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.

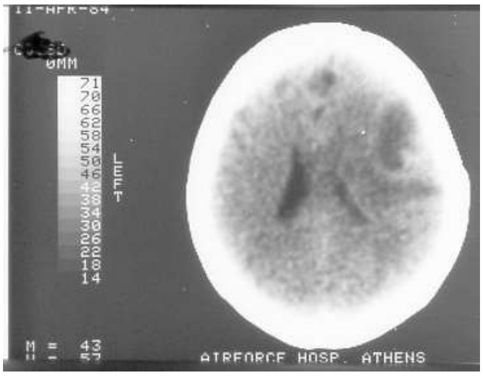


Image enhancement techniques may be distinguished into 2 major categories:

i/ Grey scale manipulation for increasing image contrast

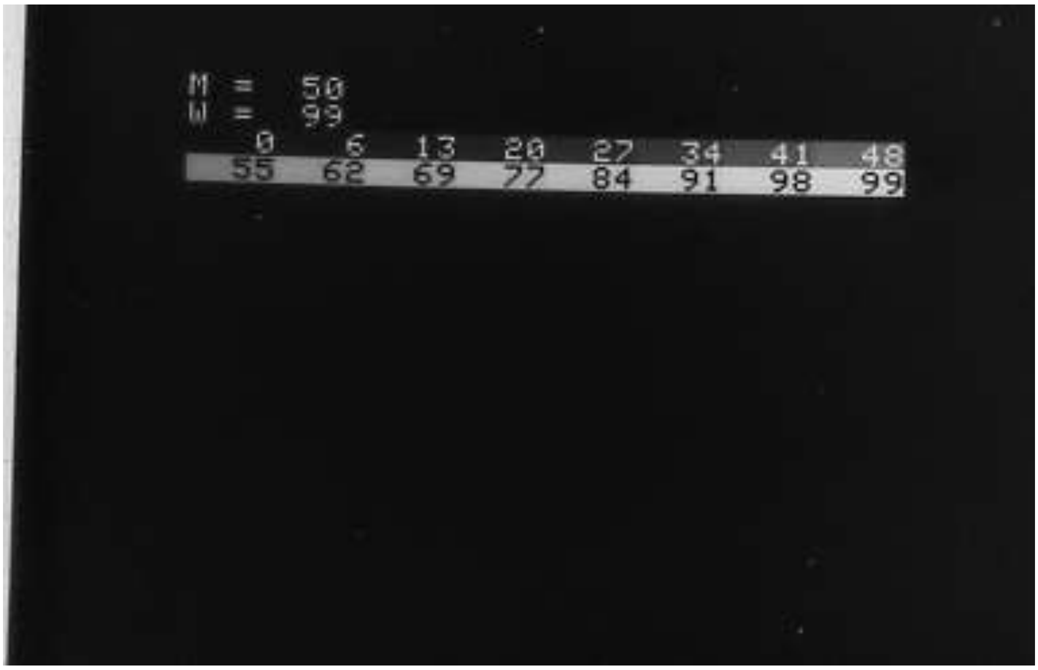
ii/ Image matrix manipulation for decreasing image noise and blur.
Image matrix manipulation techniques can be applied in either the spatial or the frequency domain, since the convolution process can be carried out in either domain. For this reason image enhancement will examined in both domains.

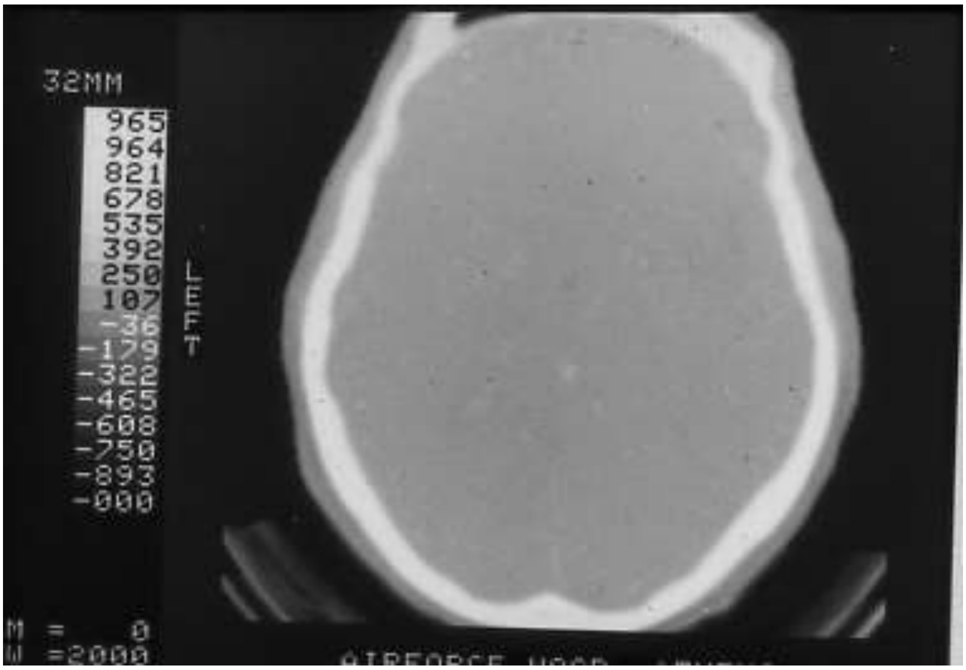
Grey Scale Manipulation Techniques: Windowing Techniques & Histogram Modification techniques

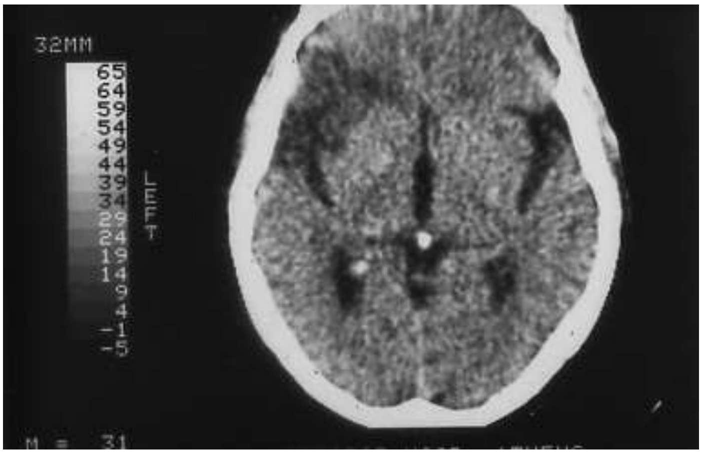
----

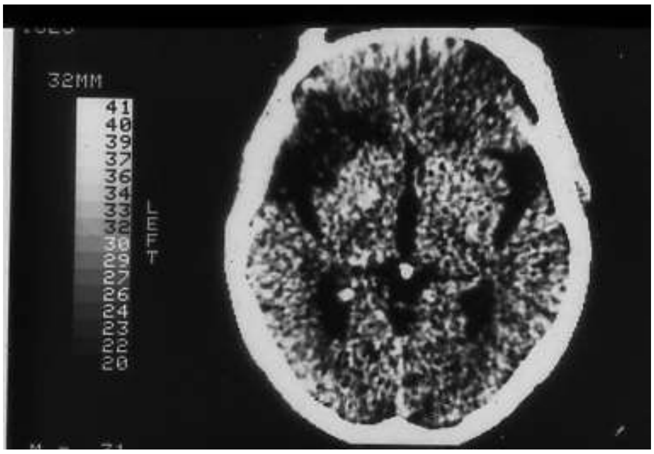
1/Windowing Techniques

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.

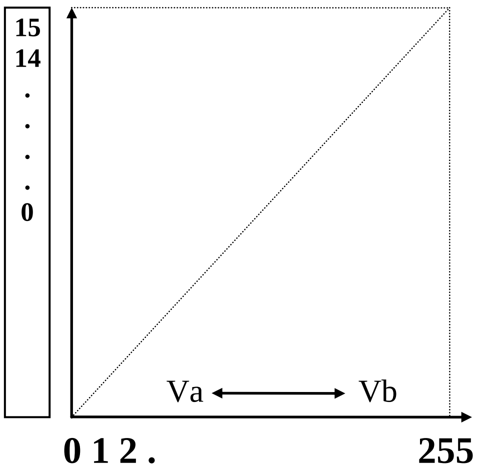



32MM
965
964
821
678
535
392
250
107
-36
-179
-322
-465
-608
-750
-893
-000
LEFT
M = 0
W =2000


32MM
65
64
59
54
49
44
39
34
29
24
19
14
9
4
-1
-5
LEFT

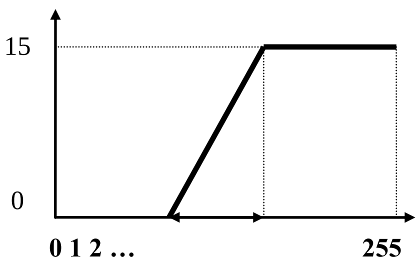

32MM
LEFT

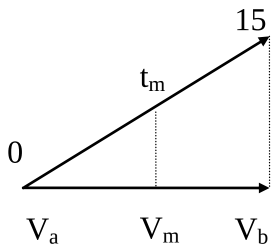
Simple Window



WW: Vb-Va

WC: $\dfrac{Vb+Va}{2}$

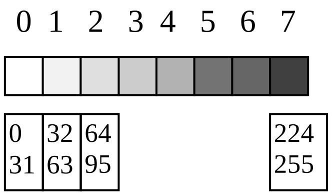


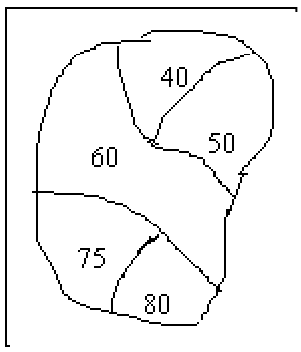


$$t_m = \frac{15-0}{V_b - V_a}(V_m - V_a) + 0$$

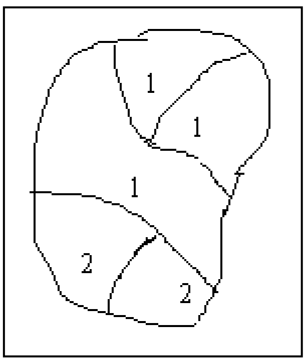
Simple Window Example

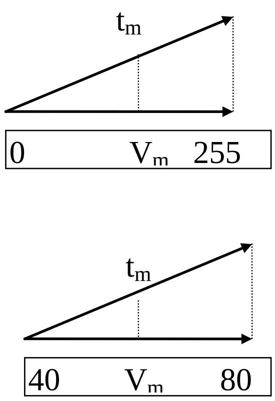
A/

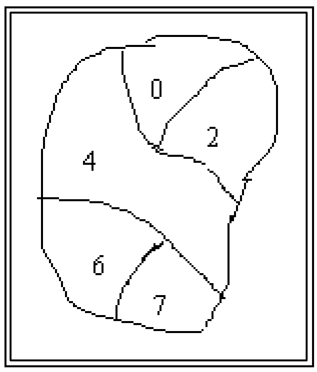




B/

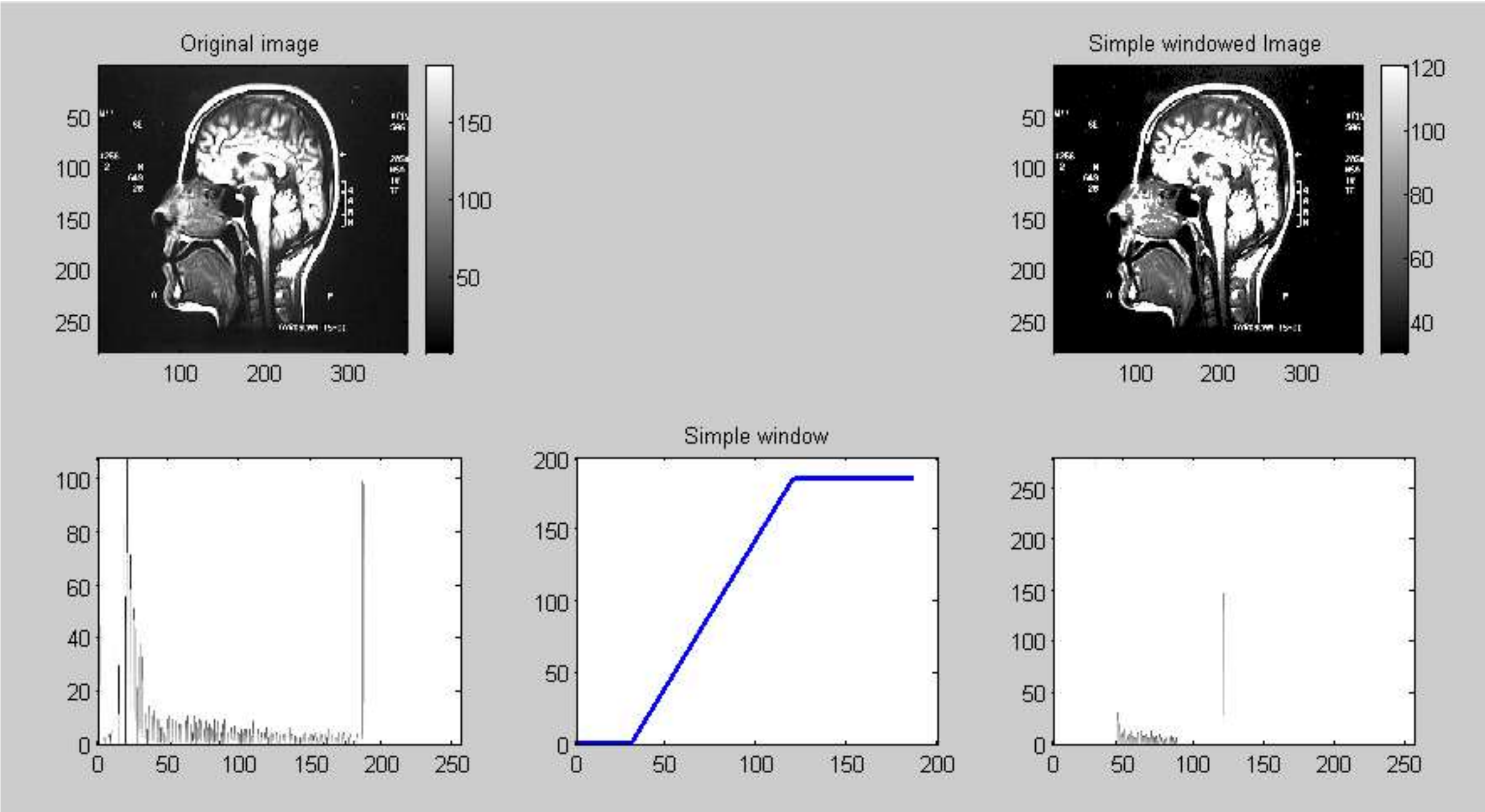






$$tm = \frac{7-0}{255-0}(Vm-0)+0$$

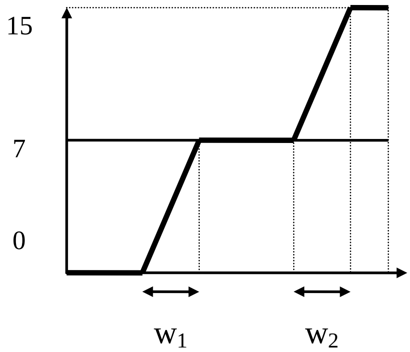
$$t_m = \frac{7-0}{80-40}\{Vm-40)+0$$

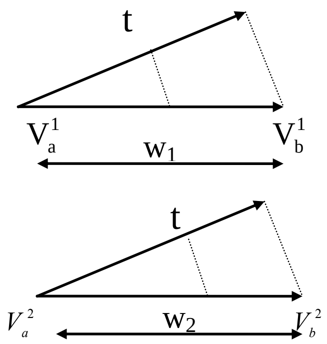

Original image
Simple windowed Image
Simple window

Double window

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 relatively good contrast

e.g. lungs (-1000 #CT to -400 #CT) και Spine (150 to 500 #CT)





Algorithm:

Scan image matrix, if value< V_a^1 , then t=0

if value within 1st window then use 1st triangle to compute t

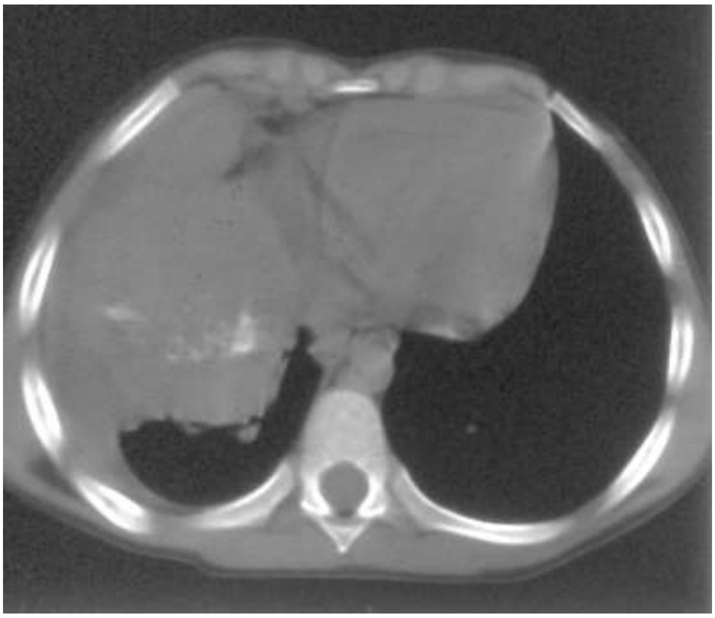
if value within 2nd window then use 2nd triangle to compute t

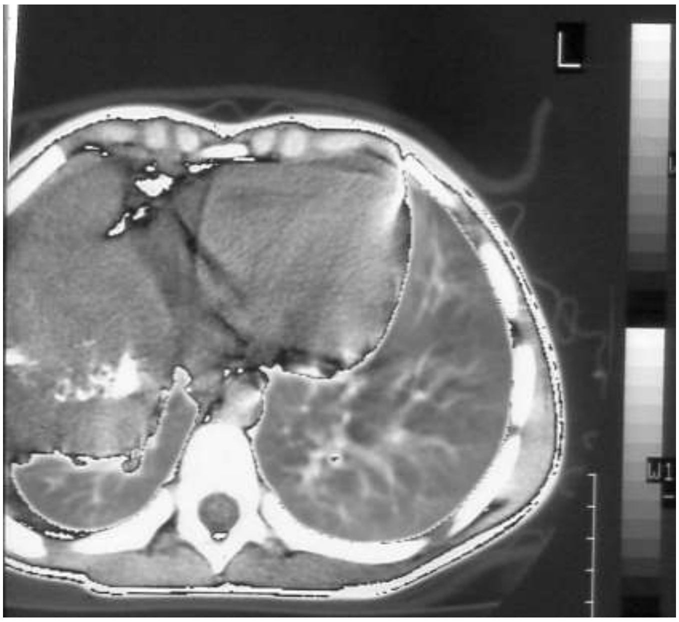
if value > V_b^2 , then t=15

if V_b^1 <value< V_a^2 , then t=7

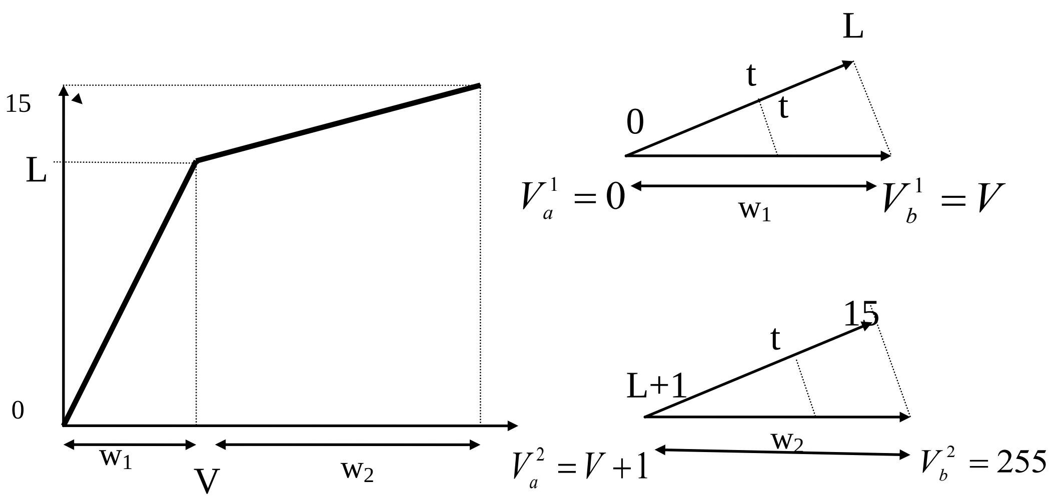
Beware of overlapping windows (split the overlap)


M68
M1=-561
W1=1000
M2= 60
W2= 200
-062-936-770-603-436-270-103 -41
-19 10 38 67 95 124 152 153
ANT




L

Broken window



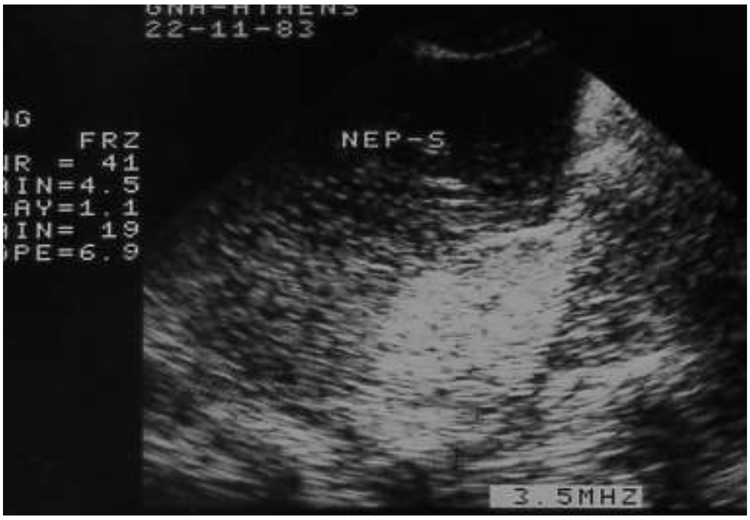
<u>Algorithm:</u>

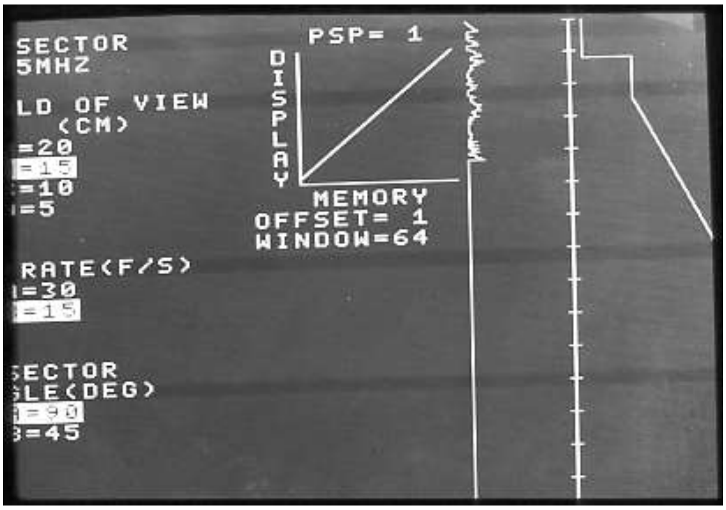
Let L,V be the greylevel and greytone

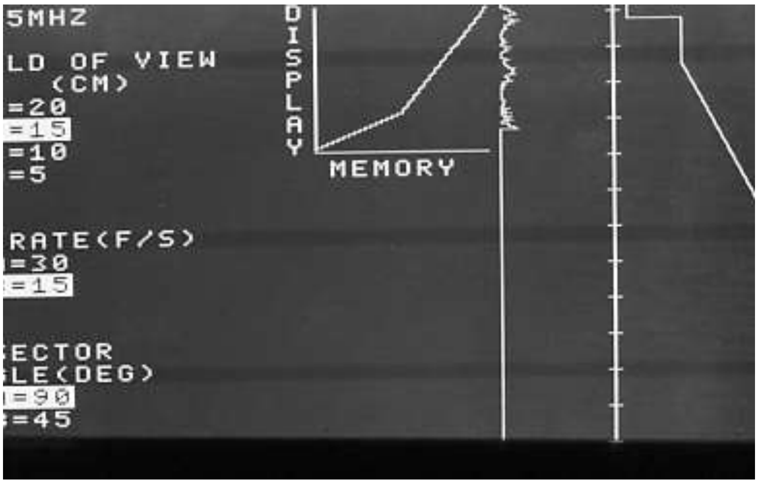
Scan image matrix,

if value< V, use 1st triangle to compute t

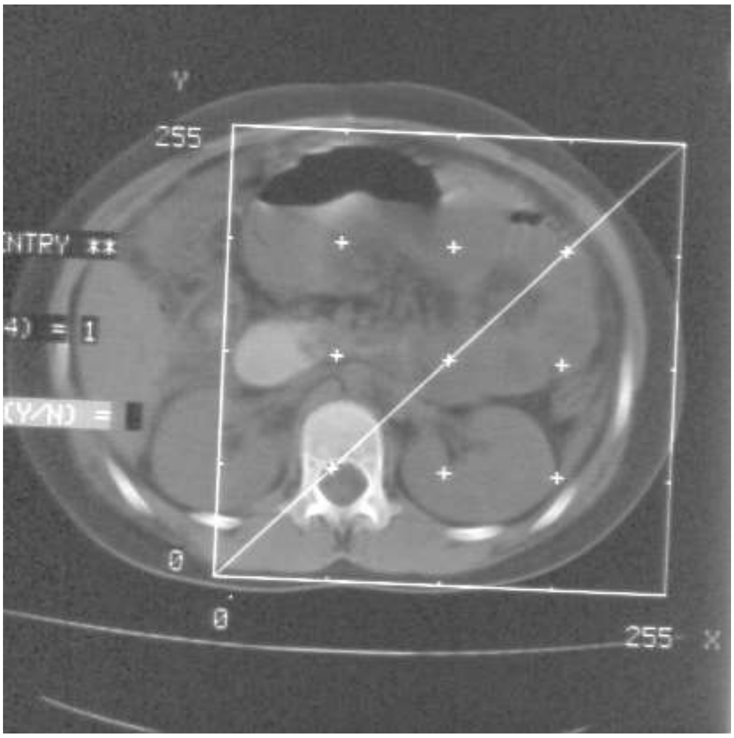
if value>V, use 2nd triangle to compute t

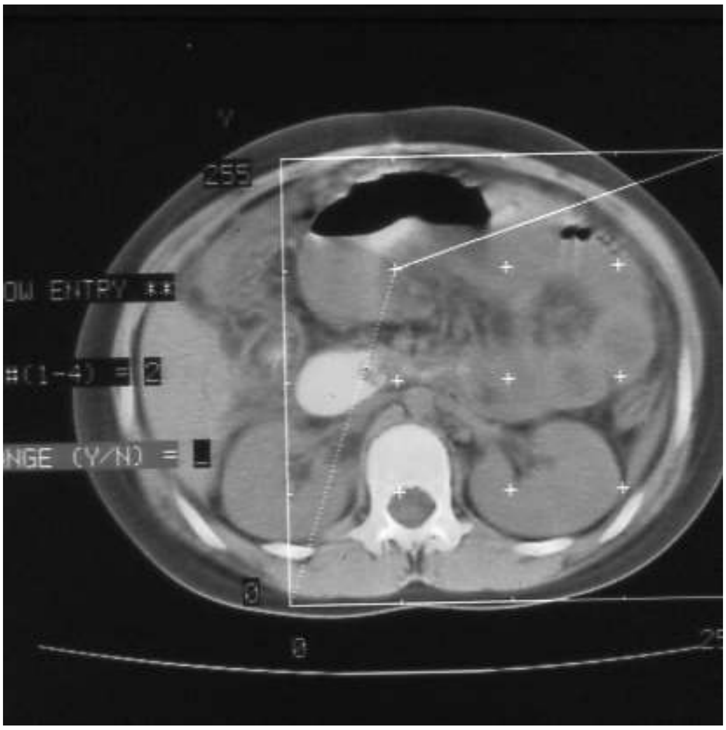

22-11-83
FRZ
NEP-S
3.5MHZ


SECTOR
PSP= 1
DISPLAY
MEMORY
OFFSET= 1
WINDOW=64
RATE(F/S)

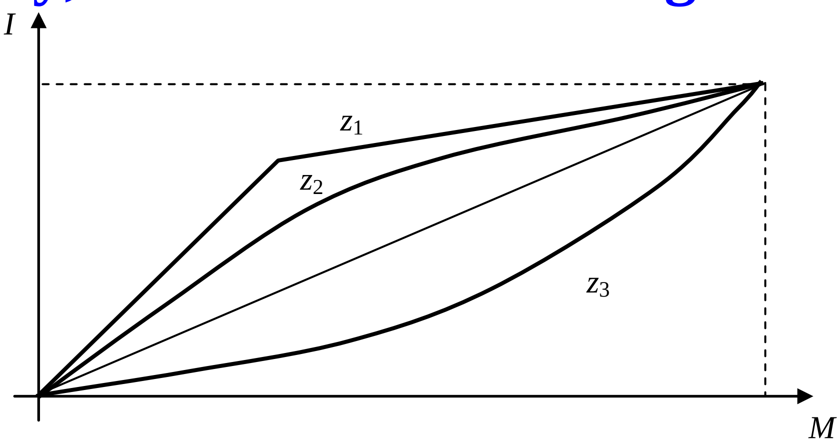

DISPLAY
MEMORY
RATE(F/S)


22-11-83
FRZ
NEP-S





Finally, non-linear windowing functions



```
%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;

disp('displayed image ');%disp(A);
B=My_plot(A,tones);% B holds grey-tone values only

value=2;
switch value
    case 1
        disp('simple window');
        WW=20;WL=20;
        B=My_simple_window(A,image_depth,tones,WW,WL);
    case 2
         disp('broken window');
        gray_val=5;im_val=21;
        B=My_broken_window(A,image_depth,tones,gray_val,im_val);%Have to construct it

    case 3
         disp('double window');
        ww1=10;ww2=10;wl1=10;wl2=25;
        B=My_double_window(A,image_depth,tones,ww1,ww2,wl1,wl2);%Have to construct it

    case 4
        disp('non-linear window');
```

```
        funct=1;
        B=My_non_linear_window(A,image_depth,tones,funct);%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;
```

```
function [C]=My_broken_window(A,image_depth,tones,gray_val,im_val)
C=A;
% put your code here
%-------------------------------------------------------------

function [C]=My_double_window(A,image_depth,tones,ww1,ww2,wl1,wl2)
C=A;
% put your code here

function [C]=My_non_linear_window(A,image_depth,tones,funct);%Have to construct it
C=A;
% put your code here


%==========================================================================
%Lab work:
%1/copy main program with  simple_window function, and compare results with ones calculated by hand
%2/Construct broken and double window and compare results with ones calculated by hand
%3/Construct non-linear window: use cosine, sine, exponential, quadratic, cubic functions.
%Monotonically increasing or decreasing.
%Note: Make sure function values range between 0 and 1 (have to find min-max values of each
%function and thus normalize to values %between 0 and 1). Then multiply each function by (tones-1)
%so that gray tones range between 0 and tones-1.
```

```
%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;
%Put your images folder into the same folder as your .m program
A=imread('Images\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=4;
switch value
    case 1
        disp('simple window');
        WW=100;WL=80;
        B=My_simple_window(A,image_depth,tones,WW,WL);
        %End of code for simple window here
    case 2
        disp('broken window');
        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
         disp('double window');
        ww1=50;ww2=50;wl1=50;wl2=220;
        B=My_double_window(A,image_depth,tones,ww1,ww2,wl1,wl2);%Have to construct it
    case 4
        disp('non-linear window');
        funct=1;
        B=My_non_linear_window(A,image_depth,tones,funct);%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)]);
    case 4
        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;
% put your code here
%---------------------------------------------------------------

function [C]=My_broken_window(A,image_depth,tones,gray_val,im_val)
C=A;
% put your code here
%---------------------------------------------------------------
```

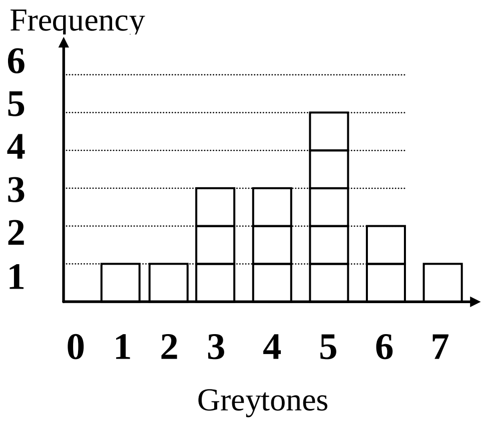
```
function [C]=My_double_window(A,image_depth,tones,ww1,ww2,wl1,wl2)
C=A;
% put your code here

function [C]=My_non_linear_window(A,image_depth,tones,funct);%Have to construct it
C=A;
% put your code here
```

Histogram Modification Techniques.

Image Histogram:

3	3	5	1
6	5	6	5
7	4	4	4
2	5	5	3

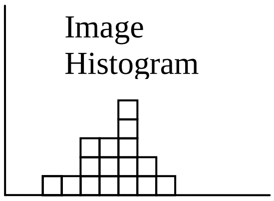
Displayed Image



Histogram equalization:
In the resulting image all greytones have the same number of pixels.

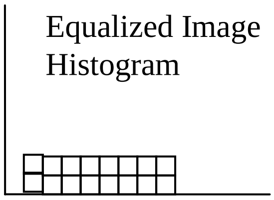
3	3	5	1
6	5	6	5
7	4	4	4
2	5	5	3

Original Image



1	1	4	0
6	4	7	5
7	2	3	3
0	5	6	2

Equalized Image



t= [0,1,2,3,4,5,6,7]
h(n)=[0,1,1,3,3,5,2,1]
q(n)=[2,2,2,2,2,2,2,2]

Αλγόριθμος:

$q(0)=\frac{1}{16}\ h(1)+\frac{1}{16}\ h(2)$ Compression

$q(1)=\frac{2}{16}\ h(3)$ Spread

$q(2)=\frac{1}{16}\ h(3)+\frac{1}{16}\ h(4)$ Spread

$q(3)=\frac{2}{16}\ h(4)$ Spread

$q(4)=\frac{2}{16}\ h(5)$ Spread

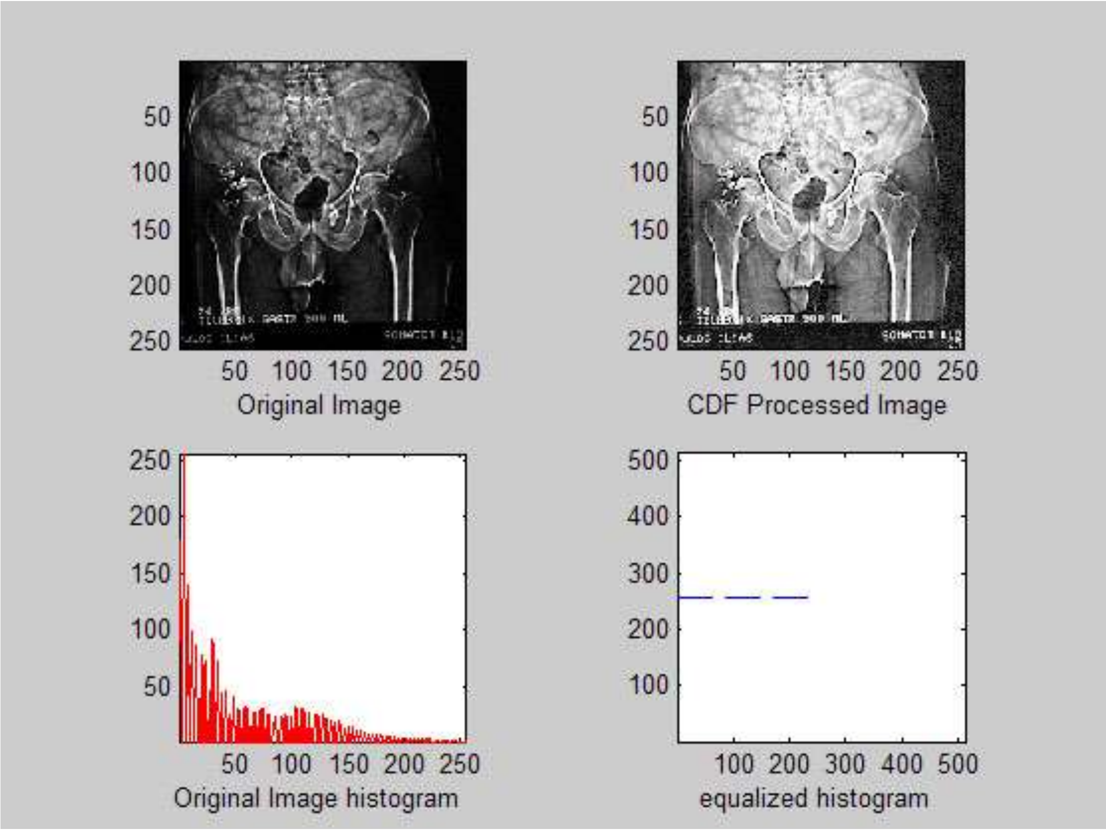
$q(5)=\frac{2}{16}\ h(5)$ Spread

$q(6)=\frac{1}{16}\ h(5)+\frac{1}{16}\ h(6)$

$q(7)=\frac{1}{16}\ h(6)+\frac{1}{16}\ h(7)$ Compression

Choice of Pixels:

- At random, as we scan the image
- Other more time consuming methods



Original Image
CDF Processed Image
Original Image histogram
equalized histogram

CDF: Cumulative Distribution Function:

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

- Fast
- Effective with good results

Levels	h(i)	CDFh(i)	q(i)	CDFq(i)	
0	0(0)	0	2	2	C
1	1(0)	1	2	4	
2	1(0)	2	2	6	S
3	3(1)	5	2	8	
4	3(3)	8	2	10	S
5	5(5)	13	2	12	
6	2(6)	15	2	14	
7	1(7)	16	2	16	

$$t = (\text{int})\left(\frac{CDFh(i)}{q(i)}\right)$$ approximate function for computer application

3	3	5	1
6	5	6	5
7	4	4	4
2	5	5	3

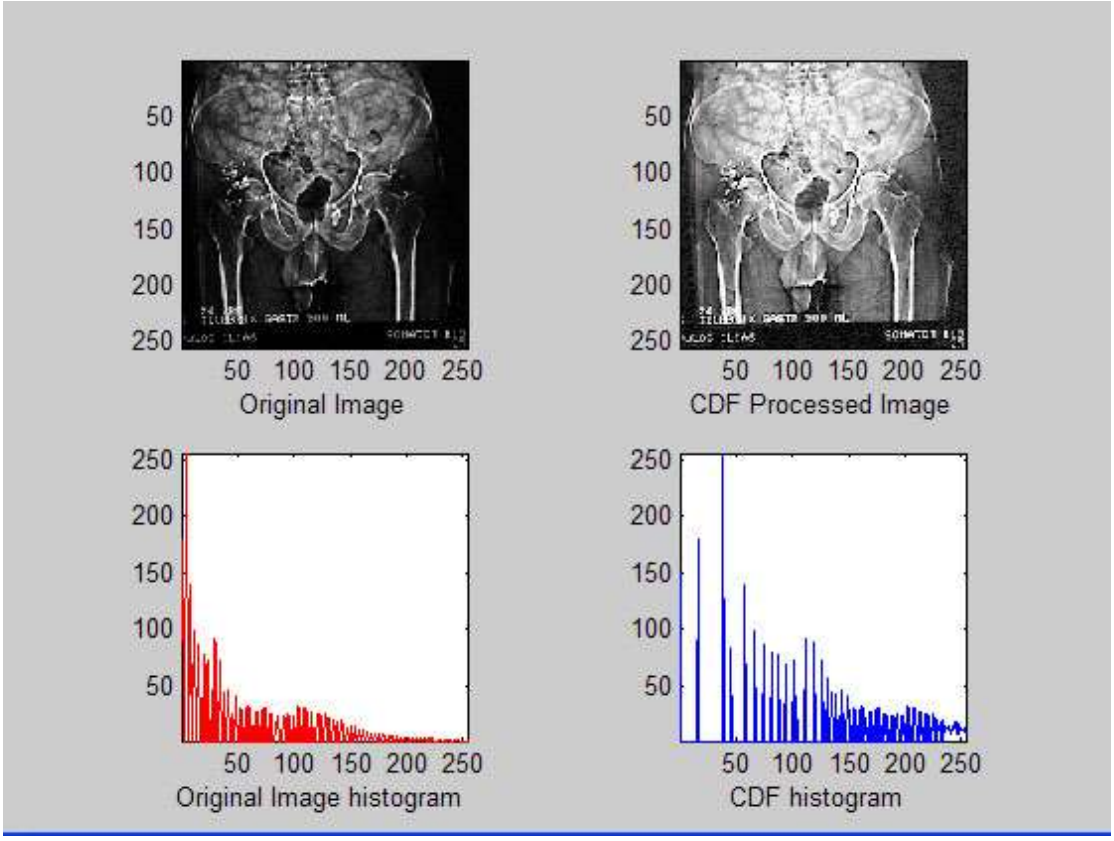
Original Image

1	1	4	0
6	4	6	4
7	2	2	2
0	4	4	1

CDF- Image

After formula application CDF-image

1	1	4	0
6	4	6	4
7	2	2	2
0	4	4	1


Original Image
CDF Processed Image
Original Image histogram
CDF histogram

```
%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);

B=CDF(A,tones);%Have to construct it

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 [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;
%----------------------------------------------------------

function [B]=CDF(A,tones)
B=A;
```

%===

%Lab work:

%1/Construct your own routine CDF using formula $t = (\text{int})\left(\frac{CDFh(i)}{q(i)}\right)$ page 21**).**

%Use it in the graphics version of the program.

```
%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\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=1;
switch value
case 1
        B=My_CDF(A,tones);%Have to construct it
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)]);
end

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

%----------------------------------------------------------
function [B]=My_CDF(A,tones)
```