones);
maxh=max(h);minh=min(h);h=(h-minh)*(tones/(maxh-minh));%normalize for plotting;
subplot(2,2,3);plot(h,'red');xlabel('Original Image histogram ');
axis equal;axis([1 255 1 max(h)]);

switch value
```

```
        case 1

                eq=My_hist_A(B,tones);
                maxeq=max(eq);mineq=min(eq);eq=(eq-mineq)*(tones/(maxeq-mineq));%normalize for plotting;
                subplot(2,2,4);plot(eq,'blue');xlabel('CDF histogram ');
                axis equal;axis([1 255 1 max(eq)]);
        case 2
                eq=My_hist_A(B,tones);
                subplot(2,2,4);plot(eq,'blue --');xlabel('equalized histogram ');
                axis equal;axis([1 512 1 512]);

end

%=================================================
function [m]= My_hequal(im,tones)
tic
m=im;
%Put the code below
toc

%=============================================================
function [h]=My_hist_A(A,tones)
x=size(A,1);y=size(A,2);
h=zeros(tones,1);

maxi=max(max(A));if (maxi<=0) maxi=1;end;
for i=1:x,
  for j=1:y,
          p=(tones-1)*A(i,j)/maxi;
          h(round(p+1))=h(round(p+1))+1;
  end;
end;
toc
```

%Program 3

```
% Read & plot & find spectrum of *.dat image
function []=Program_3();
clc;echo off;close all;
A=[30,31,12, 9,
        17,12,25,10,
        12, 8,17, 9,
        31,12,26,22];

A=double(A);B=A;
disp(A);
image_depth=31;tones=8;

value =1;
switch value
        case 1
                B=ampl_fft(A);
        case 2
                B=ampl_fft2(A);
end;
max_B=max(max(B));min_B=0;B=(B-min_B)*(image_depth/(max_B-min_B));%back to image_depth
disp(round(B));

%=====================================================================
function [C]=ampl_fft(A)
x=size(A,1);y=size(A,2);
%/*--------   2D - FFT  ---------*/
%/*----------   do rows first   ----------*/
C=A;
for i=1:x,
        for j=1:y,
                if ( rem((i+j),2) == 1) C(i,j) = -C(i,j);
                else C(i,j) = C(i,j);
                end;%if
```

```
        end;%j
end;%i

for i=1:x

        for j=1:y,
                Cy(j)=C(i,j);
        end;%j
        Cy=fft(Cy,y);
        for j=1:y,
                C(i,j) = Cy(j);
        end;%j

end;%of i
%/*------   do columns next---*/
for j=1:y %for each column

        for i=1:x
                Cx(i)=C(i,j);
        end;%i
        Cx=fft(Cx,x);
        %/*-----   Compute Log Amplitude   -----*/
        for i=1:x
                C(i,j) = round(10.0 * log(1+ abs(Cx(i)) ));
        end;%i

end;%j

%=====================================================================
function [C]=ampl_fft2(A)
x=size(A,1);y=size(A,2);

%/*--------   2D - FFT  ---------*/
for i=1:x;
        for j=1:y;
```

```
                    if ( rem((i+j),2) == 1) A(i,j) = -A(i,j);
                    else A(i,j) =A(i,j);
                    end;%if
          end;%j
end;%i
C=fft2(A);
C=round(10.0 * log(abs(C)+1));
%=======================================


%Lab Assignment:
%form the fft_spectrum by typing main program and 2 routines:
%1/1d-fft routine:ampl_fft(A); 2d-fft routine:ampl_fft2(A);
%2/write down the time-interval for assessing the time difference
%between the 2 routines.
%inspect the spectra of various images using the graphics version of Program 3
```

%Graphics version of program 3

```
% BMP image read & plot & calculate image spectrum
function []=Program_3_gr();
clc;echo off;close all;
% 128x128 images
% BODY1.BMP,BODY2.BMP,BODY2.BMP,BODY3.BMP,BODY4.BMP, BODY5.BMP,
% CHEST.BMP,Pelvis.bmp
% HEAD1.BMP,HEAD2.BMP,HEAD3.BMP,HEAD4.BMP,HEAD5.BMP,HEAD6.BMP,
% mam1.bmp,Mam2.bmp,Mam3.bmp,MAM10a.bmp
A=imread('..\Images 128x128\Pelvis.bmp');
A=double(A);
x=size(A,1);y=size(A,2);
sprintf('x= %f  y=%f',x,y)
max_A=max(max(A));min_A=0;A=(A-min_A)*(255/(max_A-min_A));%back to 0-255
B=A;

tic
%---------------------------------------
%Amplitude Spectrum
value =2;
switch value
        case 1
                B=ampl_fft(A);
        case 2
                B=ampl_fft2(A);
end;
%============PLOT IMAGES===============
max_B=max(max(B));min_B=min(min(B));B=(B-min_B)*(255/(max_B-min_B));%back to 0-255
colormap('gray');
subplot(1,2,1);imagesc(A);xlabel('Original Image');
axis equal;axis([1 size(A,2) 1 size(A,1)]);
subplot(1,2,2);imagesc(B);xlabel('Image Spectrum');
axis equal;axis([1 size(B,2) 1 size(B,1)]);
toc
```

```
%=====================================================================
function [C]=ampl_fft(A)
%Put your code here
end;%j

%=====================================================================
function [C]=ampl_fft2(A)
%Put your code here
%=======================================
```

%Program 4

```
% Read & plot & convolve *.dat image
function []=Program_4();
clc;echo off;close all;
A=[30,31,12, 9,
           17,12,25,10,
           12, 8,17, 9,
           31,12,26,22];
sm1  = [1,1,1;1,1,1;1,1,1];%Low pass or smoothing
sm2  = [-1,-1,-1;-1, 9,-1;-1,-1,-1]; %High Emphasis
sm4 = [  0, -1, 0;-1, 5,-1; 0,-1, 0]; %High Emphasis
sm3 = [  0, 1, 0 ; 1,-4, 1; 0, 1, 0]; %Laplacian



A=double(A);B=A;C=A;D=A;
disp ('A');disp((A));
image_depth=31;tones=8;

value =1;
switch value
case 1
   B=convolve (A,sm1);
%          B=conv2(A,sm1,'same');
case 2
   B=median_filter(A);
case 3
   B=sharp(A,0.1); %%  if A(i,j)> local_mean  then B(i,j)=A(i,j)*threshold*A(i,j); (let threshold=01.)
end;
C=ampl_fft2(A);
D=ampl_fft2(B);
B=Normalize_Matrix(B,image_depth);C=Normalize_Matrix(C,image_depth);D=Normalize_Matrix(D,image_depth);
disp ('B');disp(round(B));disp ('Amplitute A');disp(round(C));disp ('Amplitude B');disp(round(D));
```

```
%=========================================================================
function [B]= convolve (A,sm)
%-------------------------------------
%convolution
x=size(A,1);y=size(A,2);
B=A;%important for frame
% Initialize
mask=sm;
mask=double(mask);
sum_mask=sum(sum(mask));if sum_mask<=0 sum_mask=1;end;
%filter image
for i=2:x-1,
        for j=2:y-1,
                value=0;icount=0;
                for ii=i-1:i+1,
                        icount=icount+1; jcount=0;
                        for jj=j-1:j+1,
                                jcount=jcount+1;
                                value=value+A(ii,jj)*mask(icount,jcount);
                        end;
                end;
                if (value<0) value=0;end;
                B(i,j)= value/sum_mask;
        end;
end;
%=========================================================================
function [C]=ampl_fft2(A)
x=size(A,1);y=size(A,2);

%/*--------   2D - FFT  ---------*/
for i=1:x;
        for j=1:y;
                if ( rem((i+j),2) == 1) A(i,j) = -A(i,j);
                else A(i,j) =A(i,j);
                end;%if
```

```
        end;%j
end;%i
C=fft2(A);
C=round(10.0 * log(abs(C)+1));
%=========================================
function [C]=Normalize_Matrix(A,tones)
max_A=max(max(A));
min_A=min(min(A));min_A=0;
C=(A-min_A)*(tones)/(max_A-min_A);%back to 0-(tones)

%================================================================================
%Lab Assignment:
%1/type in function convolve and compare results with ones calculated
%2/use routine in graphics program Program_4_gr.m

%Homework:
%2/form the median filter function and the sharp filter function, compare
%results with ones calculated, and use routines in graphics program Program_4_gr.m
```

%Graphics version of Program 4

```
% Convolution in time domain & spectra
function []=Program_4_gr();
clc;echo off;close all;
% 128x128 images
% BODY1.BMP,BODY2.BMP,BODY2.BMP,BODY3.BMP,BODY4.BMP, BODY5.BMP,
% CHEST.BMP,Pelvis.bmp
% HEAD1.BMP,HEAD2.BMP,HEAD3.BMP,HEAD4.BMP,HEAD5.BMP,HEAD6.BMP,
% mam1.bmp,Mam2.bmp,Mam3.bmp,MAM10a.bmp
A=imread('..\Images 128x128\Pelvis.bmp');%Pelvis.bmp HEAD6.BMPA=double(A);
x=size(A,1);y=size(A,2);
sprintf('x= %f  y=%f',x,y)
Normalize_Matrix(A,255);
sm1  = [1,1,1;1,1,1;1,1,1];%Low pass or smoothing
sm2 = [  0, 1, 0 ; 1,-4, 1; 0, 1, 0]; %Laplacian
sm3  = [-1,-1,-1;-1, 9,-1;-1,-1,-1]; %High Emphasis
sm4 = [  0, -1, 0;-1, 5,-1; 0,-1, 0]; %High Emphasis

tic
B=A;C=A;D=A;

value =1;
switch value
        case 1
                B=convolve (A,sm1);
        case 2
                B=median_filter(A);
   case 3
      B=sharp(A,0.1);
end;

C=ampl_fft2(A);
D=ampl_fft2(B);
image_depth=255;
```

```
B=Normalize_Matrix(B,image_depth);C=Normalize_Matrix(C,image_depth);D=Normalize_Matrix(D,image_depth);
%==============PLOT IMAGES=======================
colormap('gray');
subplot(2,2,1);imagesc(A);xlabel('Original Image');
axis equal;axis([1 size(A,2) 1 size(A,1)]);
subplot(2,2,2);imagesc(B);xlabel('Processed Image');
axis equal;axis([1 size(B,2) 1 size(B,1)]);

subplot(2,2,3);imagesc(C);xlabel('Original Image Spectrum');
axis equal;axis([1 size(C,2) 1 size(C,1)]);

subplot(2,2,4);imagesc(D);xlabel('Processed Image Spectrum');
axis equal;axis([1 size(D,2) 1 size(D,1)]);
toc


%=========================================================================
function [B]= convolve (A,sm)
%Put your code here
%=========================================================================
function [C]=ampl_fft2(A)
%Put your code here
%=======================================
function [C]=Normalize_Matrix(A,tones)
%Put your code here
%=========================================================================
```

%Program 5

```
% Read & plot & convolve in frequency domain *.dat image
function []=Program_5();
clc;echo off;close all;
A=[30,31,12, 9,
        17,12,25,10,
        12, 8,17, 9,
        31,12,26,22];
A=double(A);B=A;
A=double(A);
x=size(A,1);y=size(A,2);
disp ('A');disp((A));
image_depth=31;tones=8;

%-------------Filtering Parameters -------------------------
value=2;
switch value
        case 1
                lowhi=1;%1:low-pass, 2:high-pass
                ndegree=2;%degree of filter
                ifco=round(y*0.2);%cut-off frequency !!! change 0.2 between 0.2 and 0.9!!!
        case 2
                lowhi=2;%1:low-pass, 2:high-pass
                ndegree=2;%degree of filter
                ifco=round(y*0.2);%cut-off frequency !!! change 0.2 between 0.2 and 0.9!!!
                trans=round(y*0.06);%translation for not removing low freq info
end;


%-----------------------------------------------------------
fh=Butt_filt(x,lowhi,ndegree,ifco,trans);
disp('fh');for i=1:4;disp(fh(i));end;
B=filt_in_freq2(A,image_depth,fh);
C=ampl_fft2(A);
```

```
D=ampl_fft2(B);
B=Normalize_Matrix(B,image_depth);C=Normalize_Matrix(C,image_depth);D=Normalize_Matrix(D,image_depth);
disp ('B');disp(round(B));disp ('Amplitute A');disp(round(C));disp ('Amplitude B');disp(round(D));

%=====================================================================
%//-------------------------------------------------
function [fh]=Butt_filt(N,lowhi,ndegree,ifco,trans)
for i=1:N; fh(i)=0;end;
%//---------Read in filter parameters--------
Ndiv2=N/2;
for k=1:Ndiv2,
        fh(k) = k/ifco;
        if (lowhi == 2) fh(k) = 1.0/fh(k);end;%if
end;%k
if (lowhi == 2) %// i.e. high-pass filter
        for k=1:Ndiv2,
                if (k<(Ndiv2-trans))  fh(k) = fh(k+trans);end;%if
                if (k>=(Ndiv2-trans)) fh(k) = fh(Ndiv2);end;%if
        end;%k
end;%if
for k=Ndiv2+1:N,
        fh(k) = fh(N+1-k);
end;%//i.e. mirror N/2..N-1
for k=1:N,
        val=fh(k)*fh(k);
        fh(k) = exp(ndegree* log(val) );
        fh(k) = 1.0 / (1.0+(0.414*fh(k)));
end;%k
%=====================================================================
function [C]=ampl_fft2(A);
x=size(A,1);y=size(A,2);

%/*--------   2D - FFT  ---------*/
for i=1:x;
        for j=1:y;
```

```
                    if ( rem((i+j),2) == 1) A(i,j) = -A(i,j);
                    else A(i,j) =A(i,j);
                    end;%if
          end;%j
end;%i
C=fft2(A);
C=round(10.0 * log(abs(C)+1));
%=========================================================================
function [C]=Normalize_Matrix(A,tones)
max_A=max(max(A));
min_A=min(min(A));min_A=0;
C=(A-min_A)*(tones)/(max_A-min_A);%back to 0-(tones)
%=========================================================================
%//-----------------------------------------------------
function [C] = filt_in_freq2(A,image_depth,fh)
x=size(A,1);y=size(A,2);
C=fft2(A);

for i=1:x
          for j=1:y
                    ir = sqrt(j*j + i*i);
                    if (ir > x ) ir = x;end;%if
                    C(i,j)=C(i,j)*fh(round(ir));
          end;
end;

C=ifft2(C);

C=real(C);
for i=1:x;
          for j=1:y;
                    if(C(i,j)<0) C(i,j)=0;end;
          end;
end;
%=========================================================
```

```
%Lab Assignment:
%1/Type main program, routine for Butt_filt and  filt_in_freq2.
%2/use routines in graphics program Program_5_gr.m
%       for various fco and degree of filter and for low- or high- process images
%       and inspect spectra.

%Homework:
%       1/Form Ideal_filter function for high-pass, low-pass,
%                               band-pass and band-reject
%       2/Form exponential_filter function
```

%Graphics version of Program 5

```
%Filtering in Frequency Domain
function []=Program_5_gr();
clc;echo off;close all;
% 128x128 images
% BODY1.BMP,BODY2.BMP,BODY2.BMP,BODY3.BMP,BODY4.BMP, BODY5.BMP,
% CHEST.BMP,AA1a.BMP,Pelvis.bmp
% HEAD1.BMP,HEAD2.BMP,HEAD3.BMP,HEAD4.BMP,HEAD5.BMP,HEAD6.BMP,
% mam1.bmp,Mam2.bmp,Mam3.bmp,MAM10a.bmp
A=imread('..\Images 128x128\chest.BMP');%Pelvis.bmp HEAD6.BMP
A=double(A);
x=size(A,1);y=size(A,2);
sprintf('x= %f  y=%f',x,y)
image_depth=255;tones=256;
Normalize_Matrix(A,image_depth);
B=A;

tic
trans=0;
%-------------Filtering Parameters -------------------------
value=1;
switch value
        case 1
                lowhi=2;%1:low-pass, 2:high-pass
                ndegree=2;%degree of filter
                ifco=round(y*0.1);%cut-off frequency !!! change 0.2 between 0.2 and 0.9!!!
        case 2
                lowhi=2;%1:low-pass, 2:high-pass
                ndegree=2;%degree of filter
                ifco=round(y*0.2);%cut-off frequency !!! change 0.2 between 0.2 and 0.9!!!
                trans=round(y*0.15);%translation for not removing low freq info
end;
%-------------------------------------------------------------
fh=Butt_filt(x,lowhi,ndegree,ifco,trans);
```

```
B=filt_in_freq2(A,image_depth,fh);
%B=A+0.7*B; %Sharpen in freq domain
%===============PLOT IMAGES=====================================
colormap('gray');
subplot(2,3,1);imagesc(A);xlabel('Original Image');
axis equal;axis([1 size(A,2) 1 size(A,1)]);

subplot(2,3,2);plot(y*fh,'red');xlabel('Filter Function');
axis equal;axis([1 x 1 y]);

subplot(2,3,3);imagesc(B);xlabel('Processed Image');
axis equal;axis([1 size(B,2) 1 size(B,1)]);

A=ampl_fft2(A);
subplot(2,3,4);imagesc(A);xlabel('Or. Im. Spectrum');
axis equal;axis([1 size(A,2) 1 size(A,1)]);

B=ampl_fft2(B);
subplot(2,3,6);imagesc(B);xlabel('Processed Im. Spectrum');
axis equal;axis([1 size(B,2) 1 size(B,1)]);
toc
%//-------------------------------------------------
function [fh]=Butt_filt(N,lowhi,ndegree,ifco,trans)
%Put your code here
%=================================================================
function [C]=ampl_fft2(A);
%Put your code here
%=================================================================
function [C]=Normalize_Matrix(A,tones)
%Put your code here
%=================================================================
function [C] = filt_in_freq2(A,image_depth,fh)
%Put your code here
%=================================================================
```

%Program 6

```
% Read & plot & restore in frequency domain *.dat image
function []=Program_6();
clc;echo off;close all;
A=[30,31,12, 9,
        17,12,25,10,
        12, 8,17, 9,
        31,12,26,22];
A=double(A);B=A;
A=double(A);
x=size(A,1);y=size(A,2);
disp ('A');disp((A));
image_depth=31;tones=8;
B=A;
%-------------------------------------------------------
[B,fh]=degradeimage(A,image_depth);
B=Normalize_Matrix(B,image_depth);
disp('degraded B');disp(round(B));
disp('degradation filter');
disp('fh');for i=1:4;disp(fh(i));end;

value=3;
switch value
        case 1
                f_inv=Inverse_Filter(fh, x);
                C=filt_in_freq2(B,image_depth,f_inv);
        case 2
                f_wien=Wiener_Filter(fh, x);
                C=filt_in_freq2(B,image_depth,f_wien);
        case 3
                f_power=Power_Filter(fh, x);
                C=filt_in_freq2(B,image_depth,f_power)
end;
```

```
%-------------------------------------------------------------
B=Normalize_Matrix(B,image_depth);C=Normalize_Matrix(C,image_depth);
switch value
        case 1
                disp('Restored by inverse filter');
                for i=1:4;disp(f_inv(i));end;
        case 2
                disp('Restored by wiener filter');
                for i=1:4;disp(f_wien(i));end;
        case 3
                disp('Restored by Power filter');
                for i=1:4;disp(f_power(i));end;
        end;

        disp('Restored image');disp(round(C));

%=====================================================================
%//-------------------------------------------------
function [fh]=Butt_filt(N,lowhi,ndegree,ifco,trans)
for i=1:N; fh(i)=0;end;
%//---------Read in filter parameters--------
Ndiv2=N/2;
for k=1:Ndiv2,
        fh(k) = k/ifco;
        if (lowhi == 2) fh(k) = 1.0/fh(k);end;%if
end;%k
if (lowhi == 2) %// i.e. high-pass filter
        for k=1:Ndiv2,
                if (k<(Ndiv2-trans))  fh(k) = fh(k+trans);end;%if
                if (k>=(Ndiv2-trans)) fh(k) = fh(Ndiv2);end;%if
        end;%k
end;%if
for k=Ndiv2+1:N,
        fh(k) = fh(N+1-k);
```

```
end;%//i.e. mirror N/2..N-1
for k=1:N,
        val=fh(k)*fh(k);
        fh(k) = exp(ndegree* log(val) );
        fh(k) = 1.0 / (1.0+(0.414*fh(k)));
end;%k
%//-------------------------------------------------------
function [C] = filt_in_freq2(A,image_depth,fh)
x=size(A,1);y=size(A,2);
C=fft2(A);

for i=1:x
        for j=1:y
                ir = sqrt(j*j + i*i);
                if (ir > x ) ir = x;end;%if
                C(i,j)=C(i,j)*fh(round(ir));
        end;
end;

C=ifft2(C);

C=real(C);
for i=1:x;
        for j=1:y;
                if(C(i,j)<0) C(i,j)=0;end;
        end;
end;
%-------------------------------------------------------
%-----------------------------------------------------------------------
function [C,degradefilt]=degradeimage(A,image_depth)
x=size(A,1);y=size(A,2);
%//---------- degrade image --
C=A;
lowhi=1;
ndegree=2;%degree of filter
```

```
ifco=round(y*0.2);%cut-off frequency
trans=round(y*0.16);%translation for not removing low freq info

degradefilt=Butt_filt(x,lowhi,ndegree,ifco,trans);%//form degradation function i.e. Fh[k] in theory
C=filt_in_freq2(A,image_depth,degradefilt);%blur image

noise=30;    % additive Noise
R=randint(x,y,[0,noise])-noise/2;

C=C+(C.*R/100);
Normalize_Matrix(C,image_depth);
%---------------------------------------------------------------
function [C]=Normalize_Matrix(A,tones)
max_A=max(max(A));
min_A=min(min(A));min_A=0;
C=(A-min_A)*(tones)/(max_A-min_A);%back to 0-(tones)
%---------------------------------------------------------------
function [fhh]=Inverse_Filter(degradefilt, N)

for i=1:N
   fhh(i)=1.0/(degradefilt(i));
end;

%Otherwise
%fhh=degradefilt.^-1;
%---------------------------------------------------------------
function [fhh]=Wiener_Filter(degradefilt, N)
A=0.22;
for i=1:N,
   fhh(i) = degradefilt(i) / (degradefilt(i)*degradefilt(i) + A);
end;

%---------------------------------------------------------------
function [fhh]=Power_Filter(degradefilt, N)
A=0.172;
```

```
for i=1:N,
   fhh(i) = 1.0 / sqrt(degradefilt(i)*degradefilt(i) + A);
end;

%=================================================================

%Lab Assignment:
%1/Type main program, routines for inverse and wiener and power filters.
%2/use routines in graphics program Program_6_gr.m
%          for various parameters process images and inspect spectra.

%Homework:
%          Form Generalized Wiener Filter Function for various parameters process
%          images,inspect spectra, and determine best parameters (optically evaluated)
%   for various images.
```

%Graphics version of program 6

```
%Filtering in Frequency Domain
function []=Program_6_gr();
clc;echo off;close all;
% 128x128 images
% BODY1.BMP,BODY2.BMP,BODY2.BMP,BODY3.BMP,BODY4.BMP, BODY5.BMP,
% CHEST.BMP,AA1a.BMP,Pelvis.bmp
% HEAD1.BMP,HEAD2.BMP,HEAD3.BMP,HEAD4.BMP,HEAD5.BMP,HEAD6.BMP,
% mam1.bmp,Mam2.bmp,Mam3.bmp,MAM10a.bmp
A=imread('..\Images 128x128\mam10a.BMP');%Pelvis.bmp HEAD6.BMP
A=double(A);
x=size(A,1);y=size(A,2);
sprintf('x= %f  y=%f',x,y)
image_depth=255;tones=256;
Normalize_Matrix(A,image_depth);
B=A;

tic
[B,fh]=degradeimage(A,image_depth);

f_inv=Inverse_Filter(fh, x);
C=filt_in_freq2(B,image_depth,f_inv);

f_wien=Wiener_Filter(fh, x);
D=filt_in_freq2(B,image_depth,f_wien);

f_power=Power_Filter(fh, x);
E=filt_in_freq2(B,image_depth,f_power);

%==============PLOT IMAGES====================================

colormap('gray');
subplot(3,3,1);imagesc(A);xlabel('Original Image');
axis equal;axis([1 size(A,2) 1 size(A,1)]);colorbar
```

```
subplot(3,3,2);plot(y*fh,'red');xlabel('degradation filter ');
axis equal;axis([1 x 1 y]);

subplot(3,3,3);imagesc(B);xlabel('Degraded Image');
axis equal;axis([1 size(B,2) 1 size(B,1)]);

subplot(3,3,4);imagesc(C);xlabel('Inverse Filter');
axis equal;axis([1 size(C,2) 1 size(C,1)]);
  maxh=max(f_inv);minh=min(f_inv);f_inv=(f_inv-minh)*(1.0/(maxh-minh));%back to 0-1;
subplot(3,3,7);plot(y*f_inv,'red');xlabel('inverse filter ');
axis equal;axis([1 x 1 y]);

subplot(3,3,5);imagesc(D);xlabel('Wiener Filter');
axis equal;axis([1 size(D,2) 1 size(D,1)]);
 maxh=max(f_wien);minh=min(f_wien);f_wien=(f_wien-minh)*(1.0/(maxh-minh));%back to 0-1;
subplot(3,3,8);plot(y*f_wien,'red');xlabel('wiener filter ');
axis equal;axis([1 x 1 y]);


subplot(3,3,6);imagesc(E);xlabel('Power Filter');
axis equal;axis([1 size(E,2) 1 size(E,1)]);
 maxh=max(f_power);minh=min(f_power);f_power=(f_power-minh)*(1.0/(maxh-minh));%back to 0-1;
subplot(3,3,9);plot(y*f_power,'red');xlabel('power filter ');
axis equal;axis([1 x 1 y]);
toc
%//-------------------------------------------------
function [fh]=Butt_filt(N,lowhi,ndegree,ifco,trans)
%Put your code here

%//-------------------------------------------------------
function [C] = filt_in_freq2(A,image_depth,fh)
%Put your code here

%-------------------------------------------------------------------------
```

```
function [C,degradefilt]=degradeimage(A,image_depth)
%Put your code here

%---------------------------------------------------------------
function [C]=Normalize_Matrix(A,tones)
%Put your code here

%---------------------------------------------------------------
function [fhh]=Inverse_Filter(degradefilt, N)
%Put your code here

%---------------------------------------------------------------
function [fhh]=Wiener_Filter(degradefilt, N)
%Put your code here

%---------------------------------------------------------------
function [fhh]=Power_Filter(degradefilt, N)
%Put your code here

%=================================================================
```

%Graphics version of program 7

```
%test Tomographic reconstruction
clc;echo off;close all;
% 128x128 images
% BODY1.BMP,BODY2.BMP,BODY2.BMP,BODY3.BMP,BODY4.BMP, BODY5.BMP,
% CHEST.BMP,AA1a.BMP,Pelvis.bmp
% HEAD1.BMP,HEAD2.BMP,HEAD3.BMP,HEAD4.BMP,HEAD5.BMP,HEAD6.BMP,
% mam1.bmp,Mam2.bmp,Mam3.bmp,MAM10a.bmp
A=imread('..\Images 128x128\Pelvis.BMP');%Pelvis.bmp HEAD6.BMP
A=double(A);
x=size(A,1);y=size(A,2);
sprintf('x= %f  y=%f',x,y)
max_A=max(max(A));min_A=min(min(A));A=(A-min_A)*(255/(max_A-min_A));%back to 0-255
B=A;

% 256x256 images
% AA1a.BMP,AA2a.BMP,Angio1.BMP,Angio2.BMP,
% CY16.BMP,Lungs.BMP,Pelvis.bmp,
% mam1.bmp,Mam2.bmp,Mam3.bmp,MAM10a.bmp
% MRHEAD1.BMP,Mrhead2.BMP,Mrhead3.BMP,
% US1.BMP,US2.BMP,US3.BMP
% A=imread('C:\MATLAB\work\work_cav\Images 256x256\Angio1.BMP');
tic
C=A;D=A;E=A;

%==============PLOT IMAGES================================


N_proj=180;

theta=1:N_proj;
R=radon(A,theta);

   colormap('gray');
```

```
    subplot(2,2,1);imagesc(A);xlabel('Original Image');
    axis equal;axis([1 size(A,2) 1 size(A,1)]);

    for theta=1:N_proj,

       maxh=max(max(R(:,theta)));minh=min(min(R(:,theta)));h=(R(:,theta)-minh)*(1.0/(maxh-minh));%back to 0-1;
       subplot(2,2,3);plot(y*h,'red');xlabel('Projections');
                        axis equal;axis([1 x 1 y]);pause(0.01);
    end;

R = radon(A,1:N_proj);
I = iradon(R,1:N_proj,'nearest',    'Shepp-Logan' );


subplot(2,2,4);imagesc(R);xlabel('All Projections');
axis equal;axis([1 size(R,2) 1 size(R,1)]);

subplot(2,2,2);imagesc(I);xlabel('Reconstructed Image');
axis equal;axis([1 size(I,2) 1 size(I,1)]);
toc
```