Scilab Textbook Companion for Digital Image Processing by R. C. Gonzalez and R. E. Woods[1]

Created by
Pinkesh Vasantbhai Patel
DIGITAL IMAGE PROCESSING
Electronics Engineering
DHARMSINH DESAI UNIVERSITY
Cross-Checked by
Scilab TBC Team

July 2, 2020

[1]Funded by a grant from the National Mission on Education through ICT, http://spoken-tutorial.org/NMEICT-Intro. This Textbook Companion and Scilab codes written in it can be downloaded from the "Textbook Companion Project" section at the website http://scilab.in

Book Description

Title: Digital Image Processing

Author: R. C. Gonzalez and R. E. Woods

Publisher: Pearson, New Delhi

Edition: 3

Year: 2009

ISBN: 978-81-317-2695-2

Scilab numbering policy used in this document and the relation to the above book.

Exa Example (Solved example)

Eqn Equation (Particular equation of the above book)

AP Appendix to Example(Scilab Code that is an Appednix to a particular Example of the above book)

For example, Exa 3.51 means solved example 3.51 of this book. Sec 2.3 means a scilab code whose theory is explained in Section 2.3 of the book.

Contents

List of Scilab Codes

List of Figures

Chapter 2

Digital Image Fundamental

check Appendix **??** for dependency:

`Ex2_2.tif`

Scilab code Exa 2.2 `Illustration of the Effects of Reducing Image Spatial Resoluti`

```
//Ex2_2
//Illustration of the Effects of Reducing Image Spatial Resolution
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).
gray=imread("Ex2_2.tif");
```

```
figure,ShowImage(gray,'Gray Image');
title('Original Image (1250 DPI)');
[M,N]=size(gray);
a1=imresize(gray,[443 337],'nearest');
figure,ShowImage(a1,'Resize Image');
title('Resize Image (300 DPI)');

a2=imresize(gray,[886 675],'nearest');
figure,ShowImage(a2,'Resize Image');
title('Resize Image (150 DPI)');

a3=imresize(gray,[213 162],'nearest');
figure,ShowImage(a3,'Resize Image');
title('Resize Image (72 DPI)');
```

check Appendix ?? for dependency:

`Ex2_3.png`

Scilab code Exa 2.3 `Typical Effects of Varying the Number of Intensity Levels in a`

```
//Ex2_3
//Typical Effects of Varying the Number of Intensity Levels in a digital Image
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

clc;
close;
clear;
```

```
xdel(winsid())//to close all currently open figure(s).
gray=rgb2gray(imread("Ex2_3.png"));
figure,ShowImage(gray,'Gray Image');
title('Original Image');
[nr nc]=size(gray);
b=gray;
level=[128 64 32 16 8 4 2];
for x=1:length(level)
k=level(x);
for y=1:k
for i=1:nr
    for j=1:nc
        if(gray(i,j)>=((255/k)*(y-1)) & gray(i,j)<((255/k)*y))
            b(i,j)=((255/k)*(y-1))+((255/k)/2);
         end
    end
end
end
figure,ShowImage(b,'OutPut Image');
title('Image With Less Number of Gray Level');
end
```

check Appendix ?? for dependency:

`Ex2_4.tif`

Scilab code Exa 2.4 `Comparision of Interpolation Approaches for Image Shrinking an`

```
//Ex2_4
//Comparision of Interpolation Approaches for Image Shrinking and Zooming
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
```

```
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).
gray=imread("Ex2_4.tif");
figure,ShowImage(gray,'Gray Image');
title('Original Image (1250 DPI)');
[M,N]=size(gray);

a2=imresize(gray,[213 162],'nearest');  //nearest neigubour Interpolation
figure,ShowImage(a2,'Resize Image');
title('Resize Image (72 DPI) nearest neigubour Interpolation');
a2=imresize(gray,[213 162],'bilinear');   ///bilinear Interpolation
figure,ShowImage(a2,'Resize Image');
title('Resize Image (72 DPI) with bilinear Interpolation');
a2=imresize(gray,[213 162],'bicubic');   //bicubic Interpolation
figure,ShowImage(a2,'Resize Image');
title('Resize Image (72 DPI) with bicubic Interpolation');

a3=imresize(gray,[886 675],'nearest');   //nearest neigubour Interpolation
figure,ShowImage(a3,'Resize Image');
title('Resize Image (150 DPI) with nearest neigubour Interpolation');
a3=imresize(gray,[886 675],'nearest');   ///bilinear Interpolation
```

```
figure,ShowImage(a3,'Resize Image');
title('Resize Image (150 DPI) with bilinear Interpolation');
a3=imresize(gray,[886 675],'nearest');    //bicubic Interpolation
figure,ShowImage(a3,'Resize Image');
title('Resize Image (150 DPI) with bicubic Interpolation');
```

check Appendix ?? for dependency:

`Ex2_5.tif`

Scilab code Exa 2.5 `Addition of Noisy Images for Noise Reduction`

```
//Ex2_5
//Addition of Noisy Images for Noise Reduction
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).
gray=imread("Ex2_5.tif");
//gray=rgb2gray(a);
gray=im2double(gray);

figure,ShowImage(gray,'Gray Image');
title('Original Image');
```

```
[nr nc]=size(gray);
noise_image=gray;

out_image=double(zeros(nr,nc));
level=[5 10 20 50 100];
for i=1:length(level)
No=level(i);
disp(No);
for k=1:No
    noisy_image=imnoise(noise_image,'gaussian',0,0.02);
//     figure,ShowImage(noisy_image,'Image corrupted by salt & pepper noise');//ShowImage() is used to sho     w image, figure is command to view images in separate window.
//     title('Image corrupted by  Gaussian noise');// title() is used for providing  a title to an image.
//     disp(size(noise_image));
    out_image=imadd(out_image,noisy_image);
end
out_image=out_image/No;
out_image=mat2gray(out_image);

figure,ShowImage(out_image,'Image Recoverd from the Noise');//ShowImage() is used to show image, figure is command to view images in separate window.
    title('Image Recoverd from the Noise');//title() is used for providing  a title to an image.
//Recoverd_Image=0.5*out_image.^0.15;//Gamma Transformation
//figure,ShowImage(Recoverd_Image,'Recoverd Image after Gamma Transormation');//ShowImage() is used to show image, figure is command to view images in separate window.
//title('Image Recoverd from the Noise');//title() is used for providing  a title to an image.
```

```
end
```

---

check Appendix ?? for dependency:

Ex2_6.tif

Scilab code Exa 2.6 Image Subtraction for Enhancing differences

```
//Ex2_6
//Image Subtraction for Enhancing differences
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E.
   Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s
   ).
gray=imread("Ex2_6.tif");
gray=imresize(gray,0.25,'bicubic'); //Resize the
   Image with bicubic Interpolation
figure,ShowImage(gray,'Gray Image');
title('Original Image');
[nr nc]=size(gray);
for i=1:8
    c(:,:,i)=mtlb_logical(bitget(gray,9-i)); //
       Separate bit Planes from the Gray Scale Image

end
c(:,:,8)=zeros(nr,nc);   // Set Zeros to LSB
//c(:,:,7)=zeros(nr,nc);   // Set Zeros to LSB
```

```

for i=1:nr
    for j=1:nc
        mask(i,j)=(2^7)*c(i,j,1)+(2^6)*c(i,j,2)+(2^5)*c(i,j,3)+(2^4)*c(i,j,4)+(2^3)*c(i,j,5)+(2^2)*c(i,j,6)+(2^1)*c(i,j,7)+(2^0)*c(i,j,8);
    end
end
figure;ShowImage(mask,'Modified Image');
title('Image Obtained by Setting Zeros to LSB ');
mask=uint8(mask);  //Convert the Image to uint8 Data Type
Diff_image=imsubtract(gray,mask);  //Subtract two Images
Diff_image=mat2gray(Diff_image);
figure;ShowImage(Diff_image,'Modified Image');
title('Difference of two images');
```

check Appendix ?? for dependency:

`Ex2_7.tif`

check Appendix ?? for dependency:

`Ex2_7_1.tif`

check Appendix ?? for dependency:

`Ex2_7_2.tif`

check Appendix ?? for dependency:

`Ex2_7_3.tif`

Scilab code Exa 2.7 `Image Multiplication for Shadding Correction`

```
//Ex2_7
// Image Multiplication for Shadding Correction.
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).

//////////////////  Image Division  /////////////////////
gray=imread("Ex2_7.tif");
shade=imread("Ex2_7_1.tif");
gray=im2double(imresize(gray,0.5,'bicubic')); // Resize the Image with Bicubic Interpolation
shade=im2double(imresize(shade,0.5,'bicubic')); // Resize the Image with Bicubic Interpolation
figure,ShowImage(gray,'Gray Image');
title('Original Image');
figure,ShowImage(shade,'Sahde Image');
title('Shading Pattern Image');
[nr nc]=size(gray);
Enhance_image=imdivide(gray,shade);
Enhance_image=mat2gray(Enhance_image);
figure,ShowImage(Enhance_image,'Enhance Image');
title('Enhance Image after Shading Correction');

//////////////////  Image Multiplication  /////////////////////
gray=imread("Ex2_7_2.tif");
mask=imread("Ex2_7_3.tif");
```

```
gray=im2double(imresize(gray,0.5,'bicubic')); //
    Resize the Image with Bicubic Interpolation
mask=im2double(imresize(mask,0.5,'bicubic')); //
    Resize the Image with Bicubic Interpolation
figure,ShowImage(gray,'Gray Image');
title('Original Image');
figure,ShowImage(mask,'mask Image');
title('mask Pattern Image(ROI)');
[nr nc]=size(gray);
Enhance_image=immultiply(gray,mask);
Enhance_image=mat2gray(Enhance_image);
figure,ShowImage(Enhance_image,'Enhance Image');
title('ROI Selection');
```

check Appendix ?? for dependency:

`Ex2_12.tif`

check Appendix ?? for dependency:

`Ex2_12_1.tif`

check Appendix ?? for dependency:

`Ex2_12_2.tif`

Scilab code Exa 2.12 `Standard Deviation`

```
//Ex2_12
// Standard Deviation
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E.
    Woods
```

```

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).

////////////////// Image Rotation /////////////////////
gray1=imread("Ex2_12.tif");
gray1=im2double(gray1); //Convert the data type into double range
figure,ShowImage(gray1,'Gray Image');
title('Original Image');
gray2=imread("Ex2_12_1.tif");
gray2=im2double(gray2); ///Convert the data type into double range
figure,ShowImage(gray2,'Gray Image');
title('Original Image');
gray3=imread("Ex2_12_2.tif");
gray3=im2double(gray3); //Convert the data type into double range
figure,ShowImage(gray3,'Gray Image');
title('Original Image');

y=variance(gray1);  // calculate variance
disp('Variance of Image 1:')
disp(y);
y=variance(gray2);   // calculate variance
disp('Variance of Image 2:')
disp(y);
y=variance(gray3);    // calculate variance
disp('Variance of Image 3:')
disp(y);
```

Chapter 3

Intensity Transformation and Spatial Filtering

check Appendix ?? for dependency:

`Ex3_1.tif`

Scilab code Exa 3.1 `Gamma Intensity transformation`

```
//Ex3_1
// Gamma Intensity transformation
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E.
   Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s
   ).
```

```
gray=imread("Ex3_1.tif");
figure,ShowImage(gray,'Gray Image');
title('Original Image','color','blue','fontsize',4);
[M,N]=size(gray);
c=1;
gamma=[0.6 0.4 0.3];
for i=1:length(gamma)
    b=c.*(gray).^gamma(i); //Gamma transformation
    b=mat2gray(b);
    figure,ShowImage(b,'Gray Image');
    title('Enhance Image after Gamma transformation','color','blue','fontsize',4);
end
```

check Appendix ?? for dependency:

`Ex3_2.tif`

Scilab code Exa 3.2 `Illustration of Power Law Transformation`

```
//Ex3_2
// Illustration of Power Law Transformation
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).
gray=imread("Ex3_2.tif");
```

```
gray=im2double(gray);
figure,ShowImage(gray,'Gray Image');
title('Original Image','color','blue','fontsize',4);
[M,N]=size(gray);
c=1;
gamma=[3 4 5];
for i=1:length(gamma)
    b=c.*(gray).^gamma(i); //Gamma transformation
    b=mat2gray(b);
    figure,ShowImage(b,'Gray Image');
    title('Enhance Image after Gamma transformation'
       ,'color','blue','fontsize',4);
end
```

check Appendix ?? for dependency:

`Ex3_3.tif`

Scilab code Exa 3.3 `Intensity Level Slicing`

```
//Ex3_3
// Intensity Level Slicing
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E.
   Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s
   ).
gray=imread("Ex3_3.tif");
```

```
//gray=im2double(gray);
figure,ShowImage(gray,'Gray Image');
title('Original Image','color','blue','fontsize',4);
[M,N]=size(gray);
A=145;
B=245;
for i=1:M
    for j=1:N
        if(gray(i,j)>A & gray(i,j)<=B)
        b(i,j)=255;
        c(i,j)=255;
    else
        b(i,j)=0;
        c(i,j)=gray(i,j);
    end
    end
end
figure,ShowImage(b,'Gray Image');
title('Image after Intensity Slicing transformation'
   ,'color','blue','fontsize',4);

figure,ShowImage(c,'Gray Image');
title('Image after Intensity Slicing transformation(
   Linear)','color','blue','fontsize',4);
```

Scilab code Exa 3.5 A Simple Illustration of Histogram Equalization

```
//Ex3_5
// A Simple Illustration of Histogram Equalization
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E.
```

```
Woods



clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).
r=[0 1 2 3 4 5 6 7];  // Intensity
nk=[790 1023 850 656 329 245 122 81];  //Total No. of Pixels having Same Intensity
M=sum(nk);
probability_r=nk/M;  // Probability calculation
for i=1:length(r)
    sum_1=0;
    for j=1:i
        sum_1=sum_1+probability_r(j);
    end
    s(i)=max(r)*sum_1;
end
s=round(s); // Rounding Approach
disp(s);
[nr nc]=size(s);


for i=0:max(r)
    [row col]=find(s==i);
    len=length(row);
    if(len>0)
        sum_1=0;
        for j=1:len
            sum_1=sum_1+probability_r(row(j)); // Addition of Probability of same intensity after Equalization
        end
        Hist_equ(i+1)=sum_1;
    else
```

```
        Hist_equ(i+1)=0;
    end
end
disp('Histogram Equalization:')
disp(Hist_equ);

figure,bar(r,probability_r,0.1);
title('Original Histogram','color','blue','fontsize'
    ,4);
xlabel('Intensity');
ylabel('Probability of Same Intensity');

figure,bar(r,Hist_equ,0.1);
title('Equalized Histogram','color','blue','fontsize
    ',4);
xlabel('Intensity');
ylabel('Probability of Same Intensity');
```

check Appendix ?? for dependency:

`Ex3_6.tif`

check Appendix ?? for dependency:

`Ex3_6_1.tif`

check Appendix ?? for dependency:

`Ex3_6_2.tif`

Scilab code Exa 3.6 `Histogram Equalization with probability calculation`

```
//Ex3_6
// Histogram Equalization
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
```

```
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods



clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).
a=imread('Ex3_6.tif');
[P Q]=size(a);
[count cell]=imhist(a);
figure,ShowImage(a,'Original Image');
title('Original Image','color','blue','fontsize',4);

r=cell';  // Transpose of matrix
nk=round(count)';    // Transpose of matrix

//r=[0 1 2 3 4 5 6 7];   // Intensity
//nk=[790 1023 850 656 329 245 122 81];   //Total No. of Pixels having Same Intensity
M=sum(nk);
probeblity_r=nk/M;   // Probablity calculation
for i=1:length(r)
    sum_1=0;
    for j=1:i
        sum_1=sum_1+probeblity_r(j);
    end
    s(i)=max(r)*sum_1;
end
s=round(s); // Rounding Approach
disp(s);
[nr nc]=size(s);
temp=s';     // Transpose of matrix
for i=1:P      // Intensity Replacement in Original
```

```
    Image
    for j=1:Q
        b(i,j)=temp(double(a(i,j))+1);
    end
end
figure,ShowImage(b,'histogram Equlized Image');
title('histogram Equlized Image','color','blue','fontsize',4);

for i=0:max(r)
    [row col]=find(s==i);
    len=length(row);
    if(len>0)
        sum_1=0;
        for j=1:len
            sum_1=sum_1+probeblity_r(row(j)); // Addition of Probability of same intensity after Equqlization
        end
        Hist_equ(i+1)=sum_1;
    else
        Hist_equ(i+1)=0;
    end
end
disp('Histogram Equlization:')
disp(Hist_equ);

figure,bar(r,Hist_equ,0.1);
title('Equalized Histogram','color','blue','fontsize',4);
xlabel('Intensity');
ylabel('Probability of Same Intensity');
```

Scilab code Exa 3.8 `Histogram Specification`

```
//Ex3_8
// Histogram Specification
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods



clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).
r=[0 1 2 3 4 5 6 7];  // Intensity
nk=[790 1023 850 656 329 245 122 81];  //Total No. of Pixels having Same Intensity
probability_Specified=[0.00 0.00 0.00 0.15 0.20 0.30 0.20 0.15]; // Histogram Specification
M=sum(nk);
probability_r=nk/M;  // Probablity calculation
for i=1:length(r)
    sum_1=0;
    sum_2=0;
    for j=1:i
        sum_1=sum_1+probability_r(j);          // Histogram Equalization
        sum_2=sum_2+probability_Specified(j); // Histogram Specification
    end
    s(i)=max(r)*sum_1;
    G(i)=max(r)*sum_2;
end

s=round(s); // Rounding Approach
```

```
disp('Histogram Equalization:')
disp(s);
G=round(G); // Rounding Approach
disp('Histogram Specification G(Zq):')
disp(G);
[nr nc]=size(s);

for i=0:max(r)
    [row col]=find(G(i+1)==s);
    len=length(row);
    if(len>0)
        sum_1=0;
        for j=1:len
            sum_1=sum_1+probability_r(row(j));
        end
        Hist_Spe(i+1)=sum_1;
    end
    if(len==0)
        if(G(i+1)==0)
        Hist_Spe(i+1)=0;
    else
        Hist_Spe(i+1)=probability_r(G(i+1));
    end
end
end
disp('Histogram After Matching:')
disp(Hist_Spe);

figure,bar(r,probability_r,0.1);
title('Original Histrogram','color','blue','fontsize
   ',4);
xlabel('Intensity');
ylabel('Probability of Same Intensity');

figure,bar(r,probability_Specified,0.1);
title('Specified Histogram','color','blue','fontsize
   ',4);
xlabel('Intensity');
```

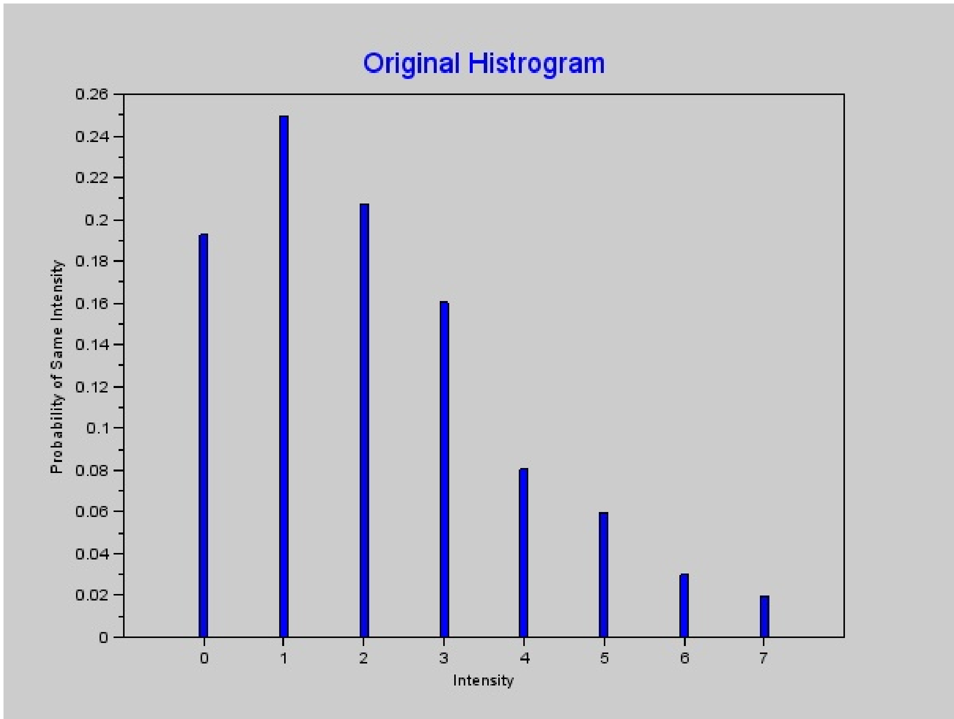


Figure 3.1: Histogram Specification

```
ylabel('Probability of Same Intensity');

figure,bar(r,Hist_Spe,0.1);
title('Histogram matching','color','blue','fontsize'
    ,4);
xlabel('Intensity');
ylabel('Probability of Same Intensity');
```

check Appendix ?? for dependency:

`Ex3_10.tif`

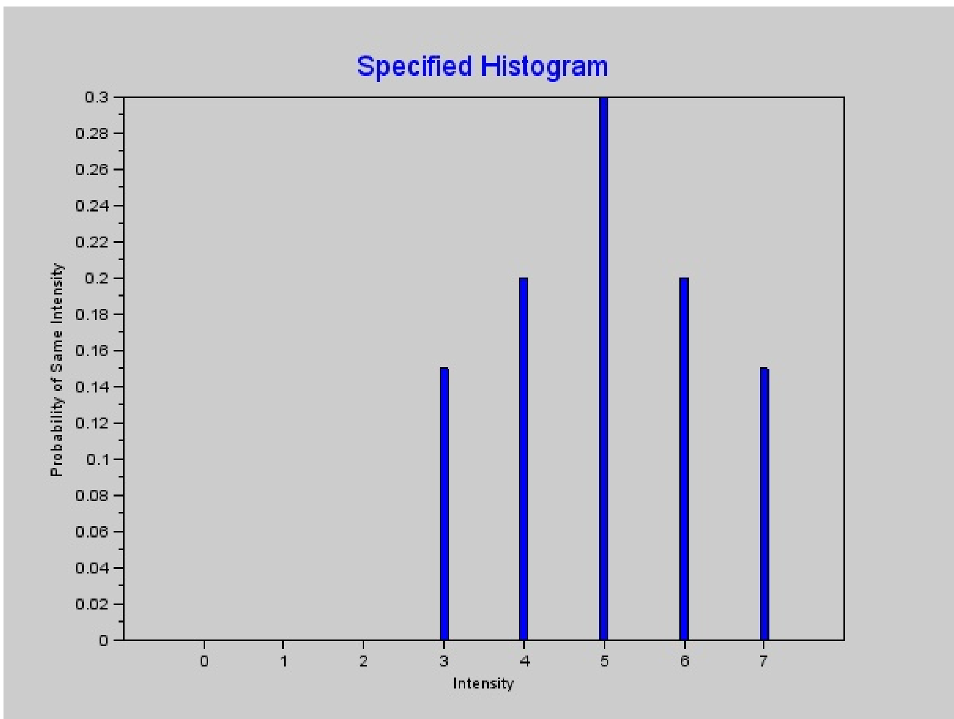


Figure 3.2: Histogram Specification

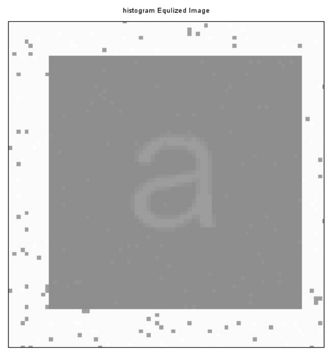


Figure 3.3: Local Histogram Equalization

Scilab code Exa 3.10 `Local Histogram Equalization`

```
//Ex3_10
// Local Histogram Equalization
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods
clc;
close;
```

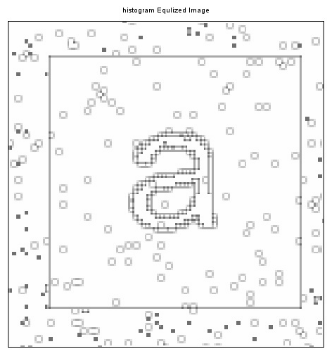


Figure 3.4: Local Histogram Equalization

```
clear;
xdel(winsid())//to close all currently open figure(s).
p1=imread('Ex3_10.tif');
a=imcrop(p1,[175 178 155 160]);
//a=imresize(a,0.5,'bicubic');
[P Q]=size(a);

/////////////////////////// Global Histogram Equalization //////////////////////////
[count cell]=imhist(a);
figure,ShowImage(a,'Original Image');
title('Original Image','color','blue','fontsize',4);

r=cell';  // Transpose of matrix
nk=round(count)';   // Transpose of matrix
M=sum(nk);
probeblity_r=nk/M;  // Probablity calculation
for i=1:length(r)
    sum_1=0;
    for j=1:i
        sum_1=sum_1+probeblity_r(j);
    end
```

```
    s(i)=max(r)*sum_1;
end
s=round(s); // Rounding Approach
//disp(s);
[nr nc]=size(s);
temp=s';    // Transpose of matrix
for i=1:P     // Intensity Replacement in Original Image
    for j=1:Q
        b(i,j)=temp(double(a(i,j))+1);
    end
end
figure,ShowImage(b,'histogram Equlized Image');
title('Image Enhancement using Global Histogram Equalization','color','blue','fontsize',4);

/////////////////////////// Local Histogram Equalization  /////////////////////////
mask=3;
for i=1+floor(mask/2):P-floor(mask/2)
    for j=1+floor(mask/2):Q-floor(mask/2)
        a1=a(i-floor(mask/2):1:i+floor(mask/2),j-floor(mask/2):1:j+floor(mask/2)) ;
        [count cell]=imhist(a1);
        r=cell';  // Transpose of matrix
        nk=round(count)';   // Transpose of matrix
        M=sum(nk);
        probeblity_r=nk/M;  // Probablity calculation
        for x=1:length(r)
            sum_1=0;
            for y=1:x
                sum_1=sum_1+probeblity_r(y);
            end
        s(x)=max(r)*sum_1;
        end
        s=round(s); // Rounding Approach
        //disp(s);
```

```
        [nr nc]=size(s);
        temp=s';    // Transpose of matrix
        b(i,j)=temp(double(a(i,j))+1);
    end
    disp(i);
end
figure,ShowImage(b,'histogram Equlized Image');
title('Image Enhancement using Local Histogram Equalization','color','blue','fontsize',4);
```

Scilab code Exa 3.11 Computing Histogram Statistic

```
//Ex3_11
// Computing Histogram Statistic
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods



clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).
a=uint8([0 0 1 1 2;1 2 3 0 1;3 3 2 2 0;2 3 1 0 0;1 1 3 2 2]);
L=max(a);
[P Q]=size(a);

/////////////////////////// Global Histogram
```

```
   Equalization  //////////////////////////
[count cell]=imhist(a);
//figure,bar(cell(1:L+1),count(1:L+1),0.2);
//title('Histogram');
r=cell(1:L+1)';  // Transpose of matrix
nk=round(count(1:L+1))';   // Transpose of matrix
M=sum(nk);
probeblity_r=nk/M;  // Probablity calculation
sum_1=0;
for i=1:length(r)
        sum_1=sum_1+(r(i)*probeblity_r(i));
end
Mean=sum_1;
disp('Probablity:');
disp(probeblity_r);
disp('Mean:');
disp(Mean);

Mean1=mean(double(a));
disp('Mean Calculated from (5*5)Image:');
disp(Mean1);
```

check Appendix ?? for dependency:

`Ex3_12.tif`

Scilab code Exa 3.12 `Local Enhancement using Histogram Statistic`

```
//Ex3_12
// Local Enhancement using Histogram Statistic
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
```

```
//book author: Rafael C. Gonzalez and Richard E. Woods



clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).
a=imread('Ex3_12.tif');
//a=double(a);
[M N]=size(a);

/////////////////////////// Global Histogram Equalization /////////////////////////
[count cell]=imhist(a);  // Histogram Calculation
figure,ShowImage(a,'Original Image');
title('Original Image','color','blue','fontsize',4);

r=cell';  // Transpose of matrix
nk=round(count)';   // Transpose of matrix
P=sum(nk);
probeblity_r=nk/P;  // Probablity calculation
for i=1:length(r)
    sum_1=0;
    for j=1:i
        sum_1=sum_1+probeblity_r(j);
    end
    s(i)=max(r)*sum_1;
end
s=round(s); // Rounding Approach
disp(s);
[nr nc]=size(s);
temp=s';    // Transpose of matrix
for i=1:M     // Intensity Replacement in Original Image
    for j=1:N
```

```
        b(i,j)=temp(double(a(i,j))+1);
    end
end
figure,ShowImage(b,'histogram Equlized Image');
title('Image Enhancement using Global Histogram Statistic ','color','blue','fontsize',4);


/////////////////////////// Image Enhancement using Local Histogram Statistic /////////////////////////
E=4;K0=0.4;K1=0.02;K2=0.4;
mask=3;
Mean_G=mean(double(a));  // Global Mean Value
Variance_G=variance(double(a));   // Global Variance Value

for i=1+floor(mask/2):M-floor(mask/2)
    for j=1+floor(mask/2):N-floor(mask/2)
        a1=a(i-floor(mask/2):1:i+floor(mask/2),j-floor(mask/2):1:j+floor(mask/2)) ;
        Mean_L=mean(double(a1));   // Local Mean Value
        Variance_L=variance(double(a1));   // Local Variance Value
        if((Mean_L<=K0*Mean_G) & (K1*Variance_G<=Variance_L) & (Variance_L<=K2*Variance_G))
            g(i,j)=E*a(i,j);
        else
            g(i,j)=a(i,j);
        end
    end
end

figure,ShowImage(g,'Local Histogram Statistic');
title('Image Enhancement using Local Histogram Statistic','color','blue','fontsize',4);
```

check Appendix ?? for dependency:

`Ex3_13.tif`

Scilab code Exa 3.13 `Image Smoothing`

```
//Ex3_13
// Image Smoothing
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E.
    Woods



clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s
    ).
a=imread('Ex3_13.tif');
[M N]=size(a);
figure,ShowImage(a,'Original Image');
title('Original Image','color','blue','fontsize',4);

/////////////////////////// Smoothing with Mask Size
    (3*3) /////////////////////////
F=fspecial('average',3);
Image=imfilter(a,F);
figure,ShowImage(Image,'Original Image');
title('Filtered Image with Mask Size(3*3)','color','
    blue','fontsize',4);
```

```

////////////////////////// Smoothing with Mask Size (5*5) /////////////////////////
F=fspecial('average',5);
Image=imfilter(a,F);
figure,ShowImage(Image,'Original Image');
title('Filtered Image with Mask Size(5*5)','color','blue','fontsize',4);

////////////////////////// Smoothing with Mask Size (5*5) /////////////////////////
F=fspecial('average',9);
Image=imfilter(a,F);
figure,ShowImage(Image,'Original Image');
title('Filtered Image with Mask Size(9*9)','color','blue','fontsize',4);

////////////////////////// Smoothing with Mask Size (5*5) /////////////////////////
F=fspecial('average',15);
Image=imfilter(a,F);
figure,ShowImage(Image,'Original Image');
title('Filtered Image with Mask Size(15*15)','color','blue','fontsize',4);

////////////////////////// Smoothing with Mask Size (5*5) /////////////////////////
F=fspecial('average',35);
Image=imfilter(a,F);
figure,ShowImage(Image,'Original Image');
title('Filtered Image with Mask Size(35*35)','color','blue','fontsize',4);
```

check Appendix ?? for dependency:

`Ex3_14.tif`

Scilab code Exa 3.14 Median Filtering for Noise Reduction

```
//Ex3_14
// Median Filtering for Noise Reduction
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods



clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).
a=imread('Ex3_14.tif');
[M N]=size(a);
figure,ShowImage(a,'Original Image');
title('Original Image','color','blue','fontsize',4);

/////////////////////////// Averaging Filter with Mask Size (3*3) /////////////////////////
F=fspecial('average',3);
Image=imfilter(a,F);
figure,ShowImage(Image,'Original Image');
title('Filtered Image with Averaging Filter(3*3)','color','blue','fontsize',4);

/////////////////////////// Median Filtering with Mask Size (5*5) /////////////////////////
Image=MedianFilter(a,[3 3]);
figure,ShowImage(Image,'Original Image');
title('Median Filtered Image with Median Filter(3*3)','color','blue','fontsize',4);
```

check Appendix ?? for dependency:

`Ex3_15.tif`

Scilab code Exa 3.15 `Image Sharpning using Laplacian`

```
//Ex3_15
// Image Sharpning using Laplacian
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods



clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).
a=imread('Ex3_15.tif');
[M N]=size(a);
figure,ShowImage(a,'Original Image');
title('Original Image','color','blue','fontsize',4);

//////////////////////////// Laplacian Filtering ///////////////////////////
F=fspecial('laplacian',0);
Image1=imfilter(a,F);
figure,ShowImage(Image1,'Original Image');
title('Filtered Image with Laplacian Mask','color','blue','fontsize',4);
```

```
//////////////////////////// Laplacian Filtering //////////////////////////
F=[1 1 1;1 -8 1;1 1 1];
Image2=imfilter(a,F);
figure,ShowImage(Image2,'Original Image');
title('Filtered Image with Laplacian Mask','color','blue','fontsize',4);

//////////////////////////// Laplacian Filtering //////////////////////////
b=a-(1*Image1);
figure,ShowImage(b,'Original Image');
title('Filtered Image with Laplacian Mask','color','blue','fontsize',4);


//////////////////////////// Laplacian Filtering //////////////////////////
b=a-(1*Image2);
figure,ShowImage(b,'Original Image');
title('Filtered Image with Laplacian Mask','color','blue','fontsize',4);
```

check Appendix **??** for dependency:

`Ex3_16.tif`

Scilab code Exa 3.16 `Image Sharpning using UnSharp Masking and HighBoost Filtering`

```
//Ex3_16
// Image Sharpning using Un-Sharp Masking and High-Boost Filtering
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
```

```
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods



clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).
a=imread('Ex3_16.tif');
[M N]=size(a);
figure,ShowImage(a,'Original Image');
title('Original Image','color','blue','fontsize',4);

//////////////////////////// Laplacian Filtering //////////////////////////
F=fspecial('gaussian',5,3);
Image1=imfilter(a,F);
figure,ShowImage(Image1,'Original Image');
title('Filtered Image with gaussian Filter(3*3)','color','blue','fontsize',4);

Unsharp_Mask=a-Image1;
figure,ShowImage(Unsharp_Mask,'Original Image');
title('Unsharp Mask Image','color','blue','fontsize',4);

//////////////////////////// Un-Sharp Filtering //////////////////////////
k=1;
Unsharp=a+(k.*Image1);
figure,ShowImage(Unsharp,'Original Image');
title('Unsharp Filtered Image','color','blue','fontsize',4);

```

```
////////////////////////////// High−Boost Filtering
    //////////////////////////
k=4.5;
High_Boost=a+(k.*Image1);
figure,ShowImage(High_Boost,'Original Image');
title('High_Boost Filtered Image','color','blue','
    fontsize',4);
```

check Appendix ?? for dependency:

Ex3_17.png

Scilab code Exa 3.17 `Use of gradient for Edge Enhancement`

```
//Ex3_17
// Use of gradient for Edge Enhancement
// Version : Scilab 5.4.1
// Operating System : Window−xp, Window−7
//Toolbox: Image Processing Design 8.3.1−1
//Toolbox: SIVP 0.5.3.1−2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E.
    Woods



clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s
    ).
a=rgb2gray(imread('Ex3_17.png'));
[M N]=size(a);
figure,ShowImage(a,'Original Image');
title('Original Image','color','blue','fontsize',4);

```

```
////////////////////////// Laplacian Filtering
    //////////////////////////
F=fspecial('sobel');  // Sobel Mask
Image1=imfilter(a,F);
figure,ShowImage(Image1,'Original Image');
title('Filtered Image with Sobel Gradient(3*3)','
    color','blue','fontsize',4);
```

Chapter 4

Filtering in Frequency Domain

Scilab code Exa 4.1 `Obtaining the Fourier Transform of a Simple Function`

```
//Ex4_1
// Obtaining the Fourier Transform of a Simple
   Function
//Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E.
   Woods


clc; //to clear command window.
clear; //to kill previously defined variables.
xdel(winsid());//to close all currently open figure(
   s).

f=-5:0.01:5;
[nr nc]=size(f);
signal=ones(nr,nc);
A=1;
```

```
W=length(f);
for i=1:W
    if(f(i)==0)
        mag(i)=A;
    else
        mag(i)=A*W*(sin(%pi*f(i)*W)/((%pi*f(i)*W)+%eps));
    end

end

figure,mtlb_axis([-6 6 0 2]);
bar(f,signal,0.1);
xlabel('Time Index','color','blue','fontsize',2);
ylabel('Amplitude','color','blue','fontsize',2);
title('Rectangle Function','color','blue','fontsize',4);


figure,//mtlb_axis([-15 15 0 2]);
plot(f,mag);
xlabel('Frequency','color','blue','fontsize',2);
ylabel('Amplitude','color','blue','fontsize',2);
title('Frequency Spectrum Plot','color','blue','fontsize',4);


figure,//mtlb_axis([-15 15 0 2]);
plot(f,abs(mag));
xlabel('Frequency','color','blue','fontsize',2);
ylabel('Amplitude','color','blue','fontsize',2);
title('Frequency Spectrum Plot','color','blue','fontsize',4);
```

Scilab code Exa 4.4 `The Mechanics of Computing the DFT`

```
//Ex4_4
// The Mechanics of Computing the DFT
//Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).
a=[1 2 4 4];
//b=fft2(a);
disp('Original Signal:')
disp(a);
M=length(a);
for i=1:M
    b(i)=0;
    for j=1:M
    b(i)=b(i)+(a(j)*exp((-%i*2*%pi*(i-1)*(j-1)/M)));
    end
end
disp('DFT of Signal:')
disp(b);

for i=1:M
    d(i)=0;
    for j=1:M
    d(i)=d(i)+((b(j)*exp((%i*2*%pi*(i-1)*(j-1)/M)))/M);
    end
end
disp('IDFT:')
disp(abs(d));
```

check Appendix ?? for dependency:

Ex4_7.tif

Scilab code Exa 4.7 Illustration of Aliasing in Resampled Images

```
//Ex4_7
// Illustration of Aliasing in Resampled Images
//Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods


clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).
a=imread("Ex4_7.tif");
figure,ShowImage(a,'Gray Image');
title('Original Image [1025 1025]');
//[M,N]=size(a);
b=imresize(a,0.5,'nearest');
[M,N]=size(b);
d=[];
f=[]
for i=1:N
    temp=b(:,i);
    d=[d temp temp];
end
for i=1:M
```

```
    temp=d(i,:);
    f=[f;temp;temp];
end
figure,ShowImage(f,'Gray Image');
title('Resize Image with Pixels Replication','color'
    ,'blue','fontsize',4);


/////////////////////////// Averaging Approach to
    Reduce Jaggies Effect ////////////////
filt=fspecial('average',3);
a_filter=imfilter(a,filt);
b=imresize(a_filter,0.5,'nearest');
//figure,ShowImage(b,'Gray Image');
//title('Resize Image with nearest Interpolation');
[M,N]=size(b);
d=[];
f=[]
for i=1:N
    temp=b(:,i);
    d=[d temp temp];
end
for i=1:M
    temp=d(i,:);
    f=[f;temp;temp];
end
figure,ShowImage(f,'Gray Image');
title('Resize Image with Pixels Replication After
    Averaging','color','blue','fontsize',4);
```

check Appendix ?? for dependency:

`Ex4_8.tif`

Scilab code Exa 4.8 `Illustration of Jaggies in Image Shrinking`

```
//Ex4_8
// Illustration of Jaggies in Image Shrinking
//Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods


clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).
a=imread("Ex4_8.tif");
figure,ShowImage(a,'Gray Image');
title('Original Image [1024 1024]','color','blue','fontsize',4);
//[M,N]=size(a);
b=imresize(a,[256 256],'bicubic');
//figure,ShowImage(b,'Gray Image');
//title('Resize Image [256 256] with Bicubic Interpolation');
[M,N]=size(b);
d=[];
f=[]
for i=1:N
    temp=b(:,i);
    d=[d temp temp temp temp];
end
for i=1:M
    temp=d(i,:);
    f=[f;temp;temp;temp;temp];
end
figure,ShowImage(f,'Gray Image');
title('Resize Image [1024 1024] with Pixels
```

```
Replication','color','blue','fontsize',4);


////////////////////////////  Averaging Approach to Reduce Jaggies Effect  ////////////////
filt=fspecial('average',5);
a_filter=imfilter(a,filt);
b=imresize(a_filter,[256 256],'bicubic');
//figure,ShowImage(b,'Gray Image');
//title('Resize Image [256 256] with Bicubic Interpolation');
[M,N]=size(b);
d=[];
f=[]
for i=1:N
    temp=b(:,i);
    d=[d temp temp temp temp];
end
for i=1:M
    temp=d(i,:);
    f=[f;temp;temp;temp;temp];
end
figure,ShowImage(f,'Gray Image');
title('Resize Image [1024 1024] with Pixels Replication After Averaging','color','blue','fontsize',4);
```

check Appendix ?? for dependency:

`Ex4_9.tif`

Scilab code Exa 4.9 `Illustration of Jaggies in Image Zooming`

```
//Ex4_9
// Illustration of Jaggies in Image Zooming
//Version : Scilab 5.4.1
```

```
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods


clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).
a=imread("Ex4_9.tif");
a=imcrop(a,[323 377 256 256]);
//figure,ShowImage(a,'Gray Image');
//title('Original Image [1025 1025]');
b=imresize(a,[256 256],'bicubic');
[M,N]=size(b);
d=[];
f=[]
for i=1:N
    temp=b(:,i);
    d=[d temp temp temp temp];
end
for i=1:M
    temp=d(i,:);
    f=[f;temp;temp;temp;temp];
end
figure,ShowImage(f,'Gray Image');
title('Resize Image [1024 1024] with Pixels Replication','color','blue','fontsize',4);


//////////////////////////////  Bi-linear Interpolation  ////////////////

f=imresize(a,[1024 1024],'bilinear');
```

```
figure,ShowImage(f,'Gray Image');
title('Resize Image [1024 1024] with Bi-linear
   Interpolation','color','blue','fontsize',4);
```

check Appendix ?? for dependency:

`Ex4_13_1.tif`

check Appendix ?? for dependency:

`Ex4_13_2.png`

check Appendix ?? for dependency:

`Ex4_13_3.png`

Scilab code Exa 4.13 2 D Fourier Spectrum of a Simple Function

```
//Ex4_13
//The 2-D Fourier Spectrum of a Simple Function
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E.
   Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s
   ).
a=imread("Ex4_13_1.tif");
a=imresize(a,0.5);
//gray=rgb2gray(a);
gray=im2double(a);
```

```

figure,ShowImage(gray,'Gray Image');
title('Original Image','color','blue','fontsize',4);
[M,N]=size(gray);

h1=fft2(gray);//fft2() is used to find 2-Dimensional Fast Fourier Transform of an matrix
figure,ShowImage(mat2gray(abs(h1)),'Frequency spectrum');
title('Frequency spectrum','color','blue','fontsize',4);

in=fftshift(h1);//fftshift() is used to rearrange the fft output, moving the zero frequency to the center of the spectrum.
figure,ShowImage(mat2gray(abs(in)),'Frequency spectrum');
title('Centred Frequency spectrum','color','blue','fontsize',4);

i=log(1+abs(in));
inm=mat2gray(i)
figure,ShowImage(inm,'Frequency Spectrum');//ShowColorImage() is used to show color image, figure is command to view images in separate window.
title('Frequency Spectrum','color','blue','fontsize',4);//title() is used for providing a title to an image.

/////////////////////////// Effect of Translation ///////////////////////////////
a=imread("Ex4_13_2.png");
gray=rgb2gray(a);
gray=im2double(gray);
figure,ShowImage(gray,'Gray Image');
title('Original Image','color','blue','fontsize',4);
[M,N]=size(gray);
```

```
h2=fft2(gray);//fft2() is used to find 2-Dimensional Fast Fourier Transform of an matrix
i=log(1+abs(h2));
in=fftshift(i);//fftshift() is used to rearrange the fft output, moving the zero frequency to the center of the spectrum.
inm=mat2gray(in)
figure,ShowImage(inm,'Frequency Spectrum');//ShowColorImage() is used to show color image, figure is command to view images in separate window.
title('Frequency Spectrum','color','blue','fontsize',4);//title() is used for providing a title to an image.

////////////////////////// Effect of Rotation ///////////////////////////////
a=imread("Ex4_13_3.png");
gray=rgb2gray(a);
gray=im2double(gray);
figure,ShowImage(gray,'Gray Image');
title('Original Image','color','blue','fontsize',4);
[M,N]=size(gray);
h3=fft2(gray);//fft2() is used to find 2-Dimensional Fast Fourier Transform of an matrix
i=log(1+abs(h3));
in=fftshift(i);//fftshift() is used to rearrange the fft output, moving the zero frequency to the center of the spectrum.
inm=mat2gray(in)
figure,ShowImage(inm,'Frequency Spectrum');//ShowColorImage() is used to show color image, figure is command to view images in separate window.
title('Frequency Spectrum','color','blue','fontsize',4);//title() is used for providing a title to an image.

```

```

/////////////////////////// Phase Spectrum ///////////////////////////////
phase=atand(imag(h1),real(h1));
phase_1=mat2gray(phase);
figure,ShowImage(phase_1,'phase Spectrum');
title('phase Spectrum','color','blue','fontsize',4);

phase=atand(imag(h2),real(h2));
phase_1=mat2gray(phase);
figure,ShowImage(phase_1,'phase Spectrum');
title('phase Spectrum of Translated Object','color','blue','fontsize',4);

phase=atand(imag(h3),real(h3));
phase_1=mat2gray(phase);
figure,ShowImage(phase_1,'phase Spectrum');
title('phase Spectrum of Rotated Object','color','blue','fontsize',4);
```

check Appendix ?? for dependency:

`Ex4_14.tif`

check Appendix ?? for dependency:

`Ex4_14_2.tif`

Scilab code Exa 4.14 `Illustration of the Properties of the Fourier Spectrum and Ph`

```
//Ex4_14
// Futher Illustration of a Properties of a Fourier Spectrum and Phase Angle
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
```

```
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).
a=imread("Ex4_14.tif");
mask=imread("Ex4_14_2.tif");
mask=im2double(imresize(mask,[512 512]));
//gray=rgb2gray(a);
gray=im2double(a);

figure,ShowImage(gray,'Gray Image');
title('Original Image','color','blue','fontsize',4);
[M,N]=size(gray);

h=fft2(gray);//fft2() is used to find 2-Dimensional Fast Fourier Transform of an matrix
in=fftshift(h);//fftshift() is used to rearrange the fft output, moving the zero frequency to the center of the spectrum.
i=log(1+abs(in));

inm=mat2gray(i);
//figure,ShowImage(inm,'Center Frequency Spectrum');
//title('Center Frequency Spectrum');

phase=atand(imag(h),real(h));
phase_1=mat2gray(phase);
figure,ShowImage(phase_1,'phase Spectrum');
title('phase Spectrum','color','blue','fontsize',4);

phase_mask=atand(imag(fft2(mask)),real(fft2(mask)));
phase_2=mat2gray(phase_mask);
//figure,ShowImage(phase_2,'phase Spectrum');
```

```
//title('phase Spectrum 2');

Image_recoverd=real(ifft(phase));
Image_recoverd=mat2gray(Image_recoverd)
//figure,ShowImage(Image_recoverd,'recoverd Image');
//title('recoverd Image by only Phase');


Image_recoverd=fftshift(real(ifft(abs(h))));
Image_recoverd=mat2gray(Image_recoverd)
figure,ShowImage(Image_recoverd,'recoverd Image');
title('recoverd Image by only Spectrum','color','blue','fontsize',4);


Image_recoverd=real(ifft(fftshift((mask.*in)+phase)));
Image_recoverd=(mat2gray(Image_recoverd));
figure,ShowImage(Image_recoverd,'recoverd Image');
title('recoverd Image by Magnitude in mask and Phase','color','blue','fontsize',4);


Image_recoverd=real(ifft(fftshift(in)+abs(fft2(mask))));
Image_recoverd=(mat2gray(Image_recoverd));
figure,ShowImage(Image_recoverd,'recoverd Image');
title('recoverd Image by phase in mask and magnitude','color','blue','fontsize',4);
```

check Appendix ?? for dependency:

`Ex4_15.tif`

Scilab code Exa 4.15 `Obtaining a Frequency Domain Filtering from a Small Spatial M`

```
//Ex4_15
// Obtaining a Frequency domain Filter from a Small
   Spatial Mask
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E.
   Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s
   ).

function[H]=sobelfilter(mask_pad)//lowpassfilter is
   used to filter an image.
    x=fft2(mask_pad);
    [nr nc]=size(mask_pad);
    x_real=real(x);
    x_imag=imag(x);
    z=zeros(nr,nc)+%i*x_imag
    H=fftshift(z);
    y=log(1+abs(H));
    y=mat2gray(y)
    figure,ShowImage(y,'Frequency Spectrum');
    title('Frequency Spectrum','color','blue','
       fontsize',4);
endfunction




a=imread("Ex4_15.tif");
gray=im2double(a);
mask=[-1 0 1;-2 0 2;-1 0 1];
```

```
figure,ShowImage(gray,'Gray Image');
title('Original Image','color','blue','fontsize',4);
[M,N]=size(gray);
gray_pad=zeros(M+2,N+2);    // Zero Padding
mask_pad=zeros(M+2,N+2);    // Zero Padding
gray_pad(1:M,1:N)=gray(1:$,1:$);
mask_pad(1:3,1:3)=mask(1:$,1:$);

h=fft2(gray_pad);//fft2() is used to find 2-Dimensional Fast Fourier Transform of an matrix

in=fftshift(h);//fftshift() is used to rearrange the fft output, moving the zero frequency to the center of the spectrum.
i=log(1+abs(in));
inm=mat2gray(i)
figure,ShowImage(inm,'Frequency Spectrum');
title('Frequency Spectrum','color','blue','fontsize',4);

/////////////////////////////// Filtering Domain Filtering   ////////////////////////
filt=sobelfilter(mask_pad); // Function which generate Filter Mask
n=filt.*in;//Multiply the Original Spectrum with the Filter Mask.
n=fftshift(n);
Image_filter=real(ifft(n));
Image_filter=mat2gray(Image_filter)
figure,ShowImage(Image_filter,'Filtered Image');
title('Filtered Image in Frequency Domain','color','blue','fontsize',4);

/////////////////////////////// Spatial Domain Filtering   ////////////////////////

imf = imfilter(a,mask);
//imf=1*(imf.^1.2);
```

```
[r c]=find(imf==0 | imf<=110);
        for i=1:length(r)
                imf(r(i),c(i)) = 125;
        end
figure,ShowImage(imf,'Filtered Image');
title('Filtered Image in Spatial Domain','color','blue','fontsize',4);
```

---

check Appendix ?? for dependency:

`Ex4_16.tif`

Scilab code Exa 4.16 `Image Smoothing using an ILPF`

```
//Ex4_16
//Image Smooting Using an ILPF
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).

function[H]=lowpassfilter(type1,M,N,D0,n)//lowpassfilter is used to filter an image.
    u=0:(M-1);
    v=0:(N-1);
    idx=find(u>M/2);
    u(idx)=u(idx)-M;
```

```
    idy=find(v>N/2);
    v(idy)=v(idy)-N;
    [U,V]=meshgrid(v,u);
    D=sqrt(U.^2+V.^2);
    select type1

    case 'ideal' then
        H=double(D<=D0);
    else
        disp('Unknownfiltertype.')
    end
endfunction


///////////////////////////////////// Main Programm
    //////////////////////////////////
a=imread("Ex4_16.tif");
//gray=rgb2gray(a);
gray=im2double(a);

figure,ShowImage(gray,'Gray Image');
title('Original Image');
[M,N]=size(gray);

h=fft2(gray);//fft2() is used to find 2-Dimensional
   Fast Fourier Transform of an matrix
i=log(1+abs(h));
in=fftshift(i);//fftshift() is used to rearrange the
    fft output, moving the zero frequency to the
   center of the spectrum.
inm=mat2gray(in)
figure,ShowImage(inm,'Frequency Spectrum');
title('Frequency Spectrum','color','blue','fontsize'
   ,4);

//////////////////////////// Filtering With Cut-off
   Frequency 10   //////////////////////
filt=lowpassfilter('ideal',M,N,10); // Function
```

```
which generate Filter Mask Corresponding to Low Frequency
//filt_shift=fftshift(filt);
//figure,ShowImage(filt_shift,'Filter Mask');
//title('Filter Mask to Specific Cut-Off Frequency');
n=filt.*h;//Multiply the Original Spectrum with the Filter Mask.
Image_filter=real(ifft(n));
Image_filter=mat2gray(Image_filter)
figure,ShowImage(Image_filter,'Filtered Image');
title('Filtered Image with Cut-Off Frequency 10','color','blue','fontsize',4);


/////////////////////////// Filtering With Cut-off Frequency 30   ///////////////////////
filt=lowpassfilter('ideal',M,N,30); // Function which generate Filter Mask Corresponding to Low Frequency
//filt_shift=fftshift(filt);
//figure,ShowImage(filt_shift,'Filter Mask');
//title('Filter Mask to Specific Cut-Off Frequency');
n=filt.*h;//Multiply the Original Spectrum with the Filter Mask.
Image_filter=real(ifft(n));
Image_filter=mat2gray(Image_filter)
figure,ShowImage(Image_filter,'Filtered Image');
title('Filtered Image with Cut-Off Frequency 30','color','blue','fontsize',4);


/////////////////////////// Filtering With Cut-off Frequency 60   ///////////////////////
filt=lowpassfilter('ideal',M,N,60); // Function which generate Filter Mask Corresponding to Low Frequency
```

```
//filt_shift=fftshift(filt);
//figure,ShowImage(filt_shift,'Filter Mask');
//title('Filter Mask to Specific Cut-Off Frequency');
n=filt.*h;//Multiply the Original Spectrum with the Filter Mask.
Image_filter=real(ifft(n));
Image_filter=mat2gray(Image_filter)
figure,ShowImage(Image_filter,'Filtered Image');
title('Filtered Image with Cut-Off Frequency 60','color','blue','fontsize',4);


/////////////////////////// Filtering With Cut-off Frequency 160 //////////////////////
filt=lowpassfilter('ideal',M,N,160); // Function which generate Filter Mask Corresponding to Low Frequency
//filt_shift=fftshift(filt);
//figure,ShowImage(filt_shift,'Filter Mask');
//title('Filter Mask to Specific Cut-Off Frequency');
n=filt.*h;//Multiply the Original Spectrum with the Filter Mask.
Image_filter=real(ifft(n));
Image_filter=mat2gray(Image_filter)
figure,ShowImage(Image_filter,'Filtered Image');
title('Filtered Image with Cut-Off Frequency 160','color','blue','fontsize',4);


/////////////////////////// Filtering With Cut-off Frequency 460 //////////////////////
filt=lowpassfilter('ideal',M,N,460); // Function which generate Filter Mask Corresponding to Low Frequency
//filt_shift=fftshift(filt);
//figure,ShowImage(filt_shift,'Filter Mask');
```

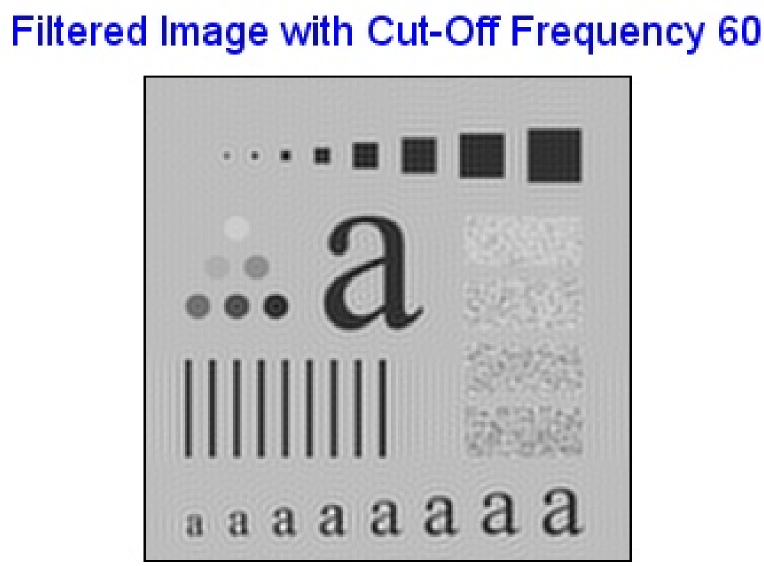


Figure 4.1: Image Smoothing using an ILPF

```
//title('Filter Mask to Specific Cut-Off Frequency');
n=filt.*h;//Multiply the Original Spectrum with the Filter Mask.
Image_filter=real(ifft(n));
Image_filter=mat2gray(Image_filter)
figure,ShowImage(Image_filter,'Filtered Image');
title('Filtered Image with Cut-Off Frequency 460','color','blue','fontsize',4);
```

check Appendix **??** for dependency:

`Ex4_17.tif`

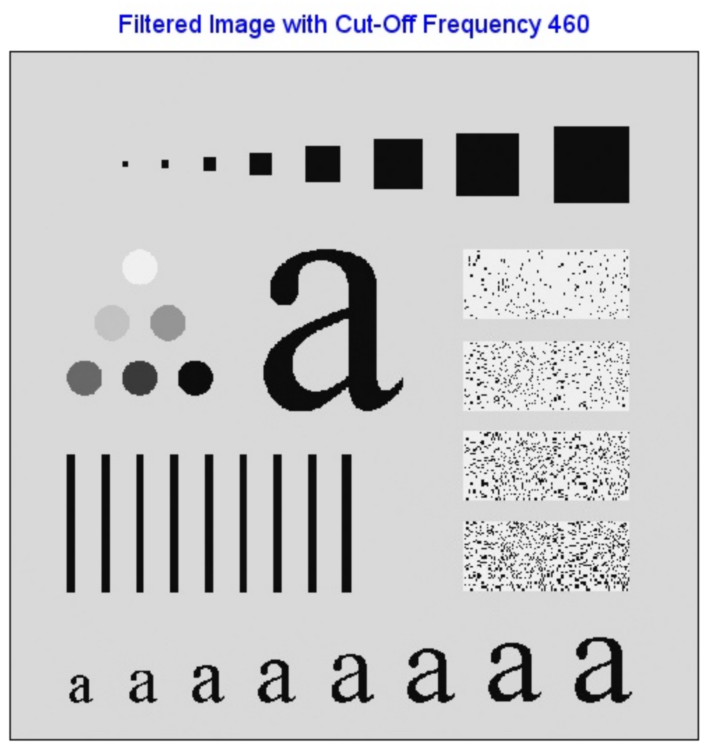


Figure 4.2: Image Smoothing using an ILPF

Scilab code Exa 4.17 `Image Smoothing with a Butterworth Lowpass Filter`

```
//Ex4_17
// Image Smoothing with a Butterworth LowPass Filter
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).

function[H]=lowpassfilter(type1,M,N,D0,n)//lowpassfilter is used to filter an image .
    u=0:(M-1);
    v=0:(N-1);
    idx=find(u>M/2);
    u(idx)=u(idx)-M;
    idy=find(v>N/2);
    v(idy)=v(idy)-N;
    [U,V]=meshgrid(v,u);
    D=sqrt(U.^2+V.^2);
    select type1

    case'butterworth'then
        if argn(2)==4 then
            n=1;
        end
```

```
        H = ones(M,N)./(1+(D./D0).^(2*n));

    else
        disp('Unknownfiltertype.')
    end
endfunction



///////////////////////////////////    Main Programm
    /////////////////////////////////
a=imread("Ex4_17.tif");
//gray=rgb2gray(a);
gray=im2double(a);

figure,ShowImage(gray,'Gray Image');
title('Original Image','color','blue','fontsize',4);
[M,N]=size(gray);

h=fft2(gray);//fft2() is used to find 2-Dimensional
   Fast Fourier Transform of an matrix
i=log(1+abs(h));
in=fftshift(i);//fftshift() is used to rearrange the
    fft output, moving the zero frequency to the
   center of the spectrum.
inm=mat2gray(in)
figure,ShowImage(inm,'Frequency Spectrum');
title('Frequency Spectrum','color','blue','fontsize'
   ,4);

////////////////////////// Filtering With Cut-off
   Frequency 10    //////////////////////
filt=lowpassfilter('butterworth',M,N,10); //
   Function which generate Filter Mask Corresponding
    to Low Frequency
//filt_shift=fftshift(filt);
//figure,ShowImage(filt_shift,'Filter Mask');
//title('Filter Mask to Specific Cut-Off Frequency')
```

```
;
n=filt.*h;//Multiply the Original Spectrum with the Filter Mask.
Image_filter=real(ifft(n));
Image_filter=mat2gray(Image_filter)
figure,ShowImage(Image_filter,'Filtered Image');
title('Filtered Image with Cut-Off Frequency 10','color','blue','fontsize',4);


/////////////////////////// Filtering With Cut-off Frequency 30 //////////////////////
filt=lowpassfilter('butterworth',M,N,30); //Function which generate Filter Mask Corresponding to Low Frequency
//filt_shift=fftshift(filt);
//figure,ShowImage(filt_shift,'Filter Mask');
//title('Filter Mask to Specific Cut-Off Frequency');
n=filt.*h;//Multiply the Original Spectrum with the Filter Mask.
Image_filter=real(ifft(n));
Image_filter=mat2gray(Image_filter)
figure,ShowImage(Image_filter,'Filtered Image');
title('Filtered Image with Cut-Off Frequency 30','color','blue','fontsize',4);


/////////////////////////// Filtering With Cut-off Frequency 60 //////////////////////
filt=lowpassfilter('butterworth',M,N,60); //Function which generate Filter Mask Corresponding to Low Frequency
//filt_shift=fftshift(filt);
//figure,ShowImage(filt_shift,'Filter Mask');
//title('Filter Mask to Specific Cut-Off Frequency');
n=filt.*h;//Multiply the Original Spectrum with the
```

```
Filter Mask.
Image_filter=real(ifft(n));
Image_filter=mat2gray(Image_filter)
figure,ShowImage(Image_filter,'Filtered Image');
title('Filtered Image with Cut-Off Frequency 60','color','blue','fontsize',4);


/////////////////////////// Filtering With Cut-off Frequency 160 //////////////////////
filt=lowpassfilter('butterworth',M,N,160); //Function which generate Filter Mask Corresponding to Low Frequency
//filt_shift=fftshift(filt);
//figure,ShowImage(filt_shift,'Filter Mask');
//title('Filter Mask to Specific Cut-Off Frequency');
n=filt.*h;//Multiply the Original Spectrum with the Filter Mask.
Image_filter=real(ifft(n));
Image_filter=mat2gray(Image_filter)
figure,ShowImage(Image_filter,'Filtered Image');
title('Filtered Image with Cut-Off Frequency 160','color','blue','fontsize',4);


/////////////////////////// Filtering With Cut-off Frequency 460 //////////////////////
filt=lowpassfilter('butterworth',M,N,460); //Function which generate Filter Mask Corresponding to Low Frequency
//filt_shift=fftshift(filt);
//figure,ShowImage(filt_shift,'Filter Mask');
//title('Filter Mask to Specific Cut-Off Frequency');
n=filt.*h;//Multiply the Original Spectrum with the Filter Mask.
Image_filter=real(ifft(n));
```

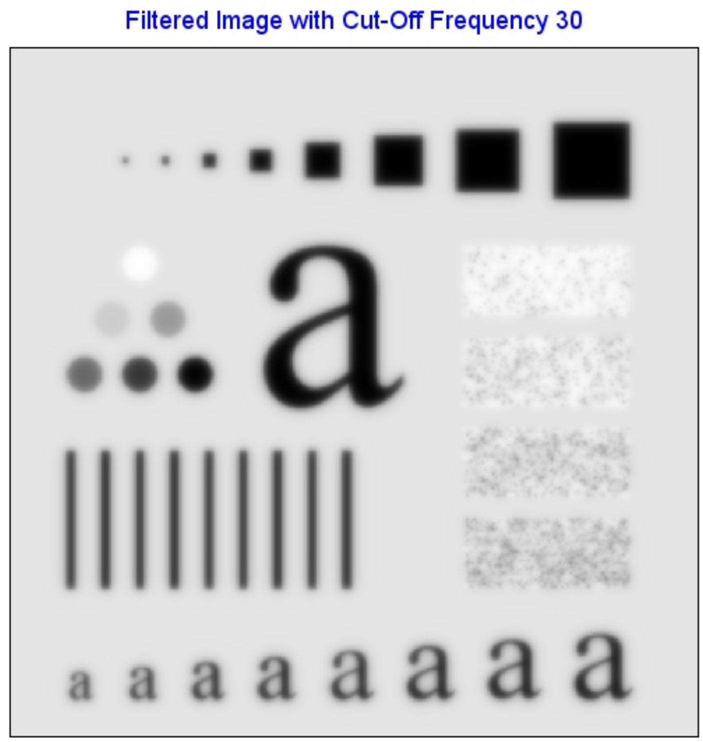


Figure 4.3: Image Smoothing with a Butterworth Lowpass Filter

```
Image_filter=mat2gray(Image_filter)
figure,ShowImage(Image_filter,'Filtered Image');
title('Filtered Image with Cut−Off Frequency 460','
    color','blue','fontsize',4);
```

check Appendix ?? for dependency:

`Ex4_18.tif`

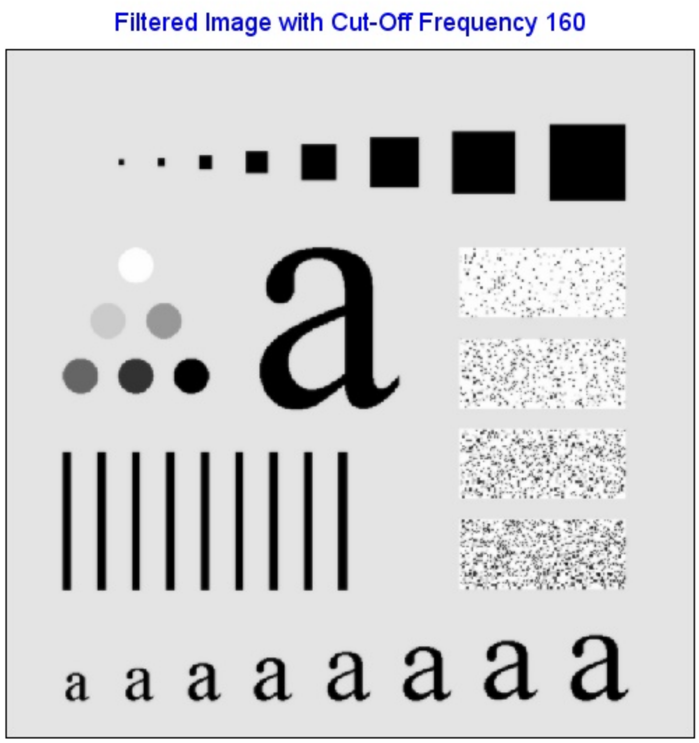


Figure 4.4: Image Smoothing with a Butterworth Lowpass Filter

Scilab code Exa 4.18 Image Smoothing using Gaussian Lowpass Filter

```
//Ex4_18
//Image Smoothing Using Gaussian Lowpass Filter.
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E.
    Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s
    ).

function[H]=lowpassfilter(type1,M,N,D0,n)//
    lowpassfilter is used to filter an image.
    u=0:(M-1);
    v=0:(N-1);
    idx=find(u&gt;M/2);
    u(idx)=u(idx)-M;
    idy=find(v&gt;N/2);
    v(idy)=v(idy)-N;
    [U,V]=meshgrid(v,u);
    D=sqrt(U.^2+V.^2);
    select type1

    case'gaussian'
        H=exp(-(D.^2)./(2*(D0^2)));
    else
        disp('Unknownfiltertype.')
```

```
    end
endfunction



//////////////////////////////////////   Main Programm   ////////////////////////////////////
a=imread("Ex4_18.tif");
//gray=rgb2gray(a);
gray=im2double(a);

figure,ShowImage(gray,'Gray Image');
title('Original Image');
[M,N]=size(gray);

h=fft2(gray);//fft2() is used to find 2-Dimensional Fast Fourier Transform of an matrix
i=log(1+abs(h));
in=fftshift(i);//fftshift() is used to rearrange the fft output, moving the zero frequency to the center of the spectrum.
inm=mat2gray(in)
figure,ShowImage(inm,'Frequency Spectrum');
title('Frequency Spectrum','color','blue','fontsize',4);

////////////////////////////// Filtering With Cut-off Frequency 10   ////////////////////////
filt=lowpassfilter('gaussian',M,N,10); // Function which generate Filter Mask Corresponding to Low Frequency
//filt_shift=fftshift(filt);
//figure,ShowImage(filt_shift,'Filter Mask');
//title('Filter Mask to Specific Cut-Off Frequency');
n=filt.*h;//Multiply the Original Spectrum with the Filter Mask.
Image_filter=real(ifft(n));
```

```
Image_filter=mat2gray(Image_filter)
figure,ShowImage(Image_filter,'Filtered Image');
title('Filtered Image with Cut-Off Frequency 10','color','blue','fontsize',4);


/////////////////////////// Filtering With Cut-off Frequency 30   //////////////////////
filt=lowpassfilter('gaussian',M,N,30); // Function which generate Filter Mask Corresponding to Low Frequency
//filt_shift=fftshift(filt);
//figure,ShowImage(filt_shift,'Filter Mask');
//title('Filter Mask to Specific Cut-Off Frequency');
n=filt.*h;//Multiply the Original Spectrum with the Filter Mask.
Image_filter=real(ifft(n));
Image_filter=mat2gray(Image_filter)
figure,ShowImage(Image_filter,'Filtered Image');
title('Filtered Image with Cut-Off Frequency 30','color','blue','fontsize',4);


/////////////////////////// Filtering With Cut-off Frequency 60   //////////////////////
filt=lowpassfilter('gaussian',M,N,60); // Function which generate Filter Mask Corresponding to Low Frequency
//filt_shift=fftshift(filt);
//figure,ShowImage(filt_shift,'Filter Mask');
//title('Filter Mask to Specific Cut-Off Frequency');
n=filt.*h;//Multiply the Original Spectrum with the Filter Mask.
Image_filter=real(ifft(n));
Image_filter=mat2gray(Image_filter)
figure,ShowImage(Image_filter,'Filtered Image');
```

```
title('Filtered Image with Cut-Off Frequency 60','color','blue','fontsize',4);


/////////////////////////// Filtering With Cut-off Frequency 160   //////////////////////
filt=lowpassfilter('gaussian',M,N,160); // Function which generate Filter Mask Corresponding to Low Frequency
//filt_shift=fftshift(filt);
//figure,ShowImage(filt_shift,'Filter Mask');
//title('Filter Mask to Specific Cut-Off Frequency');
n=filt.*h;//Multiply the Original Spectrum with the Filter Mask.
Image_filter=real(ifft(n));
Image_filter=mat2gray(Image_filter)
figure,ShowImage(Image_filter,'Filtered Image');
title('Filtered Image with Cut-Off Frequency 160','color','blue','fontsize',4);


/////////////////////////// Filtering With Cut-off Frequency 460   //////////////////////
filt=lowpassfilter('gaussian',M,N,460); // Function which generate Filter Mask Corresponding to Low Frequency
//filt_shift=fftshift(filt);
//figure,ShowImage(filt_shift,'Filter Mask');
//title('Filter Mask to Specific Cut-Off Frequency');
n=filt.*h;//Multiply the Original Spectrum with the Filter Mask.
Image_filter=real(ifft(n));
Image_filter=mat2gray(Image_filter)
figure,ShowImage(Image_filter,'Filtered Image');
title('Filtered Image with Cut-Off Frequency 460','color','blue','fontsize',4);
```

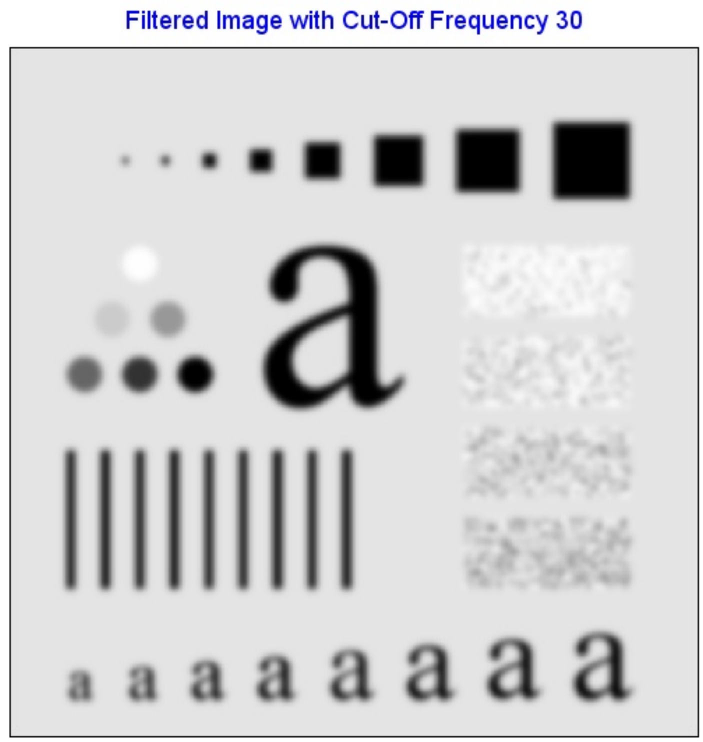


Figure 4.5: Image Smoothing using Gaussian Lowpass Filter

---

check Appendix **??** for dependency:

`Ex4_19.tif`

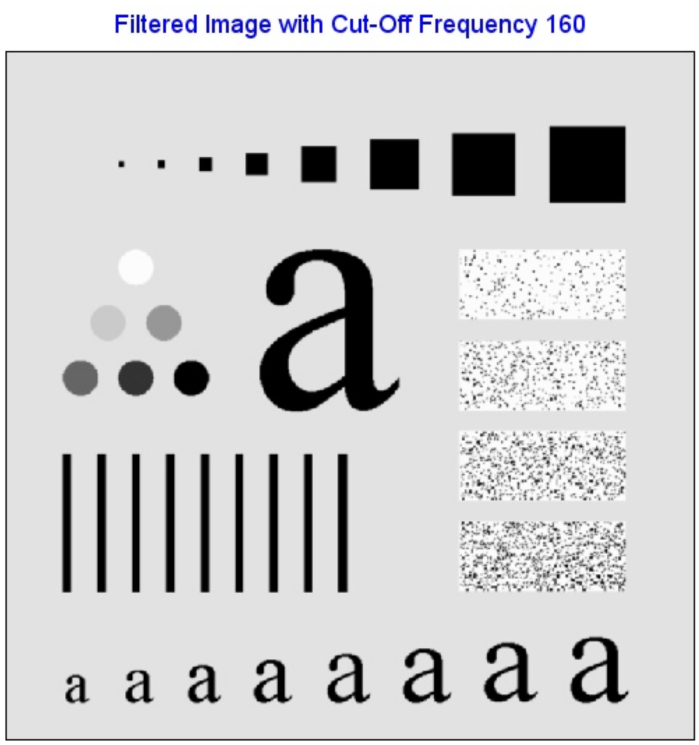


Figure 4.6: Image Smoothing using Gaussian Lowpass Filter

Scilab code Exa 4.19 Using Highpass Filter and Thresholding for Image enhancement

```
//Ex4_19
//Using Highpass Filter and Thresholding for Image Enhancement
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods


clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).

function[H]=lowpassfilter(type1,M,N,D0,n)//lowpassfilter is used to filter an image .
    u=0:(M-1);
    v=0:(N-1);
    idx=find(u&gt;M/2);
    u(idx)=u(idx)-M;
    idy=find(v&gt;N/2);
    v(idy)=v(idy)-N;
    [U,V]=meshgrid(v,u);
    D=sqrt(U.^2+V.^2);
    select type1

    case'ideal'
        H=double(D&lt;=D0);

        case'Laplacian'
        H=1+(4*(%pi)^2*D^2);

```

```
    case 'butterworth'
        if argn(2)==4
            n=1;
        end
        H = ones(M,N)./(1+(D./D0).^(2*n));

        case 'gaussian'
        H=exp(-(D.^2)./(2*(D0^2)));
    else
        disp('Unknownfiltertype.')
    end
endfunction



/////////////////////////////////// Main Programm ///////////////////////////////////
a=imread("Ex4_19.tif");
//gray=rgb2gray(a);
gray=im2double(imresize(a,[540 540]));

figure,ShowImage(gray,'Gray Image');
title('Original Image','color','blue','fontsize',4);
[M,N]=size(gray);

h=fft2(gray);//fft2() is used to find 2-Dimensional Fast Fourier Transform of an matrix
i=log(1+abs(h));
in=fftshift(i);//fftshift() is used to rearrange the fft output, moving the zero frequency to the center of the spectrum.
inm=mat2gray(in)
//figure,ShowImage(inm,'Frequency Spectrum');
//title('Frequency Spectrum');

filt=1-lowpassfilter('butterworth',M,N,50,4); // User Define Function which generate Filter Mask
filt_shift=fftshift(filt);
```

```
//figure,ShowImage(filt_shift,'Filter Mask');
//title('Filter Mask to Specific Cut−Off Frequency');

n=filt.*h;//Multiply the Original Spectrum with the Filter Mask.
Image_filter=real(ifft(n));
Image_filter=mat2gray(Image_filter)
figure,ShowImage(Image_filter,'Filtered Image');
title('Filtered Image with Specific Cut−Off Frequency','color','blue','fontsize',4);

thr = maskthresh(Image_filter);

Image_Enhance=im2bw(Image_filter,thr);
figure,ShowImage(Image_Enhance,'Filtered Image');
title('Enhance Image','color','blue','fontsize',4);
```

check Appendix ?? for dependency:

`Ex4_20.tif`

Scilab code Exa 4.20 `Image Sharpening in the Frequency Domain using the Laplacian`

```
//Ex4_20
// Image Sharping in Frequency Domain Using the Laplacian
// Version : Scilab 5.4.1
// Operating System : Window−xp, Window−7
//Toolbox: Image Processing Design 8.3.1−1
//Toolbox: SIVP 0.5.3.1−2
//Reference book name : Digital Image Processing
```

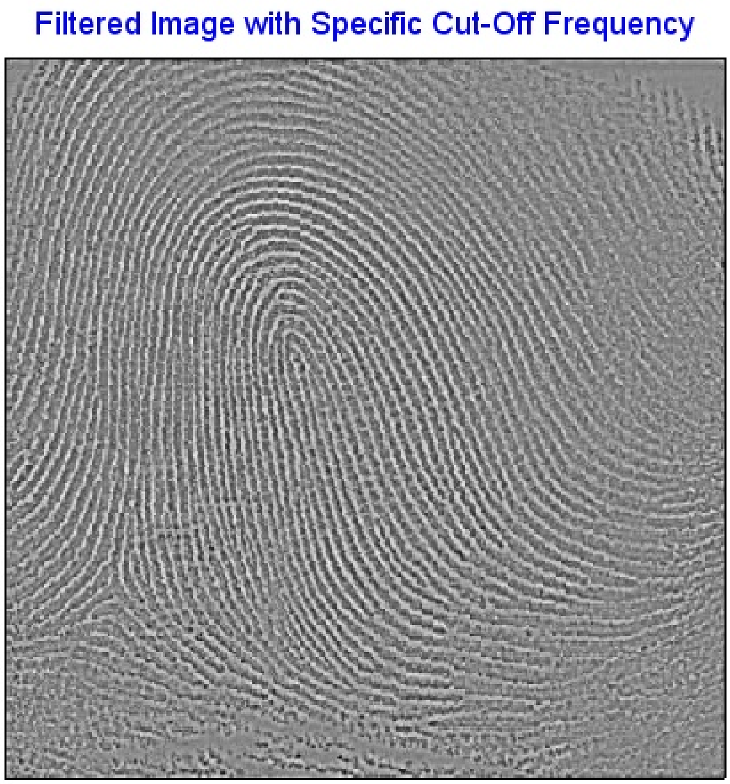


Figure 4.7: Using Highpass Filter and Thresholding for Image enhancement

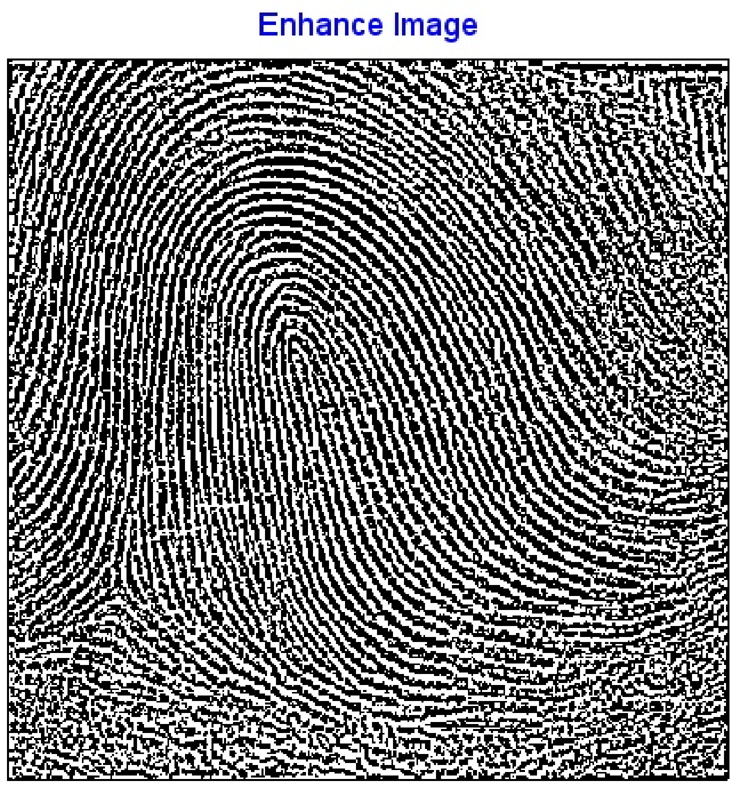


Figure 4.8: Using Highpass Filter and Thresholding for Image enhancement

```
//book author: Rafael C. Gonzalez and Richard E. Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).

function[H]=lowpassfilter(type1,M,N,D0,n)//lowpassfilter is used to filter an image .
    u=0:(M-1);
    v=0:(N-1);
    idx=find(u&gt;M/2);
    u(idx)=u(idx)-M;
    idy=find(v&gt;N/2);
    v(idy)=v(idy)-N;
    [U,V]=meshgrid(v,u);
    D=sqrt(U.^2+V.^2);
    select type1

    case'ideal'
        H=double(D&lt;=D0);

    case'Laplacian'
        H_temp=double(D&lt;=D0);
        H=(4*(%pi)^2*D^2);
        H=H.*H_temp;

    case'butterworth'
        if argn(2)==4
            n=1;
        end
        H = ones(M,N)./(1+(D./D0).^(2*n));
        H_temp=ones(M,N)+(4*(%pi)^2*D^2);
        H=H.*H_temp;

        case'gaussian'
```

```
        H=exp(-(D.^2)./(2*(D0^2)));
    else
        disp('Unknownfiltertype.')
    end
endfunction



////////////////////////////////// Main Programm ////////////////////////////////
a=imread("Ex4_20.tif");
//gray=rgb2gray(a);
gray=im2double(imresize(a,[540 540]));

figure,ShowImage(gray,'Gray Image');
title('Original Image','color','blue','fontsize',4);
[M,N]=size(gray);

h=fft2(gray);//fft2() is used to find 2-Dimensional Fast Fourier Transform of an matrix
i=log(1+abs(h));
in=fftshift(i);//fftshift() is used to rearrange the fft output, moving the zero frequency to the center of the spectrum.
inm=mat2gray(in);
filt=lowpassfilter('Laplacian',M,N,55); // User Define Function which generate Filter Mask Corresponding to Low Frequency
filt_shift=fftshift(filt);
n=filt.*h;//Multiply the Original Spectrum with the Filter Mask.
Image_filter=real(ifft(n));
Image_filter=mat2gray(Image_filter);

z=gray+Image_filter;
figure,ShowImage(mat2gray(z),'Filtered Image');
title('Filtered Image with Specific Cut-Off Frequency','color','blue','fontsize',4);
```

check Appendix ?? for dependency:

Ex4_21.tif

Scilab code Exa 4.21 `Image Enhancement using High Frequency Emphasis Filtering`

```
//Ex4_21
//Image Enhancement using High frequency Emphasis Filtering
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods


clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).

function[H]=lowpassfilter(type1,M,N,D0,n)//lowpassfilter is used to filter an image .
    u=0:(M-1);
    v=0:(N-1);
    idx=find(u&gt;M/2);
    u(idx)=u(idx)-M;
    idy=find(v&gt;N/2);
    v(idy)=v(idy)-N;
    [U,V]=meshgrid(v,u);
    D=sqrt(U.^2+V.^2);
    select type1
```

```

    case'ideal'
        H=double(D&lt;=D0);

        case'Laplacian'
        H=1+(4*(%pi)^2*D^2);


    case'butterworth'
        if argn(2)==4
            n=1;
        end
        H = ones(M,N)./(1+(D./D0).^(2*n));

        case'gaussian'
        H=exp(-(D.^2)./(2*(D0^2)));
    else
        disp('Unknownfiltertype.')
    end
endfunction



////////////////////////////////////  Main Programm //////////////////////////////////
a=imread("Ex4_21.tif");
//gray=rgb2gray(a);
gray=im2double(imresize(a,[540 540]));

figure,ShowImage(gray,'Gray Image');
title('Original Image','color','blue','fontsize',4);
[M,N]=size(gray);

h=fft2(gray);//fft2() is used to find 2-Dimensional Fast Fourier Transform of an matrix
i=log(1+abs(h));
in=fftshift(i);//fftshift() is used to rearrange the fft output, moving the zero frequency to the
```

```
center of the spectrum.
inm=mat2gray(in)
figure,ShowImage(inm,'Frequency Spectrum');
title('Frequency Spectrum','color','blue','fontsize',4);

/////////////////////////////////   Filtering With Cut−off Frequency 10   ///////////////////////
filt=1-lowpassfilter('gaussian',M,N,40); // User Define Function which generate Filter Mask
n=filt.*h;//Multiply the Original Spectrum with the Filter Mask.
Image_filter=real(ifft(n));
Image_filter=mat2gray(Image_filter)
figure(1),ShowImage(Image_filter,'Filtered Image');
title('Filtered Image (High Pass) with Cut−Off Frequency 40','color','blue','fontsize',4);


////////////////////   high boost filtering ///////////////////////////////
filt=0.5+(0.75.*(1-lowpassfilter('gaussian',M,N,40,4))); // User Define Function which generate Filter Mask
n=filt.*h;//Multiply the Original Spectrum with the Filter Mask.
Image_filter=real(ifft(n));

Image_filter=mat2gray(Image_filter)
figure,ShowImage(Image_filter,'Filtered Image');
title('Filtered Image with Specific Cut−Off Frequency','color','blue','fontsize',4);


Image_Enhance=bricontra(Image_filter,180,170,'m'); // Brightness Contrast agjustment (Intensity Transformation)
figure,ShowImage(Image_Enhance,'Filtered Image');
```

```
title('Enhance Image','color','blue','fontsize',4);
```

check Appendix **??** for dependency:

`Ex4_22.tif`

Scilab code Exa 4.22 `Image Enhancement using Homomorphic Filtering`

```
//Ex4_22
// Image Enhancement using Homomorphic Filtering
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).

function [H]=filter(type1,M,N,D0,low,high,c)//lowpassfilter is used to filter an image.
    u=0:(M-1);
    v=0:(N-1);
    idx=find(u&gt;M/2);
    u(idx)=u(idx)-M;
    idy=find(v&gt;N/2);
    v(idy)=v(idy)-N;
```

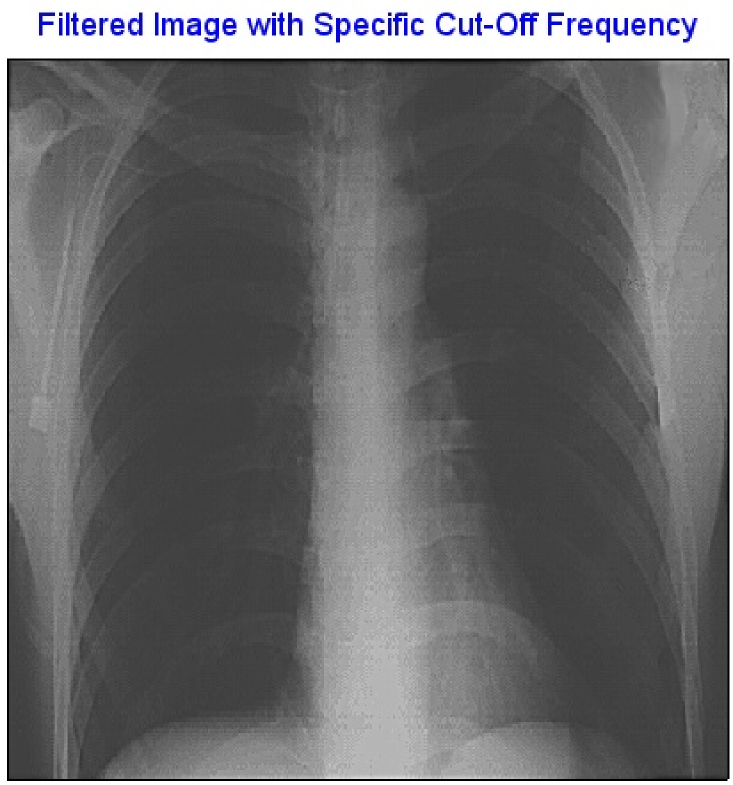


Figure 4.9: Image Enhancement using High Frequency Emphasis Filtering

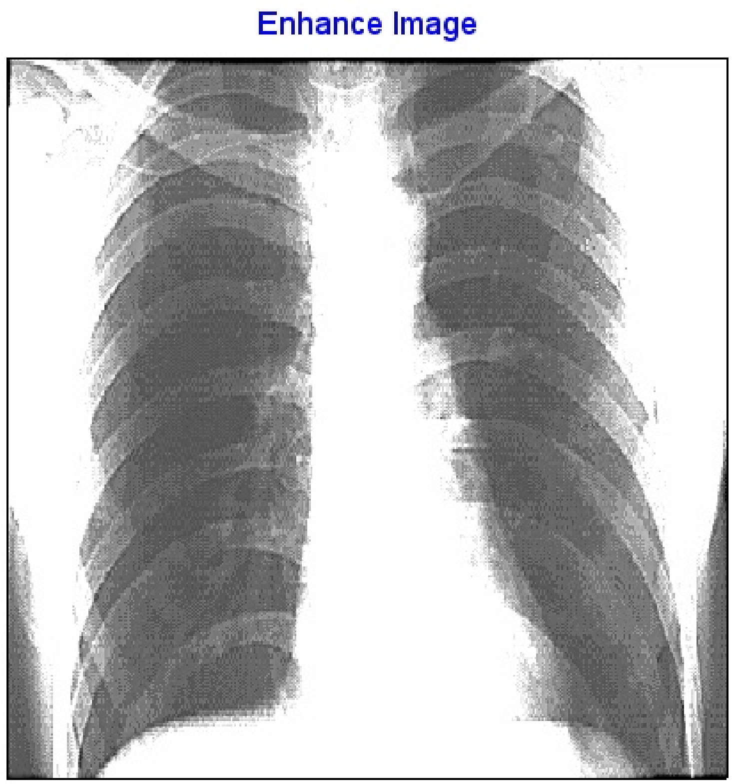


Figure 4.10: Image Enhancement using High Frequency Emphasis Filtering

```
    [U,V]=meshgrid(v,u); // Generate 2-d matrix from 1-d matrix
    D=sqrt(U.^2+V.^2);  // distnace calculation
    select type1
        case 'Homomorphic'
        H=((high-low).*(1-(exp(-c*(D.^2)./(D0^2)))))+low;
    else
        disp('Unknownfiltertype.')
    end
endfunction


//////////////////////////////////////    Main Programm ////////////////////////////////

a=imread("Ex4_22.tif");
//gray=rgb2gray(a);
gray=im2double(imresize(a,[540 540]));

figure,ShowImage(gray,'Gray Image');
title('Original Image','color','blue','fontsize',4);
[M,N]=size(gray);

h=fft2(gray);//fft2() is used to find 2-Dimensional Fast Fourier Transform of an matrix
i=log(1+abs(h));
in=fftshift(i);//fftshift() is used to rearrange the fft output, moving the zero frequency to the center of the spectrum.
inm=mat2gray(in);
low=0.25;
high=2;
c=1;
D0=80;
filt=filter('Homomorphic',M,N,D0,low,high,c); // User Define Function which generate Filter Mask

```

```
n=filt.*h;//Multiply the Original Spectrum with the Filter Mask.
Image_filter=real(ifft(n));
//Image_Enhance = hiseq(a);

Image_filter=mat2gray(Image_filter);
figure,ShowImage(Image_filter,'Filtered Image');
title('Filtered Image with Specific Cut−Off Frequency','color','blue','fontsize',4);
```

check Appendix ?? for dependency:

`Ex4_23.tif`

Scilab code Exa 4.23 `Reduction of Moire Patterns Using Notch Filtering`

```
//Ex4_23
// Reduction of Moire Pattern Using Notch Filtering
// Version : Scilab 5.4.1
// Operating System : Window−xp, Window−7
//Toolbox: Image Processing Design 8.3.1−1
//Toolbox: SIVP 0.5.3.1−2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).

function[H]=notchfilter(type1,M,N,D0,n)//notchfilter is used to filter an image .
    u=0:(M-1);
    v=0:(N-1);
```

```
    idx=find(u>M/2);
    u(idx)=u(idx)-M;
    idy=find(v>N/2);
    v(idy)=v(idy)-N;
    [U,V]=meshgrid(v,u);
    D=sqrt(U.^2+V.^2);
    x=[41 45 82 86 162 166 203 207];
    y=[112 55 112 56 114 58 115 58];
    select type1
        case'ideal'
        //H=double(D<=D0);
H=ones(M,N);
for a=1:M
    for b=1:N
        for i=1:length(x)
        d=sqrt((a-x(i))*(a-x(i))+(b-y(i))*(b-y(i)));
            if (d<D0)
           //H(a,b)=1-(1/(1+(d/D0)^(2*n)));
           H(a,b)=0
            end
        end
    end
end

    case'butterworth'
        if argn(2)==4
            n=1;
        end
        //H = ones(M,N)./(1+(D./D0).^(2*n));
        H=ones(M,N);
    for a=1:M
    for b=1:N
        for i=1:length(x)
        d=sqrt((a-x(i))*(a-x(i))+(b-y(i))*(b-y(i)));
           if (d<D0)
           H(a,b)=1-(1/(1+(d/D0)^(2*n)));
           //H(a,b)=0
           end
```

```
            end
        end
    end

            case'gaussian'
            //H=exp(-(D.^2)./(2*(D0^2)));
            H=ones(M,N);
        for a=1:M
        for b=1:N
            for i=1:length(x)
            d=sqrt((a-x(i))*(a-x(i))+(b-y(i))*(b-y(i)));
               if (d<D0)
               //H(a,b)=1-(1/(1+(d/D0)^(2*n)));
               H(a,b)=1-(exp(-(d.^2)./(2*(D0^2))));
               //H(a,b)=0
               end
            end
        end
    end
        else
            disp('Unknownfiltertype.')
        end

endfunction


//////////////////////////////////// Main Programm
     //////////////////////////////////

a=imread("Ex4_23.tif");
//gray=rgb2gray(a);
gray=im2double(a);

figure,ShowImage(gray,'Gray Image');
title('Original Image','color','blue','fontsize',4);
[M,N]=size(gray);

h=fft2(gray);//fft2() is used to find 2-Dimensional
```

```
Fast Fourier Transform of an matrix
i=log(1+abs(h));
in=fftshift(i);//fftshift() is used to rearrange the fft output, moving the zero frequency to the center of the spectrum.
inm=mat2gray(in)
figure,ShowImage(inm,'Frequency Spectrum');
title('Frequency Spectrum','color','blue','fontsize',4);

filt=notchfilter('gaussian',M,N,9,2); // User Define Function which generate Filter Mask Corresponding to Low Frequency

//filt_shift=fftshift(filt);
n=filt.*fftshift(h);//Multiply the Original Spectrum with the Filter Mask.
figure,ShowImage(abs(n),'Frequency Spectrum');
title('Spectrum After Filtering','color','blue','fontsize',4);
Image_filter=real(ifft(fftshift(n)));
Image_filter=mat2gray(Image_filter)
figure,ShowImage(Image_filter,'Filtered Image');
title('Filtered Image with Specific Cut-Off Frequency','color','blue','fontsize',4);
```

check Appendix ?? for dependency:

`Ex4_24.tif`

Scilab code Exa 4.24 `Enhancement of Corrupted Cassini Saturn Image by Notch Filter`

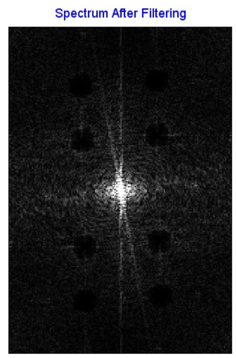


Figure 4.11: Reduction of Moire Patterns Using Notch Filtering

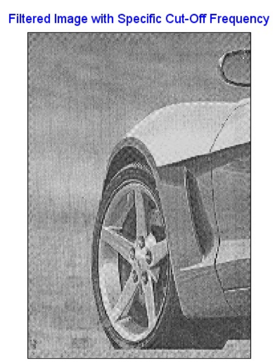


Figure 4.12: Reduction of Moire Patterns Using Notch Filtering

```
//Ex4_24
// Enhancement of Corrupted Cassini Saturn Image by
    Notch Filtering
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E.
    Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s
    ).

function[H]=notchfilter(M,N,W)//notchfilter is used
    to filter an image .
        H=ones(M,N);
        H(1:ceil(M/2-5),ceil(N/2-W/2):ceil(N/2+W/2))
            =0;
        H(ceil(M/2+5):M,ceil(N/2-W/2):ceil(N/2+W/2))
            =0;

endfunction



//////////////////////////////////    Main Programm
      //////////////////////////////////
a=imread("Ex4_24.tif");
//gray=rgb2gray(a);
gray=im2double(a);

figure,ShowImage(gray,'Gray Image');
title('Original Image','color','blue','fontsize',4);
[M,N]=size(gray);
```

```
h=fft2(gray);//fft2() is used to find 2-Dimensional Fast Fourier Transform of an matrix
i=log(1+abs(h));
in=fftshift(i);//fftshift() is used to rearrange the fft output, moving the zero frequency to the center of the spectrum.
inm=mat2gray(in)
figure,ShowImage(inm,'Frequency Spectrum');
title('Frequency Spectrum','color','blue','fontsize',4);

filt=notchfilter(M,N,7); // User Define Function which generate Filter Mask Corresponding to Low Frequency
filt_pass=1-filt;
//filt_shift=fftshift(filt);
figure,ShowImage(filt,'Filter Mask');
title('Filter Mask (Band stop) to Specific Cut-Off Frequency','color','blue','fontsize',4);

n=filt.*fftshift(h);//Multiply the Original Spectrum with the Filter Mask.
Image_filter=real(ifft(fftshift(n)));
Image_filter=mat2gray(Image_filter)
figure,ShowImage(Image_filter,'Filtered Image');
title('Filtered Image with Specific Cut-Off Frequency','color','blue','fontsize',4);


figure,ShowImage(filt_pass,'Filter Mask');
title('Filter Mask (Band Pass) to Specific Cut-Off Frequency','color','blue','fontsize',4);

n=filt_pass.*fftshift(h);//Multiply the Original Spectrum with the Filter Mask.
Image_filter=real(ifft(fftshift(n)));
Image_filter=mat2gray(Image_filter)
```

```
figure,ShowImage(Image_filter,'Filtered Image');
title('Filtered Image (Noise Pattern) with Specific
    Cut-Off Frequency','color','blue','fontsize',4);
```

Chapter 5

Image Restoration and Reconstruction

check Appendix **??** for dependency:

`Ex5_1.tif`

Scilab code Exa 5.1 `Noisy Images and their Histogram`

```
//Ex5_1
// Noisy Images and their Histogram
//To plot the PDF of different Noise Distribution and add the same to the gray scale image.
//(I)Gaussian (II)Uniform (III)Salt &amp; Pepper (IV)Log Normal (V)Rayleigh (VI)Erlang (VII)Exponetial
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

```

```
clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).

function R=imnoise2(type,M,N,a,b)
    if argn(2)==3
        a=0; b=1;
    end

    select type

    case'gaussian'
        rand("normal")
        R=a+b*rand(M,N);

    case'uniform'
        R=a+(b-a)*rand(M,N,"uniform");

    case'salt &amp; pepper'
        if argn(2)==3
            a = 0.15; b = 0.15;
        end
        if (a+b) &gt; 1
            error('The sum Pa + Pb must not exceed 1.');
        end
        R(1:M,1:N) = 0.5;
        X = rand(M,N);
        [r c] = find(X&lt;=a);
        for i=1:length(r)
                R(r(i),c(i)) = 0;
        end
        u = a + b;
        [r c] = find(X&gt;a &amp; X&lt;=u);
        for i=1:length(r)
                R(r(i),c(i)) = 255;
```

```
        end

    case 'lognormal'
        if argn(2)==3
            a = 1; b = 0.25;
        end
        R = a*exp(b*mtlb_randn(M,N));

     case 'rayleigh'
        if argn(2)==3
            a = 1; b = 0.25;
        end
        R = a + ((-b)*(log(1-rand(M,N,"uniform")))).^0.5;

     case 'exponential'
        if argn(2)==3
            a = 1;
        end
        if a&lt;=0
            error('Parameter a must be positive for exponential type.');
        end
        k = -1/a;
        R = k*log(1-rand(M,N,"uniform"));

     case 'erlang'
         if (b ~= round(b) | b &lt;= 0)
            error('Param b must be positive for integer for Erlang.')
        end
        k = -1/a;
        R = zeros(M,N);
        for j=1:b
            R = R + k*log(1-rand(M,N,"uniform"));
        end

        else
```

```
        disp('Unknownfiltertype.')
    end

endfunction



////////////////////////////////////////// Main
   Programm  //////////////////////////////
gray=imread("Ex5_1.tif");
//gray=rgb2gray(a);
//gray=im2double(gray);
figure,ShowImage(gray,'Gray Image');
title('Original Image');
[M,N]=size(gray);
[count,cell]=imhist(gray);
figure,bar(cell,count,0.2);
mtlb_axis([0 255 0 35000]);
title('Histogram of Original Image');

//////////////////////////////////////////    Gaussian
   Noise    ///////////////////
r1=imnoise2('gaussian',M,N,15,5);   // Generate
   Gaussian Noise with Given Mean and Variance
gray_noise_gaussian=gray+(r1);
figure,ShowImage(gray_noise_gaussian,'Gray Image
   with Noise');
title('Gray Image with Noise gaussian');
[count,cell]=imhist(gray_noise_gaussian);
figure;bar(cell,count,1.2);
mtlb_axis([0 255 0 3000]);
title('Gaussian');

//////////////////////////////////////////    Rayleigh
   Noise    ///////////////////
r2=imnoise2('rayleigh',M,N,0,55);   // Generate
   rayleigh Noise
gray_noise_rayleigh=gray+(r2);
```

```
figure,ShowImage(gray_noise_rayleigh,'Gray Image with Noise');
title('Gray Image with Noise rayleigh');
[count,cell]=imhist(gray_noise_rayleigh);
figure;bar(cell,count,1.2);
mtlb_axis([0 255 0 4000]);
title('Rayleigh');

////////////////////////////////////////     Erlang (Gamma) Noise     ////////////////////
r3=imnoise2('erlang',M,N,2,15);   // Generate erlang Noise
gray_noise_erlang=gray+(r3);
figure,ShowImage(gray_noise_erlang,'Gray Image with Noise');
title('Gray Image with Noise erlang(Gamma)');
[count,cell]=imhist(gray_noise_erlang);
figure;bar(cell,count,1.2);
mtlb_axis([0 255 0 9500]);
title('Erlang (Gamma)');

////////////////////////////////////////     Exponential Noise     ////////////////////
r4=imnoise2('exponential',M,N,0.15);    //Generate exponential Noise
gray_noise_exponential=gray+(r4);
figure,ShowImage(gray_noise_exponential,'Gray Image with Noise');
title('Gray Image with Noise exponential');
[count,cell]=imhist(gray_noise_exponential);
figure;bar(cell,count,1.2);
mtlb_axis([0 255 0 4500]);
title('Exponential');

////////////////////////////////////////     Uniform Noise     ////////////////////
r5=imnoise2('uniform',M,N,0,20);   // Generate uniform Noise
```

```
gray_noise_uniform=gray+(r5);
figure,ShowImage(gray_noise_uniform,'Gray Image with Noise');
title('Gray Image with Noise uniform');
[count,cell]=imhist(gray_noise_uniform);
figure;bar(cell,count,1.2);
mtlb_axis([0 255 0 2000]);
title('Uniform');

////////////////////////////////////////     Salt &amp; pepper Noise    ////////////////////
r6=imnoise2('salt &amp; pepper',M,N,0.15,0.15);    // Generate salt &amp; pepper Noise
gray_noise_salt_pepper=gray+(r6);
figure,ShowImage(gray_noise_salt_pepper,'Gray Image with Noise');
title('Gray Image with Noise salt&amp;pepper');
[count,cell]=imhist(gray_noise_salt_pepper);
figure;bar(cell,count,1.2);
mtlb_axis([0 255 0 35000]);
title('Salt &amp; pepper');

//////////////////////////////////////////     lognormal Noise    ////////////////////
//r7=imnoise2('lognormal',M,N,5,0.65);    // Generate lognormal Noise
//gray_noise_lognormal=gray+(r7);
//figure,ShowImage(gray_noise_lognormal,'Gray Image with Noise');
//title('Gray Image with Noise lognormal');
//[count,cell]=imhist(gray_noise_lognormal);
//figure;bar(cell,count,1.2);
//mtlb_axis([0 255 0 5500]);
//title('lognormal');
```

check Appendix ?? for dependency:

`Ex5_2.tif`

Scilab code Exa 5.2 Illustration of Mean Filters

```
//Ex5_2
// Illustration of Mean Filters
//To impliment the Following Mean Restoration filter
//                  (I)Arithmetic  (II)Geometric  (
   III)Harmonic (IV)Contra Harmonic

// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E.
   Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s
   ).


function [f]=arithmetic_mean(v,m,n)
        w=fspecial('average',m);
        f=imfilter(v,w);
endfunction

function [f]=geometric_mean1(g,m,n);//gmean1() is
   used to filter an image using Geometric mean
   filter
    size1=m;
    q=m*n;
    g=double(g);
```

```
    [nr,nc]=size(g);
    temp=zeros(nr+2*floor(size1/2),nc+2*floor(size1/2));
    temp(ceil(size1/2):nr+ceil(size1/2)-1,ceil(size1/2):nc+ceil(size1/2)-1)=g(1:$,1:$)
    temp=temp+1;
    for i=ceil(size1/2):nr+ceil(size1/2)-1
        for j=ceil(size1/2):nc+ceil(size1/2)-1
            t=temp(i-floor(size1/2):1:i+floor(size1/2),j-floor(size1/2):1:j+floor(size1/2)) ;
            temp2(i,j)=prod(t);
        end
    end
    temp3=temp2.^(1/q);
    nn=temp3(ceil(size1/2):nr+ceil(size1/2)-1,ceil(size1/2):nc+ceil(size1/2)-1)
    f1=nn-1;
    f=mat2gray(f1)
endfunction

function [f]=geometric_mean2(g,m,n);//gmean2() is used to filter an image using Geometric mean filter
    size1=m;
    q=m*n;
    [nr,nc]=size(g);
    temp=zeros(nr+2*floor(size1/2),nc+2*floor(size1/2));
    temp(ceil(size1/2):nr+ceil(size1/2)-1,ceil(size1/2):nc+ceil(size1/2)-1)=g(1:$,1:$)
    for i=ceil(size1/2):nr+ceil(size1/2)-1
        for j=ceil(size1/2):nc+ceil(size1/2)-1
            t=temp(i-floor(size1/2):1:i+floor(size1/2),j-floor(size1/2):1:j+floor(size1/2)) ;
            temp2(i,j)=geomean(t);
        end
```

```
    end
    nn=temp2(ceil(size1/2):nr+ceil(size1/2)-1,ceil(size1/2):nc+ceil(size1/2)-1)
    f=mat2gray(nn)
endfunction

function [f]=Harmonic_mean(g,m,n) //harmean1() is used to filter an image using Harmonic mean filter.
    size1=m;
    d=m*n;
    g=double(g);
    [nr,nc]=size(g);
    temp=zeros(nr+2*floor(size1/2),nc+2*floor(size1/2));
    temp(ceil(size1/2):nr+ceil(size1/2)-1,ceil(size1/2):nc+ceil(size1/2)-1)=g(1:$,1:$);

    for i=ceil(size1/2):nr+ceil(size1/2)-1
        for j=ceil(size1/2):nc+ceil(size1/2)-1
            t=temp(i-floor(size1/2):1:i+floor(size1/2),j-floor(size1/2):1:j+floor(size1/2)) ;
            t1=ones(m,n)./(t+%eps);
            t2=sum(t1);
            temp2(i,j)=d/t2;
        end
    end
    nn=temp2(ceil(size1/2):nr+ceil(size1/2)-1,ceil(size1/2):nc+ceil(size1/2)-1);
    f=mat2gray(nn);
endfunction

function [f]=Contra_Harmonic_mean(g,m,n,Q) //charmean1() is use to filter an image using Contra Harmonic mean filter
    size1=m;
    d=m*n;
```

```
    g=double(g);
    [nr,nc]=size(g);
    temp=zeros(nr+2*floor(size1/2),nc+2*floor(size1/2));
    temp(ceil(size1/2):nr+ceil(size1/2)-1,ceil(size1/2):nc+ceil(size1/2)-1)=g(1:$,1:$)
    disp(Q)
    for i=ceil(size1/2):nr+ceil(size1/2)-1
        for j=ceil(size1/2):nc+ceil(size1/2)-1
            t=temp(i-floor(size1/2):1:i+floor(size1/2),j-floor(size1/2):1:j+floor(size1/2)) ;
            d1=(t+%eps).^Q;
            n1=(t+%eps).^(Q+1);
            d2=sum(d1);
            n2=sum(n1);
            temp2(i,j)=n2/(d2);
        end
    end
    nn=temp2(ceil(size1/2):nr+ceil(size1/2)-1,ceil(size1/2):nc+ceil(size1/2)-1)
    f=nn;
endfunction

//////////////////////////////////////     Main Programm     ////////////////////

gray=imread("Ex5_2.tif");
//gray=rgb2gray(a);
//gray=im2double(gray);
figure,ShowImage(gray,'Gray Image');
title('Original Image');
[M,N]=size(gray);

//////////////////////////////////////     Arithmetical Mean Filter     ////////////////////
v=imnoise(gray,'gaussian',0,0.02);
figure,ShowImage(v,'Noisy Image');
```

```
title('Image with Gaussian Noise');
m=3;n=3;
[f]=arithmetic_mean(v,m,n);
figure,ShowImage(f,'Recovered Image');
title('Recovered Image with Arithmetical Mean Filter');

//////////////////////////////////////     Geometric Mean Filter     ////////////////////
v=imnoise(gray,'gaussian',0,0.02);
figure,ShowImage(v,'Noisy Image');
title('Image with Gaussian Noise');
m=3;n=3;
[f]=geometric_mean1(v,m,n);
figure,ShowImage(f,'Recovered Image');
title('Recovered Image with Geometric Mean Filter');


////////////////////////////////////////     Geometric Mean Filter     ////////////////////
//v=imnoise(gray,'gaussian',0,0.02);
//figure,ShowImage(v,'Noisy Image');
//title('Image with Gaussian Noise');
//m=3;n=3;
//[f]=geometric_mean2(v,m,n);
//figure,ShowImage(f,'Recovered Image');
//title('Recovered Image with Geometric Mean Filter');


////////////////////////////////////////     Harmonic Mean Filter     ////////////////////
//temp(1:M,1:N)=0.5;
//r3=imnoise(temp,'salt & pepper',0.1);    // Generate salt & pepper Noise
//gray_noise_salt=gray;                     // Add salt Noise Only
//[r c]=find(r3==1);
```

```
//          for  i=1:length(r)
//                  gray_noise_salt(r(i),c(i)) = 255;
//          end
//figure,ShowImage(gray_noise_salt,'Noisy Image');
//title('Image with Salt Noise');
//m=3;n=3;
//[f]=Harmonic_mean(gray_noise_salt,m,n);
//figure,ShowImage(f,'Recovered Image');
//title('Recovered Image with Harmonic Mean Filter');
//
/////////////////////////////////  Contra_Harmonic Mean Filter (Pepper)   ///////////////////
temp(1:M,1:N)=0.5;
r3=imnoise(temp,'salt & pepper',0.05);     //Generate salt & pepper Noise
gray_noise_pepper=gray;                     //Add pepper Noise Only
[r c]=find(r3==0);                          //Find pepper Noise Only
        for i=1:length(r)
                gray_noise_pepper(r(i),c(i)) = 0;
        end
figure,ShowImage(gray_noise_pepper,'Noisy Image');
title('Image with pepper Noise');
m=3;n=3;Q=1.5;
[f]=Contra_Harmonic_mean(gray_noise_pepper,m,n,Q);
figure,ShowImage(f,'Recovered Image');
title('Recovered Image with Contra Harmonic Mean Filter[ Q=1.5 ]');

//////////////////////////////////// Contra_Harmonic Mean Filter (Salt) ///////////////////
temp(1:M,1:N)=0.5;
r3=imnoise(temp,'salt & pepper',0.1);     //Generate salt & pepper Noise
gray_noise_salt=gray;                     //Add salt
```

```
Noise Only
[r c]=find(r3==1);
        for i=1:length(r)
                gray_noise_salt(r(i),c(i)) = 255;
        end
figure,ShowImage(gray_noise_salt,'Noisy Image');
title('Image with Salt Noise');
m=3;n=3;Q=-1.5;
[f]=Contra_Harmonic_mean(gray_noise_salt,m,n,Q);
figure,ShowImage(f,'Recovered Image');
title('Recovered Image with Contra Harmonic Mean Filter[ Q=-1.5 ]');


//////////////////////////////  Contra_Harmonic Mean Filter (Pepper)  ////////////////////
temp(1:M,1:N)=0.5;
r3=imnoise(temp,'salt & pepper',0.05);    //Generate salt & pepper Noise
gray_noise_pepper=gray;                         // Add pepper Noise Only
[r c]=find(r3==0);                              //Find pepper Noise Only
        for i=1:length(r)
                gray_noise_pepper(r(i),c(i)) = 0;
        end
figure,ShowImage(gray_noise_pepper,'Noisy Image');
title('Image with pepper Noise');
m=3;n=3;Q=-1.5;
[f]=Contra_Harmonic_mean(gray_noise_pepper,m,n,Q);
figure,ShowImage(f,'Recovered Image');
title('Recovered Image with Contra Harmonic Mean Filter[ Q=-1.5 ]');

/////////////////////////////////// Contra_Harmonic Mean Filter (Salt) ////////////////////
temp(1:M,1:N)=0.5;
```

```
r3=imnoise(temp,'salt & pepper',0.1);    //Generate salt & pepper Noise
gray_noise_salt=gray;                    //Add salt Noise Only
[r c]=find(r3==1);
        for i=1:length(r)
                gray_noise_salt(r(i),c(i)) = 255;
        end
figure,ShowImage(gray_noise_salt,'Noisy Image');
title('Image with Salt Noise');
m=3;n=3;Q=1.5;
[f]=Contra_Harmonic_mean(gray_noise_salt,m,n,Q);
figure,ShowImage(f,'Recovered Image');
title('Recovered Image with Contra Harmonic Mean Filter[ Q=1.5 ]');
```

check Appendix ?? for dependency:

`Ex5_3.tif`

Scilab code Exa 5.3 `Illustration of Order Statistic filter`

```
//Ex5_3
// Illustration of Order Statistic filter
//To impliment the Following Order Statistic Restoration filter
//                  (I)Median  (II)MAX  (III)MIN (IV)Mid Point (V)Alpha trimmed.

// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods
```

```
clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).

function [f]=arithmetic_mean(v,m,n)
        w=fspecial('average',m);
        f=imfilter(v,w);
endfunction

function [f]=geometric_mean1(g,m,n);//gmean1() is used to filter an image using Geometric mean filter
    size1=m;
    q=m*n;
    g=double(g);
    [nr,nc]=size(g);
    temp=zeros(nr+2*floor(size1/2),nc+2*floor(size1/2));
    temp(ceil(size1/2):nr+ceil(size1/2)-1,ceil(size1/2):nc+ceil(size1/2)-1)=g(1:$,1:$)
    temp=temp+1;
    for i=ceil(size1/2):nr+ceil(size1/2)-1
        for j=ceil(size1/2):nc+ceil(size1/2)-1
            t=temp(i-floor(size1/2):1:i+floor(size1/2),j-floor(size1/2):1:j+floor(size1/2)) ;
            temp2(i,j)=prod(t);
        end
    end
    temp3=temp2.^(1/q);
    nn=temp3(ceil(size1/2):nr+ceil(size1/2)-1,ceil(size1/2):nc+ceil(size1/2)-1)
    f1=nn-1;
    f=mat2gray(f1)
endfunction
```

```
function [f]=restoration_filter(v,type,m,n,Q,d)
    if argn(2) ==2
        m=7;n=7;Q=1.5;d=10;
    elseif argn(2)==5
        Q=parameter;d=parameter;
    elseif argn(2)==4
        Q=1.5;d=2;
    else
        disp('wrong number of inputs');
    end

    select type

    case'median'
        f=MedianFilter(v,[m n]);

    case'MIN'
       size1=m;
       [nr,nc]=size(v);
       temp=zeros(nr+2*floor(size1/2),nc+2*floor(size1/2));
       temp(ceil(size1/2):nr+ceil(size1/2)-1,ceil(size1/2):nc+ceil(size1/2)-1)=v(1:$,1:$);
        for i=ceil(size1/2):nr+ceil(size1/2)-1
            for j=ceil(size1/2):nc+ceil(size1/2)-1
                t=temp(i-floor(size1/2):1:i+floor(size1/2),j-floor(size1/2):1:j+floor(size1/2)) ;
                y=gsort(t);
                temp2(i-floor(size1/2),j-floor(size1/2))=min(y);
            end
        end
        f=mat2gray(temp2);

    case'MAX'
       size1=m;
```

```
        [nr,nc]=size(v);
        temp=zeros(nr+2*floor(size1/2),nc+2*floor(size1/2));
        temp(ceil(size1/2):nr+ceil(size1/2)-1,ceil(size1/2):nc+ceil(size1/2)-1)=v(1:$,1:$);
         for i=ceil(size1/2):nr+ceil(size1/2)-1
             for j=ceil(size1/2):nc+ceil(size1/2)-1
                 t=temp(i-floor(size1/2):1:i+floor(size1/2),j-floor(size1/2):1:j+floor(size1/2)) ;
                 y=gsort(t);
                 temp2(i-floor(size1/2),j-floor(size1/2))=max(y);
             end
         end
         f=mat2gray(temp2);

         case'Mid_Point'
        size1=m;
        [nr,nc]=size(v);
        temp=zeros(nr+2*floor(size1/2),nc+2*floor(size1/2));
        temp(ceil(size1/2):nr+ceil(size1/2)-1,ceil(size1/2):nc+ceil(size1/2)-1)=v(1:$,1:$);
         for i=ceil(size1/2):nr+ceil(size1/2)-1
             for j=ceil(size1/2):nc+ceil(size1/2)-1
                 t=temp(i-floor(size1/2):1:i+floor(size1/2),j-floor(size1/2):1:j+floor(size1/2)) ;
                 y=gsort(t);
                 temp2(i-floor(size1/2),j-floor(size1/2))=0.5*(min(y)+max(y));
             end
         end
         f=mat2gray(temp2);

         else
         disp('Unknownfiltertype.')
```

```
    end
endfunction

function [f]=alphatrim(g,m,n,d)//alphatrim() is used to filter an image using alpha-trimmed mean filter
    size1=m;
    [nr,nc]=size(g);
    temp=zeros(nr+2*floor(size1/2),nc+2*floor(size1/2));
    temp(ceil(size1/2):nr+ceil(size1/2)-1,ceil(size1/2):nc+ceil(size1/2)-1)=g(1:$,1:$)

    for i=ceil(size1/2):nr+ceil(size1/2)-1
        for j=ceil(size1/2):nc+ceil(size1/2)-1
            t=temp(i-floor(size1/2):1:i+floor(size1/2),j-floor(size1/2):1:j+floor(size1/2))
            y=gsort(t);
            a=y(:)
            b=a';
            t1=b(1+d/2:$-d/2);
            temp2(i-floor(size1/2),j-floor(size1/2))=mean(t1);
        end
    end
    f=mat2gray(temp2)
endfunction


//////////////////////////////////////     Main Programm     ////////////////////

gray=imread("Ex5_3.tif");
//gray=rgb2gray(a);
//gray=im2double(gray);
figure,ShowImage(gray,'Gray Image');
title('Original Image');
```

```
[M,N]=size(gray);

////////////////////////////////////          Median Filter     ////////////////////
v=imnoise(gray,'salt &amp; pepper',0.1);
figure,ShowImage(v,'Noisy Image');
title('Original Image with Salt &amp; Pepper Noise');
//Filtering the corrupted image with median filter
h=restoration_filter(v,'median',3,3);
figure,ShowImage(h,'Recovered Image');
title('Recovered Image with Median Filter');
//Filtering the corrupted image with median filter
h1=restoration_filter(h,'median',3,3);
figure,ShowImage(h1,'Recovered Image');
title('Recovered Image with Median Filter');
//Filtering the corrupted image with median filter
h2=restoration_filter(h1,'median',3,3);
figure,ShowImage(h2,'Recovered Image');
title('Recovered Image with Median Filter');


////////////////////////////////////          MAX Filter          ////////////////////
temp(1:M,1:N)=0.5;
r3=imnoise(temp,'salt &amp; pepper',0.1);    // Generate salt &amp; pepper Noise
gray_noise_pepper=gray;                            // Add Pepper Noise Only
[r c]=find(r3==0);
        for i=1:length(r)
                gray_noise_pepper(r(i),c(i)) = 0;
        end
figure,ShowImage(gray_noise_pepper,'Noisy Image');
title('Noisy Image with Pepper Noise');

//Filtering the Salt Noise corrupted image with MAX filter
```

```
h=restoration_filter(gray_noise_pepper,'MAX',3,3);
figure,ShowImage(h,'Recovered Image');
title('Recovered Image with MAX Filter');


//////////////////////////////////////        MIN Filter    ////////////////////
temp(1:M,1:N)=0.5;
r3=imnoise(temp,'salt &amp; pepper',0.1);   // Generate salt &amp; pepper Noise
gray_noise_salt=gray;                        // Add salt Noise Only
[r c]=find(r3==1);
        for i=1:length(r)
                gray_noise_salt(r(i),c(i)) = 255;
        end
figure,ShowImage(gray_noise_salt,'Noisy Image');
title('Noisy Image');

//Filtering the Salt Noise corrupted image with MIN filter
h=restoration_filter(gray_noise_salt,'MIN',3,3);
figure,ShowImage(h,'Recovered Image');
title('Recovered Image with MIN Filter');


///////////////////////////////////////        Mid-Point Filter    ////////////////////
//v=imnoise(gray,'gaussian',0,0.02);
//figure,ShowImage(v,'Noisy Image');
//title('Image with Gaussian Noise');
////Filtering the Salt Noise corrupted image with Mid-Point filter
//h=restoration_filter(v,'Mid_Point',3,3);
//figure,ShowImage(h,'Recovered Image');
//title('Recovered Image with Mid_Point Filter');


```

```
//////////////////////////////////          Alpha Trimmed Filter     ///////////////////
v=imnoise(gray,'gaussian',0,0.02);
v=imnoise(v,'salt &amp; pepper',0.05);
figure,ShowImage(v,'Noisy Image');
title('Image with Gaussian and Salt&amp;Pepper Noise');
m=5;n=5;d=5;
[f]=arithmetic_mean(v,m,n);  // Filtering with Arithmetical mean
figure,ShowImage(f,'Recovered Image');
title('Recovered Image with Arithmetical Mean Filter');
[f]=geometric_mean1(v,m,n);   // Filtering with Geometric mean
figure,ShowImage(f,'Recovered Image');
title('Recovered Image with Geometric Mean Filter');
//Filtering the corrupted image with median filter
h=restoration_filter(v,'median',5,5);  // Filtering with median Filtering
figure,ShowImage(h,'Recovered Image');
title('Recovered Image with Median Filter');
f=alphatrim(v,m,n,d);   // Filtering with alphatrim Filtering
figure,ShowImage(f,'Recovered Image');
title('Recovered Image with Alpha Trimmed Filter');
```

check Appendix **??** for dependency:

`Ex5_4.tif`

Scilab code Exa 5.4 `Illustration of Adaptive Local Noise Reduction Filtering`

```
//Ex5_4
//Illustration of Adaptive Local Noise Reduction Filtering
```

```
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

clc;
clear;
close;
xdel(winsid());

///////////////// Function File //////////////////////
function [f]=arithmetic_mean(v,m,n)
        w=fspecial('average',m);
        f=imfilter(v,w);
endfunction

function [f]=geometric_mean1(g,m,n);//gmean1() is used to filter an image using Geometric mean filter
    size1=m;
    q=m*n;
    g=double(g);
    [nr,nc]=size(g);
    temp=zeros(nr+2*floor(size1/2),nc+2*floor(size1/2));
    temp(ceil(size1/2):nr+ceil(size1/2)-1,ceil(size1/2):nc+ceil(size1/2)-1)=g(1:$,1:$)
    temp=temp+1;
    for i=ceil(size1/2):nr+ceil(size1/2)-1
        for j=ceil(size1/2):nc+ceil(size1/2)-1
            t=temp(i-floor(size1/2):1:i+floor(size1/2),j-floor(size1/2):1:j+floor(size1/2)) ;
            temp2(i,j)=prod(t);
```

```
        end
    end
    temp3=temp2.^(1/q);
    nn=temp3(ceil(size1/2):nr+ceil(size1/2)-1,ceil(size1/2):nc+ceil(size1/2)-1)
    f1=nn-1;
    f=mat2gray(f1)
endfunction


///////////////// Main Programm /////////////////////
A=imread("Ex5_4.tif");
B = imnoise(A,'gaussian',0,0.01);
[rw1 ,cl1]=size(B);
figure;
ShowImage(B,'Gaussian noise added');
title('Image with gaussian noise','color','blue','fontsize',4);

///////////////////////////////////// Arithmetical Mean Filter ///////////////////
m=7;n=7;
[f]=arithmetic_mean(B,m,n);
figure,ShowImage(f,'Recovered Image');
title('Restored Image with Arithmetical Mean Filter','color','blue','fontsize',4);

///////////////////////////////////// Geometric Mean Filter ///////////////////
m=7;n=7;
[f]=geometric_mean1(B,m,n);
figure,ShowImage(f,'Recovered Image');
title('Restored Image with Geometric Mean Filter','color','blue','fontsize',4);



```

```
////////////////////Adaptive Local Noise Reduction
    //////////////////////
B= double(B);
M=7;
N=7;
lvar=zeros([rw1-M+1,cl1-N+1]);
lmean=zeros([rw1-M+1,cl1-N+1]);
temp=zeros([rw1-M+1,cl1-N+1]);
F=zeros([rw1-M+1,cl1-N+1]);
sz=(rw1-M+1)*(cl1-N+1);
for i=1:rw1-M+1
    for j=1:cl1-N+1
        temp=B(i:i+(M-1),j:j+(N-1));
        lmean(i,j)=mean(temp);
        lvar(i,j)=mean(temp.*temp)-mean(temp).^2;
     end
end
nvar=sum(lvar)/sz;
lvar=max(lvar,nvar);
C=B(M/2:rw1-M/2,N/2:cl1-N/2);
F=nvar./lvar;
F=F.*(C-lmean);
F=C-F;
F=uint8(F);
figure;
ShowImage(F,'Restored');
title('Restored Image using Adaptive Local filter','
    color','blue','fontsize',4);
```

check Appendix ?? for dependency:

`Ex5_5.tif`

Scilab code Exa 5.5 `Illustration of Adaptive Median Filter`

```
//Ex5_5
```

```
// Illustration of Adaptive Median Filter
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

clc;
clear;
close;
xdel(winsid());
A=imread("Ex5_5.tif");
A=imresize(A,[256 256]);
A=imnoise(A,'salt & pepper',0.25);   // Add Sali & Pepper Noise
figure,ShowImage(A,'Salt & pepper Image');
title('Image with Salt & pepper noise (Density = 0.25)','color','blue','fontsize',4);
figure,ShowImage(MedianFilter(A,[7 7]),'Median filter with mask 7x7');
title('Restored Image using Median filter with 7*7 Mask','color','blue','fontsize',4);

///////////////// Adaptive Median Filter /////////////////////////
[r c]=size(A);
n=7 // Maximum Window size
a=(n-1)/2;
C=zeros(r-2*a,c-2*a);
for i=a+1:(r-a)
    for j=a+1:(c-a)
        for b=3:2:7
            d=(b-1)/2
        x=A(i,j);
        p=imcrop(A,[i-d j-d b b])  // Crop the Sub Image form Original Iamge
```

```
        med=median(p); // To Find Median Value
        maxx=max(p);   // To Find Max Value
        minn=min(p);   // To Find Min Value
        if (med>minn & med<maxx) then
            if(x>minn & x<minn) then
                C(j-a+1,i-a+1)=x;
                clear p;
                break;
            else
                C(j-a+1,i-a+1)=med;
                clear p;
                break;
            end
        elseif b<7 then
            continue;
        else
            C(j-a+1,i-a+1)=med;
            clear p;
            break;
        end
    end
end
end
figure;ShowImage(C,'Adaptive Median filter Image
    using code');
title('Restored Image using Adaptive Median filter',
    'color','blue','fontsize',4);
```

check Appendix ?? for dependency:

`Ex5_8.tif`

Scilab code Exa 5.8 Removal of Periodic Noise by Notch Filtering

```
//Ex5_8
//Removal of Periodic Noise by Notch Filtering
```

```
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).

function[H]=notchfilter(M,N,W)//notchfilter is used to filter an image .
        H=ones(M,N);
        H(1:ceil(M/2-10),ceil(N/2-W/2):ceil(N/2+W/2))=0;
        H(ceil(M/2+10):M,ceil(N/2-W/2):ceil(N/2+W/2))=0;

endfunction



///////////////////////////////////// Main Programm ///////////////////////////////////
a=imread("Ex5_8.tif");
//gray=rgb2gray(a);
gray=im2double(imresize(a,0.5));
figure,ShowImage(gray,'Gray Image');
title('Original Image');
[M,N]=size(gray);

h=fft2(gray);//fft2() is used to find 2-Dimensional Fast Fourier Transform of an matrix
i=log(1+abs(h));
```

```
in=fftshift(i);//fftshift() is used to rearrange the fft output, moving the zero frequency to the center of the spectrum.
inm=mat2gray(in)
figure,ShowImage(inm,'Frequency Spectrum');
title('Frequency Spectrum');

filt=notchfilter(M,N,3); // User Define Function which generate Filter Mask Corresponding to Low Frequency
filt_pass=1-filt;
//filt_shift=fftshift(filt);
figure,ShowImage(filt,'Filter Mask');
title('Filter Mask (Band stop) to Specific Cut-Off Frequency');

n=filt.*fftshift(h);//Multiply the Original Spectrum with the Filter Mask.
Image_filter=real(ifft(fftshift(n)));
Image_filter=mat2gray(Image_filter)
figure,ShowImage(Image_filter,'Filtered Image');
title('Filtered Image with Specific Cut-Off Frequency');


figure,ShowImage(filt_pass,'Filter Mask');
title('Filter Mask (Band Pass) to Specific Cut-Off Frequency');

n=filt_pass.*fftshift(h);//Multiply the Original Spectrum with the Filter Mask.
Image_filter=real(ifft(fftshift(n)));
Image_filter=mat2gray(Image_filter)
figure,ShowImage(Image_filter,'Filtered Image');
title('Filtered Image (Noise Pattern) with Specific Cut-Off Frequency');
```

check Appendix ?? for dependency:

Ex5_10.png

Scilab code Exa 5.10 Image Blurring Due to Motion

```
//Ex5_10
//Image Bluring Due to Motion
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).

gray=imread("Ex5_10.png");
gray=im2double(rgb2gray(gray));
//gray=im2double(imresize(a,0.5));
figure,ShowImage(gray,'Gray Image');
title('Original Image','color','blue','fontsize',4);
[M,N]=size(gray);

h=fft2(gray);//fft2() is used to find 2-Dimensional Fast Fourier Transform of an matrix
i=log(1+abs(h));
in=fftshift(i);//fftshift() is used to rearrange the fft output, moving the zero frequency to the center of the spectrum.
inm=mat2gray(in)

```

```
a=0.1;b=0.1;T=1; // Motion and Exposure Value
for u=1:M
    for v=1:N
        H(u,v)=(T/(%pi*(u*a+v*b)))*(sin(%pi*(u*a+v*b
           )))*exp(-%i*%pi*(u*a+v*b));  //Motion
           Blure Function
    end
end

n=h.*H;//Multiply the Original Spectrum with the
   Degradation Function.
Image_filter=abs(ifft(n));
Image_filter=mat2gray(Image_filter)
figure,ShowImage(Image_filter,'Filtered Image');
title('Motion Blure Image','color','blue','fontsize'
   ,4);
```

check Appendix ?? for dependency:

Ex5_11.png

Scilab code Exa 5.11 Inverse Filtering

```
//Ex5_11
//Inverse Filtering
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E.
   Woods

clc;
close;
clear;
```

```
xdel(winsid())//to close all currently open figure(s).

function[H,H1]=lowpassfilter(type1,M,N,D0,n,k)//lowpassfilter is used to filter an image .
    u=0:(M-1);
    v=0:(N-1);
    idx=find(u&gt;M/2);
    u(idx)=u(idx)-M;
    idy=find(v&gt;N/2);
    v(idy)=v(idy)-N;
    [U,V]=meshgrid(v,u);
    D=sqrt(U.^2+V.^2); //Distance Calculation
    D=fftshift(D);
    for i=1:M
        for j=1:N
            H(i,j)=exp(-k.*((i-(M/2))^2+(j-(N/2))^2).^(5/6)); //Atmospheric Degradation Function
        end
    end

    select type1

    case'inverse'
        if argn(2)==4
            n=1;k=0.0025;
        end
        H=H;
        H1=H;

    case'butterworth'
        if argn(2)==4
            n=1;
        end
//          H1 = (ones(M,N)./(1+(D./D0).^(2*n)));
        H1=double(D&lt;=D0);
        H=H.*H1;
```

```

        else
         disp('Unknownfiltertype.')
     end
endfunction

////////////////////////////////////   Main Programm  /////////////////////////////////
gray=imread('Ex5_11.png');
gray=im2double(rgb2gray(gray));
figure,ShowImage(gray,'Gray Image');
title('Original Image','color','blue','fontsize',4);
[M,N]=size(gray);

h=fft2(gray);//fft2() is used to find 2−Dimensional Fast Fourier Transform of an matrix
in=fftshift(h);//fftshift() is used to rearrange the fft output, moving the zero frequency to the center of the spectrum.
i=log(1+abs(in));

inm=mat2gray(i)

////////////////////////// Filtering With Cut−off Frequency 480   ///////////////////////
[filt,H1]=lowpassfilter('inverse',M,N,480,1,0.0025);// Function which generate Filter Mask Corresponding to Low Frequency
//filt_shift=fftshift(filt);
//figure,ShowImage(abs(filt),'Filter Mask');
//title('Filter Mask to Specific Cut−Off Frequency');
n=in./(filt+%eps);//Multiply the Original Spectrum with the Filter Mask.
Image_filter=abs(ifft(fftshift(n)));
Image_filter=mat2gray(Image_filter)
figure,ShowImage(Image_filter,'Filtered Image');
title('Filtered Image with Full Inverse Filter','
```

```
color','blue','fontsize',4);

/////////////////////////// Filtering With Cut-off Frequency 40   //////////////////////
[filt,H1]=lowpassfilter('butterworth',M,N,40,10,0.0025); // Function which generate Filter Mask Corresponding to Low Frequency
//filt_shift=fftshift(filt);
//figure,ShowImage(abs(filt),'Filter Mask');
//title('Filter Mask to Specific Cut-Off Frequency');
n=(in.*H1)./(filt+%eps);//Multiply the Original Spectrum with the Filter Mask.
Image_filter=abs(ifft(fftshift(n)));
Image_filter=mat2gray(Image_filter)
figure,ShowImage(Image_filter,'Filtered Image');
title('Filtered Image with Cut-Off Frequency 40','color','blue','fontsize',4);

/////////////////////////// Filtering With Cut-off Frequency 70   //////////////////////
[filt,H1]=lowpassfilter('butterworth',M,N,70,10,0.0025); // Function which generate Filter Mask Corresponding to Low Frequency
//filt_shift=fftshift(filt);
//figure,ShowImage(abs(filt),'Filter Mask');
//title('Filter Mask to Specific Cut-Off Frequency');
n=(in.*H1)./(filt+%eps);//Multiply the Original Spectrum with the Filter Mask.
Image_filter=abs(ifft(fftshift(n)));
Image_filter=mat2gray(Image_filter)
figure,ShowImage(Image_filter,'Filtered Image');
title('Filtered Image with Cut-Off Frequency 70','color','blue','fontsize',4);

/////////////////////////// Filtering With Cut-off Frequency 100   //////////////////////
```

```
[filt,H1]=lowpassfilter('butterworth',M,N,100,10,0.0025); // Function which generate Filter Mask Corresponding to Low Frequency
//filt_shift=fftshift(filt);
//figure,ShowImage(abs(filt),'Filter Mask');
//title('Filter Mask to Specific Cut-Off Frequency');
n=(in.*H1)./(filt+%eps);//Multiply the Original Spectrum with the Filter Mask.
Image_filter=abs(ifft(fftshift(n)));
Image_filter=mat2gray(Image_filter)
figure,ShowImage(Image_filter,'Filtered Image');
title('Filtered Image with Cut-Off Frequency 100','color','blue','fontsize',4);
```

check Appendix ?? for dependency:

`Ex5_12.png`

Scilab code Exa 5.12 `Comparision of Inverse Filtering and Wiener Filtering`

```
//Ex5_12
//Comparision of Inverse Filtering and Wiener Filtering
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

clc;
close;
clear;
```

```
xdel(winsid())//to close all currently open figure(s).

function[H,H1]=lowpassfilter(type1,M,N,D0,n,k)//lowpassfilter is used to filter an image .
    u=0:(M-1);
    v=0:(N-1);
    idx=find(u&gt;M/2);
    u(idx)=u(idx)-M;
    idy=find(v&gt;N/2);
    v(idy)=v(idy)-N;
    [U,V]=meshgrid(v,u);
    D=sqrt(U.^2+V.^2); //Distance Calculation
    D=fftshift(D);
    for i=1:M
        for j=1:N
            H(i,j)=exp(-k.*((i-(M/2))^2+(j-(N/2))^2).^(5/6)); //Atmospheric Degradation Function
        end
    end

    select type1

    case'inverse'
        if argn(2)==4
            n=1;k=0.0025;
        end
        H=H;
        H1=H;

    case'butterworth'
        if argn(2)==4
            n=1;
        end
//          H1 = (ones(M,N)./(1+(D./D0).^(2*n)));
        H1=double(D&lt;=D0);
        H=H.*H1;
```

```

        else
         disp('Unknownfiltertype.')
     end
endfunction

/////////////////////////////////// Main Programm ///////////////////////////////////
gray=imread('Ex5_12.png');
gray=im2double(rgb2gray(gray));
figure,ShowImage(gray,'Gray Image');
title('Original Image','color','blue','fontsize',4);
[M,N]=size(gray);

h=fft2(gray);//fft2() is used to find 2-Dimensional Fast Fourier Transform of an matrix
in=fftshift(h);//fftshift() is used to rearrange the fft output, moving the zero frequency to the center of the spectrum.
i=log(1+abs(in));

inm=mat2gray(i)

///////////////////////// Filtering With Cut-off Frequency 480 //////////////////////
[filt,H1]=lowpassfilter('inverse',M,N,480,1,0.0025); // Function which generate Filter Mask Corresponding to Low Frequency
//filt_shift=fftshift(filt);
//figure,ShowImage(abs(filt),'Filter Mask');
//title('Filter Mask to Specific Cut-Off Frequency');
n=in./(filt+%eps);//Multiply the Original Spectrum with the Filter Mask.
Image_filter=abs(ifft(fftshift(n)));
Image_filter=mat2gray(Image_filter)
figure,ShowImage(Image_filter,'Filtered Image');
title('Filtered Image with Full Inverse Filter','
```

```
color','blue','fontsize',4);


////////////////////////// Filtering With Cut-off Frequency 40   ///////////////////////
[filt,H1]=lowpassfilter('butterworth',M,N,40,10,0.0025); // Function which generate Filter Mask Corresponding to Low Frequency
//filt_shift=fftshift(filt);
//figure,ShowImage(abs(filt),'Filter Mask');
//title('Filter Mask to Specific Cut-Off Frequency');
n=(in.*H1)./(filt+%eps);//Multiply the Original Spectrum with the Filter Mask.
Image_filter=abs(ifft(fftshift(n)));
Image_filter=mat2gray(Image_filter)
figure,ShowImage(Image_filter,'Filtered Image');
title('Filtered Image with Cut-Off Frequency 40','color','blue','fontsize',4);



////////////////////////// Filtering With Cut-off Frequency 40   ///////////////////////
[filt,H1]=lowpassfilter('butterworth',M,N,40,10,0.0025); // Function which generate Filter Mask Corresponding to Low Frequency
//filt_shift=fftshift(filt);
//figure,ShowImage(abs(filt),'Filter Mask');
//title('Filter Mask to Specific Cut-Off Frequency');
n=(in.*H1)./(((1/filt+%eps).*(filt^2/(filt^2+6)))+%eps); // Wiener Filtering.
Image_filter=abs(ifft(fftshift(n)));
Image_filter=mat2gray(Image_filter)
figure,ShowImage(Image_filter,'Filtered Image');
title('Filtered Image with Cut-Off Frequency 40','color','blue','fontsize',4);
```

Chapter 6

Color Image Processing

check Appendix **??** for dependency:

`Ex6_3.tif`

Scilab code Exa 6.3 `Intensity Slicing`

```
//Ex6_3 :
//Intensity Slicing

// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E.
    Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s
    ).
gray=imread("Ex6_3.tif");
```

```
[nr nc]=size(gray);

figure,ShowImage(gray,'Gray Image');
title('Original Image');
min_image=min(gray); // Find Minimum Intensity value
max_image=max(gray); // Find Maximum Intensity value

color_RED=[0 255 0 0 0 255 255 255];  // RED Component Value of the Pseudo Color
color_GREEN=[0 0 0 255 255 255 0 255]; // GREEN Component Value of the Pseudo Color
color_BLUE=[0 255 255 255 0 0 0 255];  // BLUE Component Value of the Pseudo Color
k=8;
Slice_Image=[];
for y=1:k // Decide Total No. of Level
for i=1:nr
    for j=1:nc
        if(gray(i,j)>=((max_image/k)*(y-1)) & gray(i,j)<((max_image/k)*y))
            Slice_Image(i,j,1)=color_RED(y);
            Slice_Image(i,j,2)=color_GREEN(y);
            Slice_Image(i,j,3)=color_BLUE(y);
         end
    end
end
end
imshow(Slice_Image);//,'Intensity Slicing');
//title('Image After Intensity Slicing');
```

check Appendix ?? for dependency:

`Ex6_4.tif`

Scilab code Exa 6.4 `Use of Color to Highlight Rainfall Levels`

```
//Ex6_4 :
//Use of Color to Highlight Rainfall Levels

// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).
gray=imread("Ex6_4.tif");
gray=imresize(gray,0.25);
[nr nc]=size(gray);

figure,ShowImage(gray,'Gray Image');
title('Original Image');
min_image=min(gray); // Find Minimum Intensity value
max_image=max(gray); // Find Maximum Intensity value

color_RED=[0 255 0 0 0 255 255 255];  // RED Component Value of the Pseudo Color
color_GREEN=[0 0 0 255 255 255 0 255]; // GREEN Component Value of the Pseudo Color
color_BLUE=[0 255 255 255 0 0 0 255];  // BLUE Component Value of the Pseudo Color
k=8;
Slice_Image=[];
for y=1:k // Decide Total No. of Level
for i=1:nr
    for j=1:nc
        if(gray(i,j)>=((max_image/k)*(y-1)) & gray(i,j)<((max_image/k)*y))
```

```
                Slice_Image(i,j,1)=color_RED(y);
                Slice_Image(i,j,2)=color_GREEN(y);
                Slice_Image(i,j,3)=color_BLUE(y);
            end
        end
end
end
imshow(Slice_Image);//,'Intensity Slicing');
//title('Image After Intensity Slicing');
```

check Appendix ?? for dependency:

Ex6_5.png

Scilab code Exa 6.5 Use of Psuedocolor for highlighting Explosives Contained in Lu

```
//Ex6_5 :
//Use of Psedocolor for highlighting Exposives Contained in Luggage.

// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).

theta=0:450
RED=abs(255*sind(theta));
```

```
GREEN=abs(255*sind(theta-40));
BLUE=abs(255*sind(theta-80));
figure;
subplot(311),plot(theta,RED);
title('RED Intensity Transformation');
subplot(312),plot(theta,GREEN);
title('GREEN Intensity Transformation');
subplot(313),plot(theta,BLUE);
title('BLUE Intensity Transformation');

gray=rgb2gray(imread("Ex6_5.png"));
//gray=imresize(gray,0.25);
[nr nc]=size(gray);

figure,ShowImage(gray,'Gray Image');
title('Original Image');
//min_image=min(gray); // Find Minimum Intensity
    value
//max_image=max(gray); // Find Maximum Intensity
    value
//
//color_RED=[0 255 0 0 0 255 255 255];  // RED
    Component Value of the Pseudo Color
//color_GREEN=[0 0 0 255 255 255 0 255]; // GREEN
    Component Value of the Pseudo Color
//color_BLUE=[0 255 255 255 0 0 0 255];  // BLUE
    Component Value of the Pseudo Color
//k=8;
Slice_Image=[];
//for y=1:k // Decide Total No. of Level
for i=1:nr
    for j=1:nc
            Slice_Image(i,j,1)=RED(gray(i,j));
            Slice_Image(i,j,2)=GREEN(gray(i,j));
            Slice_Image(i,j,3)=BLUE(gray(i,j));
    end
end
//end
```

```
imshow(Slice_Image);//,'Intensity Slicing');
//title('Image After Intensity Slicing');
```

check Appendix ?? for dependency:

Ex6_6_1.TIF

check Appendix ?? for dependency:

Ex6_6_2.TIF

check Appendix ?? for dependency:

Ex6_6_3.TIF

check Appendix ?? for dependency:

Ex6_6_4.TIF

Scilab code Exa 6.6 Color Coding of Multi Spectral Images

```
//Ex6_6 :
//Color Coding of Multi Spectral Images.

// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E.
    Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s
    ).
gray1=imresize(imread("Ex6_6_1.tif"),0.5);
```

```
gray2=imresize(imread("Ex6_6_2.tif"),0.5);
gray3=imresize(imread("Ex6_6_3.tif"),0.5);
gray4=imresize(imread("Ex6_6_4.tif"),0.5);

figure,ShowImage(gray1,'Gray Image');
title('Visible RED Band Component');
figure,ShowImage(gray2,'Gray Image');
title('Visible GREEN Band Component');
figure,ShowImage(gray3,'Gray Image');
title('Visible BLUE Band Component');
figure,ShowImage(gray4,'Gray Image');
title('Near Infrared Band Image');

temp(:,:,1)=gray1;   //Visible RED Band Component
temp(:,:,2)=gray2;   //Visible GREEN Band Component
temp(:,:,3)=gray3;   //Visible BLUE Band Component
figure,ShowColorImage(temp,'Color Image');
title('Color Composite Image');

temp1(:,:,1)=gray4;   //Near Infrared Band Component
temp1(:,:,2)=gray2;   //Visible GREEN Band Component
temp1(:,:,3)=gray3;   //Visible BLUE Band Component
figure,ShowColorImage(temp1,'Color Image');
title('Color Composite Image');
```

check Appendix ?? for dependency:

`Ex6_7.tif`

Scilab code Exa 6.7 `Computing Color Image Components`

```
//Ex6_7 :
//Computing Color Image Components.

// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
```

```
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E.
    Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s
    ).
Color=imread("Ex6_7.tif");
Color=imresize(Color,0.5);
[nr nc]=size(Color);

figure,ShowColorImage(Color,'Gray Image');
title('Original Image');


Slice_Image=[];

for i=1:nr
    for j=1:nc
            Slice_Image(i,j,1)=255-Color(i,j,1);
            Slice_Image(i,j,2)=255-Color(i,j,2);
            Slice_Image(i,j,3)=255-Color(i,j,3);
   end
end

ShowColorImage(Slice_Image,'RGB Image');;
title('RGB Mapped image');
```

check Appendix ?? for dependency:

`Ex6_9_1.tif`

check Appendix ?? for dependency:

`Ex6_9_2.tif`

check Appendix ?? for dependency:

`Ex6_9_3.tif`

Scilab code Exa 6.9 `Tonal Transformations`

```
//Ex6_9 :
//Tonal Transformations.

// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).

////////////////// Tonal Correction for the Flat Image /////////////////////////
Color=imread("Ex6_9_1.tif");
Color=imresize(Color,0.5);
[nr nc]=size(Color);
figure,ShowColorImage(Color,'Gray Image');
title('Original Image');
D=0:256;
D0=155;// Cut-off Number
n=2;  // Order of Butter Wirth Approximation
H1 = 1-ones(1,1)./(1+(D./D0).^(2*n));  // Transfer Function (Design from the Butterworth Approximation)
```

```
figure,plot(H1);
title('RGB Intensity Transformation Function');
Slice_Image=[];
for i=1:nr
    for j=1:nc
            Slice_Image(i,j,1)=H1(uint16(Color(i,j,1))+1);
            Slice_Image(i,j,2)=H1(uint16(Color(i,j,2))+1);
            Slice_Image(i,j,3)=H1(uint16(Color(i,j,3))+1);
     end
end
ShowColorImage(Slice_Image,'RGB Image');
title('Tonal Corrected image');

////////////////// Tonal Correction for the Light Image ////////////////////////
Color=imread("Ex6_9_2.tif");
Color=imresize(Color,0.5);
[nr nc]=size(Color);
figure,ShowColorImage(Color,'Gray Image');
title('Original Image');
D=0:1/256:1;
H2=1*D^3.0;  // Transfer Function (Design from the Gamma Funcetion).
figure,plot(H2);
title('RGB Intensity Transformation Function');
Slice_Image=[];
for i=1:nr
    for j=1:nc
            Slice_Image(i,j,1)=H2(uint16(Color(i,j,1))+1);
            Slice_Image(i,j,2)=H2(uint16(Color(i,j,2))+1);
            Slice_Image(i,j,3)=H2(uint16(Color(i,j,3))+1);
     end
```

```
end
ShowColorImage(Slice_Image,'RGB Image');
title('Tonal Corrected image');

////////////////// Tonal Correction for the Dark Image ////////////////////////
Color=imread("Ex6_9_3.tif");
Color=imresize(Color,0.5);
[nr nc]=size(Color);
figure,ShowColorImage(Color,'Gray Image');
title('Original Image');
D=0:1/256:1;
H3=1*D^0.35;
figure,plot(H3);
title('RGB Intensity Transformation Function');
Slice_Image=[];
for i=1:nr
    for j=1:nc
            Slice_Image(i,j,1)=H3(uint16(Color(i,j,1))+1);
            Slice_Image(i,j,2)=H3(uint16(Color(i,j,2))+1);
            Slice_Image(i,j,3)=H3(uint16(Color(i,j,3))+1);
     end
end
ShowColorImage(Slice_Image,'RGB Image');
title('Tonal Corrected image');
```

check Appendix ?? for dependency:

`Ex6_10.tif`

Scilab code Exa 6.10 `Color Balancing`

```
//Ex6_10 :
```

```
//Color Balancing.

// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E.
   Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s
   ).

////////////////// Tonal Correction for the Flat
   Image ////////////////////////
Color=imread("Ex6_10.tif");
Color=imresize(Color,0.25);
[nr nc]=size(Color);
figure,ShowColorImage(Color,'Gray Image');
title('Original Image','color','blue','fontsize',4);

C=255-Color(:,:,1);
M=255-Color(:,:,2);
Y=255-Color(:,:,3);
/////////////////////  Color Balance Correction in
   Cyan Component //////////////////////
D=0:1/256:1;
H1=1*D^2.5;  // Transfer Function (Design from the
   Gamma Funcetion).
H2=1*D^0.5;  // Transfer Function (Design from the
   Gamma Funcetion).
figure,subplot(211),plot(H1);
xlabel('Intensity');
ylabel('Magnitude');
title('HSI Intensity Transformation Function(Heavy
```

```
in Cyan)');
subplot(212),plot(H2);
xlabel('Intensity');
ylabel('Magnitude');
title('HSI Intensity Transformation Function (Weak in Cyan)','color','blue','fontsize',4);

C_Modify=[];
for i=1:nr
    for j=1:nc
            C_Modify1(i,j,1)=H1(uint16(C(i,j,1))+1);
            C_Modify2(i,j,1)=H2(uint16(C(i,j,1))+1);
    end
end
Balance_Image1(:,:,1)=C_Modify1;
Balance_Image1(:,:,2)=M;
Balance_Image1(:,:,3)=Y;
figure,ShowColorImage(Balance_Image1,'RGB Image');
title('Color Balanced image','color','blue','fontsize',4);

Balance_Image2(:,:,1)=C_Modify2;
Balance_Image2(:,:,2)=M;
Balance_Image2(:,:,3)=Y;
figure,ShowColorImage(Balance_Image2,'RGB Image');
title('Color Balanced image','color','blue','fontsize',4);

/////////////////////  Color Balance Correction in Megenta Component /////////////////////
D=0:1/256:1;
H1=1*D^2.5;  // Transfer Function (Design from the Gamma Funcetion).
H2=1*D^0.5;  // Transfer Function (Design from the Gamma Funcetion).
figure,subplot(211),plot(H1);
xlabel('Intensity');
ylabel('Magnitude');
```

```
title('HSI Intensity Transformation Function(Heavy in Megenta)','color','blue','fontsize',4);
subplot(212),plot(H2);
xlabel('Intensity');
ylabel('Magnitude');
title('HSI Intensity Transformation Function (Weak in Megenta)','color','blue','fontsize',4);
for i=1:nr
    for j=1:nc
            Y_Modify1(i,j,1)=H1(uint16(Y(i,j,1))+1);
            Y_Modify2(i,j,1)=H2(uint16(Y(i,j,1))+1);
    end
end
Balance_Image1(:,:,1)=255-C;
Balance_Image1(:,:,2)=255-M;
Balance_Image1(:,:,3)=255-Y_Modify1;
figure,ShowColorImage(Balance_Image1,'RGB Image');
title('Color Balanced image','color','blue','fontsize',4);

Balance_Image2(:,:,1)=255-C;
Balance_Image2(:,:,2)=255-M;
Balance_Image2(:,:,3)=255-Y_Modify2;
figure,ShowColorImage(Balance_Image2,'RGB Image');
title('Color Balanced image','color','blue','fontsize',4);

////////////////////// Color Balance Correction in Yellow Component //////////////////////
D=0:1/256:1;
H1=1*D^2.5; // Transfer Function (Design from the Gamma Funcetion).
H2=1*D^0.5; // Transfer Function (Design from the Gamma Funcetion).
figure,subplot(211),plot(H1);
xlabel('Intensity');
ylabel('Magnitude');
title('HSI Intensity Transformation Function(Heavy
```

```
    in Yellow)','color','blue','fontsize',4);
subplot(212),plot(H2);
xlabel('Intensity');
ylabel('Magnitude');
title('HSI Intensity Transformation Function (Weak
    in Yellow)','color','blue','fontsize',4);
for i=1:nr
    for j=1:nc
        M_Modify1(i,j,1)=H1(uint16(M(i,j,1))+1);
        M_Modify2(i,j,1)=H2(uint16(M(i,j,1))+1);
    end
end
Balance_Image1(:,:,1)=255-C;
Balance_Image1(:,:,2)=255-M_Modify1;
Balance_Image1(:,:,3)=255-Y;
figure,ShowColorImage(Balance_Image1,'RGB Image');
title('Color Balanced image','color','blue','
    fontsize',4);

Balance_Image2(:,:,1)=255-C;
Balance_Image2(:,:,2)=255-M_Modify2;
Balance_Image2(:,:,3)=255-Y;
figure,ShowColorImage(Balance_Image2,'RGB Image');
title('Color Balanced image','color','blue','
    fontsize',4);
```

check Appendix ?? for dependency:

`Ex6_11.tif`

Scilab code Exa 6.11 `Histogram Equalization in the HSI Color Space`

```
//Ex6_11 :
//Histogram Equalization in the HSI Color Space

// Version : Scilab 5.4.1
```

```
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).

////////////////// Tonal Correction for the Flat Image ////////////////////////
Color=imread("Ex6_11.tif");
Color=imresize(Color,0.5);
[nr nc]=size(Color);
figure,ShowColorImage(Color,'Gray Image');
title('Original Image','color','blue','fontsize',4);

HSI=rgb2hsv(Color);
figure,ShowImage(HSI(:,:,3),'Gray Image');
title('Original Image');
[count cell]=imhist(HSI(:,:,3));
figure,bar(cell,count,0.2);

[P Q]=size(Color);
r=cell';  // Transpose of matrix
nk=round(count)';   // Transpose of matrix
M=sum(nk);
probeblity_r=nk/M;  // Probablity calculation
for i=1:length(r)
    sum_1=0;
    for j=1:i
        sum_1=sum_1+probeblity_r(j);
    end
    s(i)=max(r)*sum_1;
```

```
end
s=round(s); // Rounding Approach
disp(s);
[nr nc]=size(s);
temp=s';     // Transpose of matrix
for i=1:P      // Intensity Replacement in Original
    Image
    for j=1:Q
        b(i,j)=temp(double(HSI(i,j,3))+1);
    end
end
HSI(:,:,3)=b(:,:);
Color1=hsv2rgb(HSI);
figure,ShowColorImage(Color1,'histogram Equlized
    Image');
title('histogram Equlized Image','color','blue','
    fontsize',4);
```

check Appendix ?? for dependency:

`Ex6_12.tif`

Scilab code Exa 6.12 `Color Image Smoothning by Neighbourhood Averaging`

```
//Ex6_12
//Color Image Smoothning by Neighbourhood Averaging.

// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E.
    Woods

clc;
```

```
close;
clear;
xdel(winsid())//to close all currently open figure(s).
rgb=imread("Ex6_12.tif");
[nr nc]=size(rgb2gray(rgb));  // find the size of image

figure,ShowColorImage(rgb,'Gray Image');
title('Original Image','color','blue','fontsize',4);

R=rgb(:,:,1);//Separation of red component from image
figure,ShowImage(R,'Red component separation from original image');//ShowColorImage() is used to show color image, figure is command to view images in separate window.
title('Red component separation from original image','color','blue','fontsize',4);//title() is used for providing a title to an image.
G=rgb(:,:,2);//Separation of green component from image
figure,ShowImage(G,'Green comonent separation from original image');//ShowColorImage() is used to show color image, figure is command to view images in separate window.
title('Green component separation from original image','color','blue','fontsize',4);//title() is used for providing  a title to an image.
B=rgb(:,:,3);//Separation of blue component from image
figure,ShowImage(B,'Blue component separation from original image');//ShowColorImage() is used to show color image, figure is command to view images in separate window.
title('Blue component separation from original image','color','blue','fontsize',4);//title() is used for providing  a title to an image.
```

```

HSI=rgb2hsv(rgb);
H=HSI(:,:,1);//Separation of Hue component from image
figure,ShowImage(H,'Red component separation from original image');//ShowColorImage() is used to show color image, figure is command to view images in separate window.
title('Red component separation from original image','color','blue','fontsize',4);//title() is used for providing a title to an image.
S=HSI(:,:,2);//Separation of Saturation component from image
figure,ShowImage(S,'Green comonent separation from original image');//ShowColorImage() is used to show color image, figure is command to view images in separate window.
title('Green component separation from original image','color','blue','fontsize',4);//title() is used for providing  a title to an image.
I=HSI(:,:,3);//Separation of Intensity component from image
figure,ShowImage(I,'Blue component separation from original image');//ShowColorImage() is used to show color image, figure is command to view images in separate window.
title('Blue component separation from original image','color','blue','fontsize',4);//title() is used for providing  a title to an image.

mask=fspecial('average',5);
Filtered_Image1=imfilter(rgb,mask);
figure,ShowColorImage(Filtered_Image1,'Average Color image');//ShowColorImage() is used to show color image, figure is command to view images in separate window.
title('RGB image after Smoothing [5*5]','color','blue','fontsize',4);//title() is used for
```

```
providing  a title to an image.


HSI(:,:,3)=imfilter(I,mask);
Filtered_Image2=hsv2rgb(HSI);
figure,ShowColorImage(Filtered_Image2,'Average Color image');//ShowColorImage() is used to show color image, figure is command to view images in separate window.
title('RGB image after Smoothing Intensity Component [5*5]','color','blue','fontsize',4);//title() is used for providing  a title to an image.
gray1=im2double(rgb2gray(Filtered_Image1));
gray2=rgb2gray(Filtered_Image2);
difference=gray1-gray2;
//difference=imsubtract(rgb2gray(Filtered_Image1),rgb2gray(Filtered_Image2));
//difference=im2double(Filtered_Image1)-Filtered_Image2;
figure,ShowImage(difference,'Difference Color image');//ShowColorImage() is used to show color image, figure is command to view images in separate window.
title('Image after Subtraction','color','blue','fontsize',4);//title() is used for providing  a title to an image.
```

check Appendix ?? for dependency:

Ex6_13.tif

Scilab code Exa 6.13 Sharpning with the Laplacian

```
//Ex6_13
//Sharpning with the Laplacian
// Version : Scilab 5.4.1
```

```
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).
rgb=imread("Ex6_13.tif");
[nr nc]=size(rgb2gray(rgb));  // find the size of image
//figure,ShowColorImage(rgb,'Gray Image');
//title('Original Image');

R=rgb(:,:,1);//Separation of red component from image
G=rgb(:,:,2);//Separation of green component from image
B=rgb(:,:,3);//Separation of blue component from image
mask=fspecial('laplacian'); // Generate laplacian mask
Filtered_Image1(:,:,1)=imfilter(R,mask);
Filtered_Image1(:,:,2)=imfilter(G,mask);
Filtered_Image1(:,:,3)=imfilter(B,mask);
figure,ShowColorImage(Filtered_Image1,'Average Color image');//ShowColorImage() is used to show color image, figure is command to view images in separate window.
title('RGB image after Sharpning','color','blue','fontsize',4);// title() is used for providing a title to an image.

HSI=rgb2hsv(rgb);
```

```
H=HSI(:,:,1);//Separation of Hue component from image
S=HSI(:,:,2);//Separation of Saturation component from image
I=HSI(:,:,3);//Separation of Intensity component from image
HSI(:,:,3)=imfilter(I,mask);
Filtered_Image2=hsv2rgb(HSI); // Convert HSI to RGB Image
figure,ShowColorImage(Filtered_Image2,'Average Color image');//ShowColorImage() is used to show color image, figure is command to view images in separate window.
title('RGB image after Sharpning Intensity Component','color','blue','fontsize',4);//title() is used for providing a title to an image.
gray1=im2double(rgb2gray(Filtered_Image1));
gray2=rgb2gray(Filtered_Image2);
difference=gray1-gray2; // Difference Image
figure,ShowImage(difference,'Difference Color image');//ShowColorImage() is used to show color image, figure is command to view images in separate window.
title('Image after Subtraction','color','blue','fontsize',4);//title() is used for providing a title to an image.
```

---

check Appendix ?? for dependency:

`Ex6_14.tif`

Scilab code Exa 6.14 `Segmentation in HSI Space`

```
//Ex6_14
//Segmentation in HSI Space
// Version : Scilab 5.4.1
```

```
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).
rgb=imread("Ex6_14.tif");
[nr nc]=size(rgb2gray(rgb));  // find the size of image
//figure,ShowColorImage(rgb,'Gray Image');
//title('Original Image','color','blue','fontsize',4);
//
HSI=rgb2hsv(rgb);
H=HSI(:,:,1);//Separation of Hue component from image
figure,ShowImage(H,'Gray Image');
title('Hue Component','color','blue','fontsize',4);
S=HSI(:,:,2);//Separation of Saturation component from image
figure,ShowImage(S,'Saturation Component');
title('Saturation Component','color','blue','fontsize',4);
I=HSI(:,:,3);//Separation of Intensity component from image
figure,ShowImage(I,'Intensity Component');
title('Intensity Component','color','blue','fontsize',4);

S_Max=max(S); // Calculate Maximum Value
thresh=0.35;
S_threshold=im2bw(S,thresh);  // used for
```

```
Binarization
//S_threshold = imcomplement(S_threshold)
figure,ShowImage(S_threshold,'Binary Image');
title('Binary Saturation Mask','color','blue','fontsize',4);

temp=H.*S_threshold;
figure,ShowImage(temp,'Binary Image');
title('Binary Saturation Mask with Multiplication','color','blue','fontsize',4);

[count cell]=imhist(temp);
figure,bar(cell,count,0.2);
title('Histogram','color','blue','fontsize',4);
thresh=0.9;
temp_threshold=im2bw(temp,thresh);
figure,ShowImage(temp_threshold,'Binary Image');
title('Segmentation of Red Component','color','blue','fontsize',4);
```

---

check Appendix ?? for dependency:

`Ex6_16.tif`

Scilab code Exa 6.16 Edge Detection Vector Space

```
//Ex6_16
//Edge Detection Vector Space
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

```

```
clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).
rgb=imread("Ex6_16.tif");
[nr nc]=size(rgb2gray(rgb));  // find the size of image
figure,ShowColorImage(rgb,'Color Image');
title('Original Image','color','blue','fontsize',4);

R=rgb(:,:,1); //Separation of red component from image
G=rgb(:,:,2); //Separation of green component from image
B=rgb(:,:,3); //Separation of blue component from image

Image_Edge=edge(R,'canny',0.18);  // Gradient Computation by Canny
figure,ShowImage(Image_Edge,'Edge Image');
title('Gradient Image','color','blue','fontsize',4);

Image_Edge=edge(G,'canny',0.17);  // Gradient Computation by Canny
figure,ShowImage(Image_Edge,'Edge Image');
title('Gradient Image','color','blue','fontsize',4);

Image_Edge=edge(B,'canny',0.19);  // Gradient Computation by Canny
figure,ShowImage(Image_Edge,'Edge Image');
title('Gradient Image','color','blue','fontsize',4);
```

check Appendix ?? for dependency:

`Ex6_17_B.tif`

check Appendix ?? for dependency:

`Ex6_17_G.tif`

check Appendix ?? for dependency:

`Ex6_17_R.tif`

Scilab code Exa 6.17 `Illustration of the effects of converting noisy RGB Images to`

```
//Ex6_17
//Illustration of the effects of converting noisy RGB Images to HSI
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).
R=rgb2gray(imread("Ex6_17_R.tif"));
G=rgb2gray(imread("Ex6_17_G.tif"));
B=rgb2gray(imread("Ex6_17_B.tif"));

figure,ShowImage(R,'Red Component');
title('Red Component','color','blue','fontsize',4);
figure,ShowImage(G,'Green Component');
title('Green Component','color','blue','fontsize',4);
figure,ShowImage(B,'Blue Component');
title('Blue Component','color','blue','fontsize',4);

rgb(:,:,1)=R; //Merging of Red component from image
```

```
rgb(:,:,2)=G; //Merging of Green component from image
rgb(:,:,3)=B; //Merging of Blue component from image

figure,ShowColorImage(rgb,'Color Image');
title('Color Image','color','blue','fontsize',4);

HSI=rgb2hsv(rgb);
figure,ShowImage(HSI(:,:,1),'Hue Image');
title('Hue Component','color','blue','fontsize',4);
figure,ShowImage(HSI(:,:,2),'Saturation Image');
title('Saturation Component','color','blue','fontsize',4);
figure,ShowImage(HSI(:,:,3),'Intensity Image');
title('Intensity Component','color','blue','fontsize',4);


G=imnoise(G,'salt & pepper',0.05);
rgb(:,:,2)=G; //Merging of Green component from image
figure,ShowColorImage(rgb,'Color Image');
title('Color Image with Salt & Pepper Niose in Green Component','color','blue','fontsize',4);
HSI=rgb2hsv(rgb);
figure,ShowImage(HSI(:,:,1),'Hue Image');
title('Hue Component','color','blue','fontsize',4);
figure,ShowImage(HSI(:,:,2),'Saturation Image');
title('Saturation Component','color','blue','fontsize',4);
figure,ShowImage(HSI(:,:,3),'Intensity Image');
title('Intensity Component','color','blue','fontsize',4);
```

Chapter 8

Image Compression

check Appendix **??** for dependency:

`Ex8_2.tif`

Scilab code Exa 8.2 `Image Entropy Estimation`

```
//Ex8_2
//Image Entropy Estimation
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).
A=imread("Ex8_2.tif");
```

```
figure,ShowImage(A,'Original Image');
title('Original Image','color','blue','fontsize',4);
[nr nc]=size(A);
[Count Cell]=imhist(A);
//figure,bar(Cell,Count);
[r c]=find(Count>0);
Probablity=Count(r)/(nr*nc); //Probablity Calculation
//disp(Probablity);
Intensity=Cell(r);
//disp(Intensity);

Sum=0;
for i=1:length(r)
    p=Probablity(i);
    Sum=Sum+(-p*log2(p));
end

disp('Entropy')
disp(Sum);
```

Chapter 9

Morphological Image Processing

check Appendix **??** for dependency:

Ex9_1.tif

Scilab code Exa 9.1 Using Erosion to remove image component

```
//Ex9_1
//Using Erosion to remove image component
// Version : Scilab 5.4.1
// Operating System : Window−xp, Window−7
//Toolbox: Image Processing Design 8.3.1−1
//Toolbox: SIVP 0.5.3.1−2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E.
    Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s
    ).
```

```
//////////////////// Tonal Correction for the Flat
    Image ////////////////////////
Image=imread("Ex9_1.tif");
//Color=imresize(Color,0.25);
[nr nc]=size(Image);
figure,ShowImage(Image,'Gray Image');
title('Binary Image of Wire Bond [486*486]','color',
    'blue','fontsize',4);
Mask=CreateStructureElement('square',11);
Image_Eroide=ErodeImage(Image,Mask);
figure,ShowImage(Image_Eroide,'Eriode Image');
title('Eriode Image with 11*11 Square Mask','color',
    'blue','fontsize',4);

Mask=CreateStructureElement('square',15);
Image_Eroide=ErodeImage(Image,Mask);
figure,ShowImage(Image_Eroide,'Eriode Image');
title('Eriode Image with 15*15 Square Mask','color',
    'blue','fontsize',4);

Mask=CreateStructureElement('square',45);
Image_Eroide=ErodeImage(Image,Mask);
figure,ShowImage(Image_Eroide,'Eriode Image');
title('Eriode Image with 45*45 Square Mask','color',
    'blue','fontsize',4);
```

check Appendix **??** for dependency:

`Ex9_2.tif`

Scilab code Exa 9.2 `An Illustration of Dilation`

```
//Ex9_2
//An Illustration of Dilation
// Version : Scilab 5.4.1
```

```
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E.
    Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s
    ).

Image=imread("Ex9_2.tif");
//Color=imresize(Color,0.25);
[nr nc]=size(Image);
figure,ShowImage(Image,'Gray Image');
title('Original Image','color','blue','fontsize',4);

Mask.Width=3;
Mask.Height=3;
Mask.Data=[%F %T %F;%T %T %T;%F %T %F];

//Mask=[0 1 0;1 1 1;0 1 0];
Image_Eroide=ErodeImage(Image,Mask);
figure,ShowImage(Image_Eroide,'Eriode Image');
title('Eriode Image with 3*3 Square Mask','color','
    blue','fontsize',4);
```

check Appendix ?? for dependency:

`Ex9_4.png`

Scilab code Exa 9.4 `Use of opening and closing for Morphological Filtering`

```
//Ex9_4
```

```
//Use of opening and closing for Morphological Filtering
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).

Color=imread("Ex9_4.png");
Image=imresize(rgb2gray(Color),2,'bicubic');
Image=im2bw(Image,0.75);
[nr nc]=size(Image);
figure,ShowImage(Image,'Gray Image');
title('Noisy Image','color','blue','fontsize',4);

Mask=CreateStructureElement('square',3);  // Create Structuring Element
Image_Eroide=ErodeImage(Image,Mask);  // Erosion Operation
figure,ShowImage(Image_Eroide,'Eriode Image');
title('Eriode Image with 3*3 Square Mask','color','blue','fontsize',4);

Image_Open=OpenImage(Image,Mask);   // Opening Operation
figure,ShowImage(Image_Open,'Open Image');
title('Opening Image with 3*3 Square Mask','color','blue','fontsize',4);

Image_Dilate=DilateImage(Image_Open,Mask);  //
```

```
Dilusion of Open Image
figure,ShowImage(Image_Dilate,'Dilate Image');
title('Dilate Image with 3*3 Square Mask','color','blue','fontsize',4);

Image_Close=CloseImage(Image_Dilate,Mask);    //Opening Operation
figure,ShowImage(Image_Close,'Closing Image');
title('Closing Image with 3*3 Square Mask','color','blue','fontsize',4);
```

check Appendix ?? for dependency:

Ex9_5.png

Scilab code Exa 9.5 Boundary Extraction by Morphological Processing

```
//Ex9_5
//Boundary Extraction by Morphological Processing
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).

Color=imread("Ex9_5.png");
Image=rgb2gray(Color);
Image=im2bw(Image,0.75);
```

```
[nr nc]=size(Image);
figure,ShowImage(Image,'Binary Image');
title('Binary Image','color','blue','fontsize',4);

Mask=CreateStructureElement('square',5);  // Create Structuring Element
Image_Eroide=ErodeImage(Image,Mask);  // Erosion Operation
Image_Boundray=Image-Image_Eroide;
//Image_Open=OpenImage(Image,Mask);    // Opening Operation
figure,ShowImage(Image_Boundray,'Boundray Image');
title('Boundray Image Extracted Image with Morphological Processing ','color','blue','fontsize',4);
```

check Appendix ?? for dependency:

Ex9_7.png

Scilab code Exa 9.7 Using Connected Components to Detect Foreign Object in Package

```
//Ex9_7
//Using Connected Components to Detect Foreign Object in Packaged Food
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

clc;
close;
clear;
```

```
xdel(winsid())//to close all currently open figure(s).

Color=imread("Ex9_7.png");
Image=rgb2gray(Color);
//Image=im2bw(Image,0.65);
[nr nc]=size(Image);
figure,ShowImage(Image,'Binary Image');
title('Binary Image','color','blue','fontsize',4);

Image_Binary=im2bw(Image,0.825);  // Binarization Process with Specific Threshold
figure,ShowImage(Image_Binary,'Binary Image');
title('Binary Image','color','blue','fontsize',4);

Mask=CreateStructureElement('square',3);  // Create Structuring Element
Image_Eroide=ErodeImage(Image_Binary,Mask);  // Erosion Operation
figure,ShowImage(Image_Eroide,'Eroide Image');
title('Eroide Image ','color','blue','fontsize',4);

BlobImage=SearchBlobs(Image_Eroide);  // Connected Component labelling
IsCalculated = CreateFeatureStruct(%f); // Feature struct is generated.
IsCalculated.PixelList = %t; // The bounding box shall be calculated for each blob.
BlobStatistics = AnalyzeBlobs(BlobImage, IsCalculated);

Blob_Total=max(BlobImage);
Blob_Area=[];
for i=1:Blob_Total
    temp=size(BlobStatistics(i).PixelList);
    Blob_Area=[Blob_Area temp(1)];
end

```

```
disp(Blob_Area',"Blob Area")
```

check Appendix ?? for dependency:

`Ex9_9.png`

Scilab code Exa 9.9 `Illustration of Gray Scale Erosion and Dilation`

```
//Ex9_9
// Illustration of Gray Scale Erosion and Dilation
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E.
   Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s
   ).

function [f]=restoration_filter(v,type,m,n,Q,d)
    if argn(2) ==2
        m=7;n=7;Q=1.5;d=10;
    elseif argn(2)==5
        Q=parameter;d=parameter;
    elseif argn(2)==4
        Q=1.5;d=2;
    else
        disp('wrong number of inputs');
    end

    select type
```

```
    case 'median'
        f=MedianFilter(v,[m n]);

    case 'MIN'
       size1=m;
       [nr,nc]=size(v);
       temp=zeros(nr+2*floor(size1/2),nc+2*floor(
          size1/2));
       temp(ceil(size1/2):nr+ceil(size1/2)-1,ceil(
          size1/2):nc+ceil(size1/2)-1)=v(1:$,1:$);
        for i=ceil(size1/2):nr+ceil(size1/2)-1
            for j=ceil(size1/2):nc+ceil(size1/2)-1
                t=temp(i-floor(size1/2):1:i+floor(
                   size1/2),j-floor(size1/2):1:j+
                   floor(size1/2)) ;
                y=gsort(t);
                temp2(i-floor(size1/2),j-floor(size1
                   /2))=min(y);
            end
        end
        f=mat2gray(temp2);

    case 'MAX'
       size1=m;
       [nr,nc]=size(v);
       temp=zeros(nr+2*floor(size1/2),nc+2*floor(
          size1/2));
       temp(ceil(size1/2):nr+ceil(size1/2)-1,ceil(
          size1/2):nc+ceil(size1/2)-1)=v(1:$,1:$);
        for i=ceil(size1/2):nr+ceil(size1/2)-1
            for j=ceil(size1/2):nc+ceil(size1/2)-1
                t=temp(i-floor(size1/2):1:i+floor(
                   size1/2),j-floor(size1/2):1:j+
                   floor(size1/2)) ;
                y=gsort(t);
                temp2(i-floor(size1/2),j-floor(size1
                   /2))=max(y);
```

```
                end
            end
            f=mat2gray(temp2);

            case'Mid_Point'
           size1=m;
           [nr,nc]=size(v);
           temp=zeros(nr+2*floor(size1/2),nc+2*floor(size1/2));
           temp(ceil(size1/2):nr+ceil(size1/2)-1,ceil(size1/2):nc+ceil(size1/2)-1)=v(1:$,1:$);
            for i=ceil(size1/2):nr+ceil(size1/2)-1
                for j=ceil(size1/2):nc+ceil(size1/2)-1
                    t=temp(i-floor(size1/2):1:i+floor(size1/2),j-floor(size1/2):1:j+floor(size1/2)) ;
                    y=gsort(t);
                    temp2(i-floor(size1/2),j-floor(size1/2))=0.5*(min(y)+max(y));
                end
            end
            f=mat2gray(temp2);

            else
            disp('Unknownfiltertype.')
        end
endfunction

//////////////////////////////////////      Main Programm     ////////////////////

a=imread("Ex9_9.png");
gray=rgb2gray(a);
//gray=im2double(gray);
figure,ShowImage(gray,'Gray Image');
title('Original X-Ray Image','color','blue','fontsize',4);
[M,N]=size(gray);
```

```

//////////////////////////////////// MIN Filter ////////////////////
h=restoration_filter(gray,'MIN',3,3);
figure,ShowImage(h,'Recovered Image');
title('Erosion using Flat Structuring Element','color','blue','fontsize',4);


//////////////////////////////////// MAX Filter ////////////////////
h=restoration_filter(gray,'MAX',3,3);
figure,ShowImage(h,'Recovered Image');
title('Dilation using Flat Structuring Element','color','blue','fontsize',4);
```

check Appendix ?? for dependency:

`Ex9_10.png`

Scilab code Exa 9.10 `Illustration of Gray Scale Opening and Closing`

```
//Ex9_10
// Illustration of Gray Scale Opening and Closing
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

clc;
close;
clear;
```

```
xdel(winsid())//to close all currently open figure(s).

function [f]=restoration_filter(v,type,m,n,Q,d)
    if argn(2) ==2
        m=7;n=7;Q=1.5;d=10;
    elseif argn(2)==5
        Q=parameter;d=parameter;
    elseif argn(2)==4
        Q=1.5;d=2;
    else
        disp('wrong number of inputs');
    end

    select type

    case'median'
        f=MedianFilter(v,[m n]);

    case'MIN'
       size1=m;
       [nr,nc]=size(v);
       temp=zeros(nr+2*floor(size1/2),nc+2*floor(size1/2));
       temp(ceil(size1/2):nr+ceil(size1/2)-1,ceil(size1/2):nc+ceil(size1/2)-1)=v(1:$,1:$);
        for i=ceil(size1/2):nr+ceil(size1/2)-1
            for j=ceil(size1/2):nc+ceil(size1/2)-1
                t=temp(i-floor(size1/2):1:i+floor(size1/2),j-floor(size1/2):1:j+floor(size1/2)) ;
                y=gsort(t);
                temp2(i-floor(size1/2),j-floor(size1/2))=min(y);
            end
        end
        f=mat2gray(temp2);

```

```
    case'MAX'
        size1=m;
        [nr,nc]=size(v);
        temp=zeros(nr+2*floor(size1/2),nc+2*floor(size1/2));
        temp(ceil(size1/2):nr+ceil(size1/2)-1,ceil(size1/2):nc+ceil(size1/2)-1)=v(1:$,1:$);
         for i=ceil(size1/2):nr+ceil(size1/2)-1
             for j=ceil(size1/2):nc+ceil(size1/2)-1
                 t=temp(i-floor(size1/2):1:i+floor(size1/2),j-floor(size1/2):1:j+floor(size1/2)) ;
                 y=gsort(t);
                 temp2(i-floor(size1/2),j-floor(size1/2))=max(y);
             end
         end
         f=mat2gray(temp2);

         case'Mid_Point'
        size1=m;
        [nr,nc]=size(v);
        temp=zeros(nr+2*floor(size1/2),nc+2*floor(size1/2));
        temp(ceil(size1/2):nr+ceil(size1/2)-1,ceil(size1/2):nc+ceil(size1/2)-1)=v(1:$,1:$);
         for i=ceil(size1/2):nr+ceil(size1/2)-1
             for j=ceil(size1/2):nc+ceil(size1/2)-1
                 t=temp(i-floor(size1/2):1:i+floor(size1/2),j-floor(size1/2):1:j+floor(size1/2)) ;
                 y=gsort(t);
                 temp2(i-floor(size1/2),j-floor(size1/2))=0.5*(min(y)+max(y));
             end
         end
         f=mat2gray(temp2);

```

```
        else
        disp('Unknownfiltertype.')
    end
endfunction

//////////////////////////////////////      Main Programm     ////////////////////

a=imread("Ex9_10.png");
gray=rgb2gray(a);
//gray=im2double(gray);
figure,ShowImage(gray,'Gray Image');
title('Original X-Ray Image','color','blue','fontsize',4);
[M,N]=size(gray);

///////////////////////////// Gray Scale Opening ////////////////////
h=restoration_filter(restoration_filter(gray,'MIN',3,3),'MAX',3,3);
figure,ShowImage(h,'Recovered Image');
title('Opening using Flat Structureing Element','color','blue','fontsize',4);


//////////////////////////////// Gray Scale Closing ////////////////////
h=restoration_filter(restoration_filter(gray,'MAX',3,3),'MIN',3,3);
figure,ShowImage(h,'Recovered Image');
title('Closing using Flat Structureing Element','color','blue','fontsize',4);
```

Chapter 10

Image Segmentation

check Appendix **??** for dependency:

`Ex10_1.tif`

Scilab code Exa 10.1 `Detection of Isolated Point in an Image`

```
//Ex10_1
// Detection of Isolated Point in an Image
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).

a=imread("Ex10_1.tif");
```

```
//gray=rgb2gray(a);
//a=im2double(a);
figure,ShowImage(a,'Gray Image');
title('Original X-Ray Image','color','blue','
    fontsize',4);
[M,N]=size(a);

Mask=[1 1 1;1 -8 1;1 1 1];
Filtered_Image=imfilter(a,Mask);
figure,ShowImage(Filtered_Image,'Filter Image');
title('Original X-Ray Image','color','blue','
    fontsize',4);

thresh=uint8(229.5);
disp(thresh);
image=im2bw(Filtered_Image,0.996);
figure,ShowImage(image,'Filter Image');
title('Detection of Isolated Point','color','blue','
    fontsize',4);
```

check Appendix ?? for dependency:

`Ex10_2.tif`

Scilab code Exa 10.2 Using the Laplacian for the Detection

```
//Ex10_2
// Using the Laplacian for the Detection
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E.
    Woods

```

```
clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s
    ).

a=imread("Ex10_2.tif");
figure,ShowImage(a,'Gray Image');
title('Wire Bond Template Image','color','blue','
    fontsize',4);
[M,N]=size(a);

Mask=[1 1 1;1 -8 1;1 1 1];  // Mask for the
    Lapalacian
Filtered_Image=imfilter(a,Mask);  // Filtering the
    Original Image with the Mask
figure,ShowImage(Filtered_Image,'Filter Image');
title('Laplacian Image','color','blue','fontsize',4)
    ;
```

check Appendix ?? for dependency:

Ex10_3.tif

Scilab code Exa 10.3 Detection of Lines in Specified Direction

```
//Ex10_3
// Detection of Lines in Specified Direction
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
```

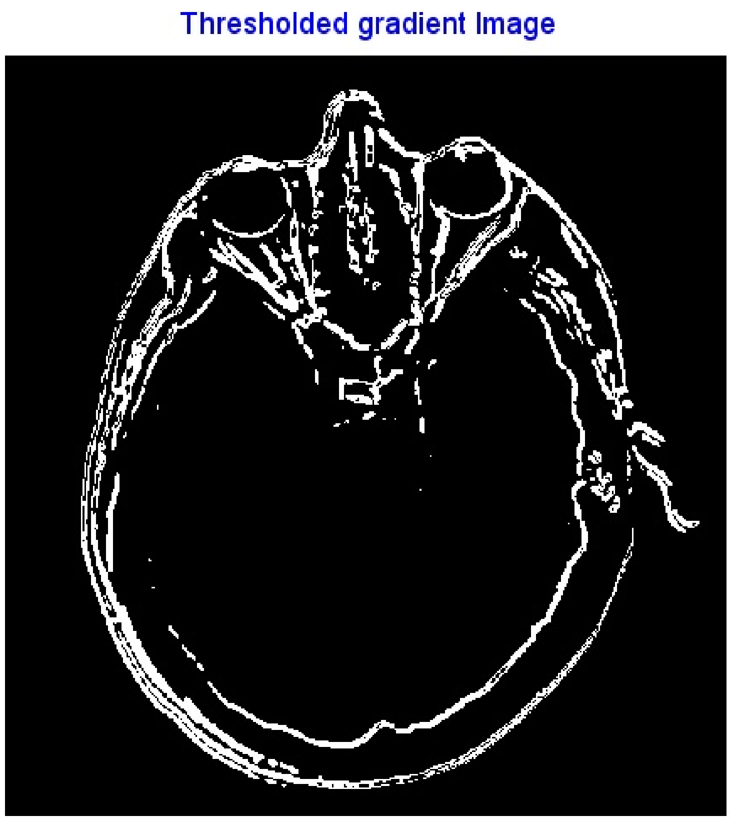


Figure 10.1: Using the Laplacian for the Detection

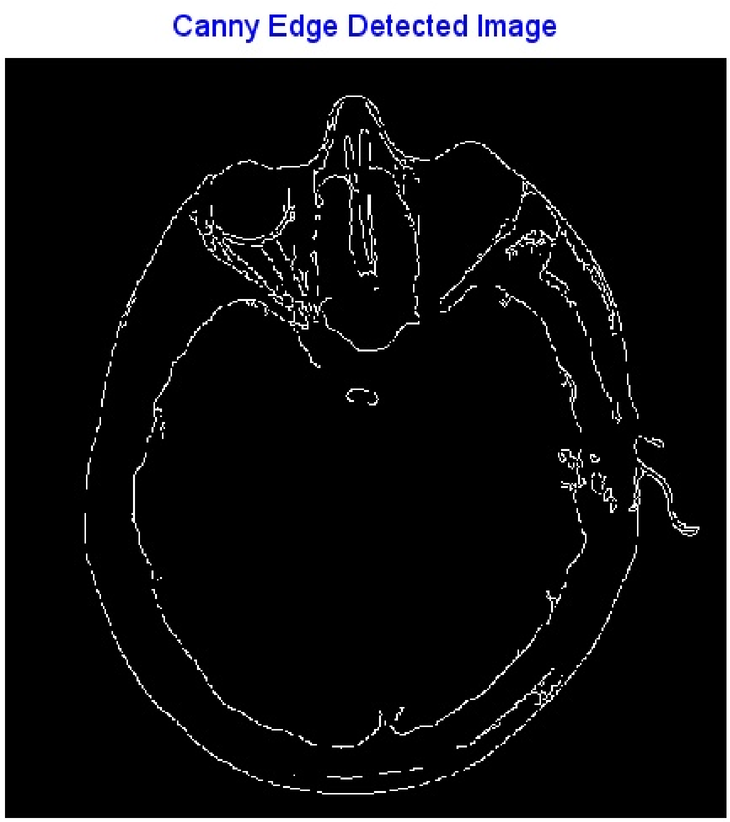


Figure 10.2: Using the Laplacian for the Detection

```
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E.
    Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s
    ).

a=imread("Ex10_2.tif");
figure,ShowImage(a,'Gray Image');
title('Wire Bond Template Image','color','blue','
    fontsize',4);
[M,N]=size(a);

Mask_Diagonal=[2 -1 -1;-1 2 -1;-1 -1 2];  // Mask
    for the +45 Line Detetion
Filtered_Image=imfilter(a,Mask_Diagonal);  //
    Filtering the Original Image with the Mask
figure,ShowImage(Filtered_Image,'Filter Image');
title('+45 Line Detected Image','color','blue','
    fontsize',4);
```

check Appendix ?? for dependency:

`Ex10_4.tif`

Scilab code Exa 10.4 Behavior of the First and Second Derivative of a Noisy Edge

```
//Ex10_4
// Behavior of the First and Second Derivative of a
    Noisy Edge
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
```

```
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).

a=imread("Ex10_4.tif");
a1=im2double(a);
figure,ShowImage(a,'Gray Image');
title('Original Image','color','blue','fontsize',4);
[M,N]=size(a);
first_order=zeros(M,N);
second_order=zeros(M,N);

for i=2:M-1
    for j=2:N-1
        first_order(i,j)=a(i,j+1)-a(i,j);
        second_order(i,j)=double(a1(i,j+1)+a1(i,j-1)-(2*a1(i,j)));
    end
end

figure,ShowImage(double(first_order),'First Order Difference Image');
title('First Order Difference Image','color','blue','fontsize',4);

forward_count=0;
reverse_count=0;
for j= 2:N-1    // Finding First Zero Crossing Point
    if(second_order(5,j)==0 &amp; second_order(5,j+1)&gt;0)
        forward_count=forward_count+1;
```

```
        if(forward_count==1)
            ther1=second_order(5,j+1);
            break;
        end
    end
end

for j= N-1:-1:2  // Finding Last Zero Crossing Point
    if(second_order(5,j)==0 &amp; second_order(5,j-1)&lt;0)
        reverse_count=reverse_count+1;
        if(reverse_count==1)
            ther2=second_order(5,j-1);
            break;
        end
    end
end

for i=1:M  // Removing unwanted Intensity range
    for j=1:N
        if(second_order(i,j)==ther1)
            second_order1(i,j)=255;
        else if(second_order(i,j)==ther2)
            second_order1(i,j)=-255;
        else
            second_order1(i,j)=128;
        end
    end
end
end

figure,ShowImage(second_order1,'Second Order Difference Image');
title('Second Order Difference Image','color','blue','fontsize',4);

t=a(5,1:N);
t1=first_order(5,1:N);
```

```
t2=second_order1(5,1:N);
figure;
subplot(311);
plot(1:length(t),t);//
title('Intensity Profile','color','blue','fontsize',4);
subplot(312);
mtlb_axis([1,N,0,1.5]);
plot(1:length(t1),t1);//
title('Intensity Profile of First order Derivative','color','blue','fontsize',4);
subplot(313);
plot(1:length(t2),t2);//
title('Intensity Profile of Second order Derivative','color','blue','fontsize',4);
```

check Appendix ?? for dependency:

`Ex10_6.tif`

Scilab code Exa 10.6 `Illustration of the 2 D Gradient Magnitude and Angle`

```
//Ex10_6
// Illustration of the 2 D Gradient Magnitude and Angle
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

clc;
close;
clear;
```

```
xdel(winsid())//to close all currently open figure(s).

a=imread("Ex10_6.tif");
//a=im2double(a);
figure,ShowImage(a,'Gray Image');
title('Original Image','color','blue','fontsize',4);
[M,N]=size(a);

Mask=[-1 -2 -1;0 0 0;1 2 1];  // Mask for the Sobel
GradientX_Image=imfilter(a,Mask);  // Filtering the Original Image with the Mask
figure,ShowImage(GradientX_Image,'Filter Image');
title('Sobel X-direction Gradient Image','color','blue','fontsize',4);

Mask=[-1 0 1;-2 0 2;-1 0 1];  // Mask for the Sobel
GradientY_Image=imfilter(a,Mask);  // Filtering the Original Image with the Mask
figure,ShowImage(GradientY_Image,'Filter Image');
title('Sobel Y-direction Gradient Image','color','blue','fontsize',4);

Gradient=GradientX_Image+GradientY_Image;
figure,ShowImage(Gradient,'Filter Image');
title('Sobel X+Y Gradient Image','color','blue','fontsize',4);

//Alpha=atan(double(GradientY_Image),double(GradientX_Image));
//figure,ShowImage(Alpha,'Angle Image');
//title('Angle Image','color','blue','fontsize',4);
```

check Appendix ?? for dependency:

`Ex10_7.tif`

Scilab code Exa 10.7 Illustration of the Marr Hildreth Edge Detection Methods

```
//Ex10_7
// Illustration of the Marr-Hildreth Edge Detection Methods
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).

a=imread("Ex10_7.tif");
a=im2double(a);
figure,ShowImage(a,'Gray Image');
title('Original Image','color','blue','fontsize',4);
[M,N]=size(a);
sigma=4;
for i=1:25
    for j=1:25
        Mask(i,j)=[(i^2+j^2-(2*sigma^2))/sigma^4]*exp(-(i^2+j^2)/(2*sigma^2));  // Mask Generation
    end
end

Filter_Image=imfilter(a,Mask);  // Filtering the Original Image with the Mask
figure,ShowImage(Filter_Image,'Filter Image');
title('Laplacian of gaussian Image','color','blue','fontsize',4);
```

```
b=zeros(M,N);
temp=Filter_Image;
for i=2:M-1  // Zero Crossing Detection
    for j=2:N-1
        //temp=[Filter_Image(i-1:i+1,j-1:j+1)];
        if((temp(i-1,j-1)>0 & temp(i+1,j+1)<0) | (
           temp(i-1,j-1)<0 & temp(i+1,j+1)>0)) then
            b(i,j)=255;
        else if ((temp(i-1,j+1)>0 & temp(i+1,j-1)<0)
            | (temp(i-1,j+1)<0 & temp(i+1,j-1)>0))
           then
            b(i,j)=255;
        else if ((temp(i,j+1)>0 & temp(i,j-1)<0) | (
           temp(i,j+1)<0 & temp(i,j-1)>0)) then
            b(i,j)=255;
        else if ((temp(i-1,j)>0 & temp(i+1,j)<0) | (
           temp(i,j+1)<0 & temp(i,j-1)>0)) then
            b(i,j)=255;
        end
        end
        end
        end
    end
end

figure,ShowImage(b,'Zero Crossing Image');
title('Zero Crossing Detected Image','color','blue',
   'fontsize',4);
```

check Appendix ?? for dependency:

`Ex10_8.tif`

Scilab code Exa 10.8 `Illustration of the Canny Edge Detection Methods`

```
//Ex10_8
```

```
// Illustration of the Canny Edge Detection Methods
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).

a=imread("Ex10_8.tif");
//a=im2double(a);
figure,ShowImage(a,'Gray Image');
title('Original Image','color','blue','fontsize',4);
[M,N]=size(a);
 ///////////////// Threshlded Gradient of Smoothed Image ////////////////////////////////
a1=imfilter(a,fspecial('average',5));
Mask=[-1 -2 -1;0 0 0;1 2 1];  // Mask for the Sobel
GradientX_Image=imfilter(a1,Mask);  // Filtering the Original Image with the Mask
//figure,ShowImage(GradientX_Image,'Filter Image');
//title('Sobel X-direction Gradient Image','color','blue','fontsize',4);

Mask=[-1 0 1;-2 0 2;-1 0 1];  // Mask for the Sobel
GradientY_Image=imfilter(a1,Mask);  // Filtering the Original Image with the Mask
//figure,ShowImage(GradientY_Image,'Filter Image');
//title('Sobel Y-direction Gradient Image','color','blue','fontsize',4);

Gradient=GradientX_Image+GradientY_Image;
```

```
//figure,ShowImage(Gradient,'Filter Image');
//title('Sobel X+Y Gradient Image','color','blue','fontsize',4);

th=84;  // 33% of the Maximum Value in Gradient Image
[row col]=find(Gradient>84);
Gradient_Thresh=zeros(M,N);
for i=1:length(row)
    Gradient_Thresh(row(i),col(i))=255;
end
figure,ShowImage(Gradient_Thresh,'Filter Image');
title('Thresholded gradient Image','color','blue','fontsize',4);

/////////////////// Marr-Hildreth Edge Detection ////////////////////////////////////
a=im2double(a);
sigma=4;
for i=1:25
    for j=1:25
        Mask(i,j)=[(i^2+j^2-(2*sigma^2))/sigma^4]*exp(-(i^2+j^2)/(2*sigma^2));  // Mask Generation
    end
end

Filter_Image=imfilter(a,Mask);  // Filtering the Original Image with the Mask
//figure,ShowImage(Filter_Image,'Filter Image');
//title('Laplacian of gaussian Image','color','blue','fontsize',4);
b=zeros(M,N);
temp=Filter_Image;
for i=2:M-1  // Zero Crossing Detection
    for j=2:N-1
        //temp=[Filter_Image(i-1:i+1,j-1:j+1)];
        if((temp(i-1,j-1)>0 & temp(i+1,j+1)<0) | (
```

```
            temp(i-1,j-1)<0 & temp(i+1,j+1)>0)) then
                b(i,j)=255;
            else if ((temp(i-1,j+1)>0 & temp(i+1,j-1)<0) | (temp(i-1,j+1)<0 & temp(i+1,j-1)>0)) then
                b(i,j)=255;
            else if ((temp(i,j+1)>0 & temp(i,j-1)<0) | (temp(i,j+1)<0 & temp(i,j-1)>0)) then
                b(i,j)=255;
            else if ((temp(i-1,j)>0 & temp(i+1,j)<0) | (temp(i,j+1)<0 & temp(i,j-1)>0)) then
                b(i,j)=255;
            end
            end
            end
            end
    end
end
figure,ShowImage(b,'Zero Crossing Image');
title('Marr-Hildreth Edge Detected Image','color','blue','fontsize',4);

/////////////////////////// Canny Edge Detecedd Image ////////////////////////////////////////
a=imread("Ex10_8.tif");
E=edge(a,'canny',[0.15 0.60]);
figure,ShowImage(E,'Canny Image');
title('Canny Edge Detected Image','color','blue','fontsize',4);
```

check Appendix ?? for dependency:

`Ex10_9.tif`

Scilab code Exa 10.9 Another illustration of the three principal Edge Detection Me

```
//Ex10_9
// Another illustration of the three principal Edge Detection Methods
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).

a=imread("Ex10_9.tif");
//a=im2double(a);
figure,ShowImage(a,'Gray Image');
title('Original Image','color','blue','fontsize',4);
[M,N]=size(a);
 //////////////////  Threshlded Gradient of Smoothed Image  /////////////////////////////////
a1=imfilter(a,fspecial('average',5));
Mask=[-1 -2 -1;0 0 0;1 2 1];  // Mask for the Sobel
GradientX_Image=imfilter(a1,Mask);  // Filtering the Original Image with the Mask
//figure,ShowImage(GradientX_Image,'Filter Image');
//title('Sobel X-direction Gradient Image','color','blue','fontsize',4);

Mask=[-1 0 1;-2 0 2;-1 0 1];  // Mask for the Sobel
GradientY_Image=imfilter(a1,Mask);  // Filtering the Original Image with the Mask
//figure,ShowImage(GradientY_Image,'Filter Image');
//title('Sobel Y-direction Gradient Image','color','blue','fontsize',4);
```

```

Gradient=GradientX_Image+GradientY_Image;
//figure,ShowImage(Gradient,'Filter Image');
//title('Sobel X+Y Gradient Image','color','blue','fontsize',4);

th=84;  // 33% of the Maximum Value in Gradient Image
[row col]=find(Gradient>84);
Gradient_Thresh=zeros(M,N);
for i=1:length(row)
    Gradient_Thresh(row(i),col(i))=255;
end
figure,ShowImage(Gradient_Thresh,'Filter Image');
title('Thresholded gradient Image','color','blue','fontsize',4);

//////////////////// Marr-Hildreth Edge Detection ////////////////////////////////////
a=im2double(a);
sigma=3;
for i=1:19
    for j=1:19
        Mask(i,j)=[(i^2+j^2-(2*sigma^2))/sigma^4]*exp(-(i^2+j^2)/(2*sigma^2));  // Mask Generation
    end
end

Filter_Image=imfilter(a,Mask);  // Filtering the Original Image with the Mask
//figure,ShowImage(Filter_Image,'Filter Image');
//title('Laplacian of gaussian Image','color','blue','fontsize',4);
b=zeros(M,N);
temp=Filter_Image;
th=0.0021;
for i=2:M-1  // Zero Crossing Detection
```

```
    for j=2:N-1
        //temp=[Filter_Image(i-1:i+1,j-1:j+1)];
        if((temp(i-1,j-1)>th & temp(i+1,j+1)<th) | (temp(i-1,j-1)<th & temp(i+1,j+1)>th)) then
            b(i,j)=255;
        else if ((temp(i-1,j+1)>th & temp(i+1,j-1)<th) | (temp(i-1,j+1)<th & temp(i+1,j-1)>th)) then
            b(i,j)=255;
        else if ((temp(i,j+1)>th & temp(i,j-1)<th) | (temp(i,j+1)<th & temp(i,j-1)>th)) then
            b(i,j)=255;
        else if ((temp(i-1,j)>th & temp(i+1,j)<th) | (temp(i,j+1)<th & temp(i,j-1)>th)) then
            b(i,j)=255;
        end
        end
        end
        end
    end
end
figure,ShowImage(b,'Zero Crossing Image');
title('Marr-Hildreth Edge Detected Image','color','blue','fontsize',4);

////////////////////////// Canny Edge Detecedd Image //////////////////////////////////////
a=imread("Ex10_9.tif");
E=edge(a,'canny',[0.05 0.95]);
figure,ShowImage(E,'Canny Image');
title('Canny Edge Detected Image','color','blue','fontsize',4);
```

check Appendix ?? for dependency:

`Ex10_15.tif`

Scilab code Exa 10.15 Global Thresholding

```
//Ex10_15
// Global Thresholding
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E.
   Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s
   ).

a=imread("Ex10_15.tif");
a1=im2double(a);
figure,ShowImage(a,'Gray Image');
title('Noisy Finger Print','color','blue','fontsize'
   ,4);
[M,N]=size(a);

[count cell]=imhist(a);
figure,plot2d3(cell,count);
title('Histogram','color','blue','fontsize',4);

b=im2bw(a1,0.495);
figure,ShowImage(b,'Binary Image');
title('Segmented Result Using Global Threshold','
   color','blue','fontsize',4);
```

check Appendix **??** for dependency:

`Ex10_16.tif`

Scilab code Exa 10.16 `Optimum Global Thresholding using Otsu Method`

```
//Ex10_16
// Optimum Global Thresholding using Otsu's Method
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E.
   Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s
   ).

a=imread("Ex10_16.tif");
a1=im2double(a);
figure,ShowImage(a,'Gray Image');
title('Original Image','color','blue','fontsize',4);
[M,N]=size(a);

[count cell]=imhist(a);
figure,plot2d3(cell,count);
title('Histogram','color','blue','fontsize',4);

 //////////////////////////////// Global Threshold
    Approach //////////////////////////
th_Global=iterthresh(a1);
b1=im2bw(a1,th_Global);
```

```
figure,ShowImage(b1,'Binary Image');
title('Segmented Result Using Global Thresholding Algorithm','color','blue','fontsize',4);

/////////////////////////////// Otsu Method ///////////////////////////////

normal_hist=count/(M*N);
Sum=0;
cumu_mean=0;
for k=1:max(cell)+1
    Sum=Sum+normal_hist(k);
    P1(k)=Sum;
    cumu_mean=cumu_mean+(k*normal_hist(k));
    m(k)=cumu_mean;
    Mg=cumu_mean;
    sigma_B(k)=(((Mg*P1(k))-m(k))^2)/(%eps+(P1(k)*(1-P1(k))));
end

th_Otsu=find(sigma_B==max(sigma_B))+10;
b2=im2bw(a1,(th_Otsu/255));
figure,ShowImage(b2,'Binary Image');
title('Segmented Result Using Otsu Thresholding Algorithm','color','blue','fontsize',4);
```

check Appendix ?? for dependency:

`Ex10_18.tif`

Scilab code Exa 10.18 `Using Edge Information Based on the Laplacian to Improve Glo`

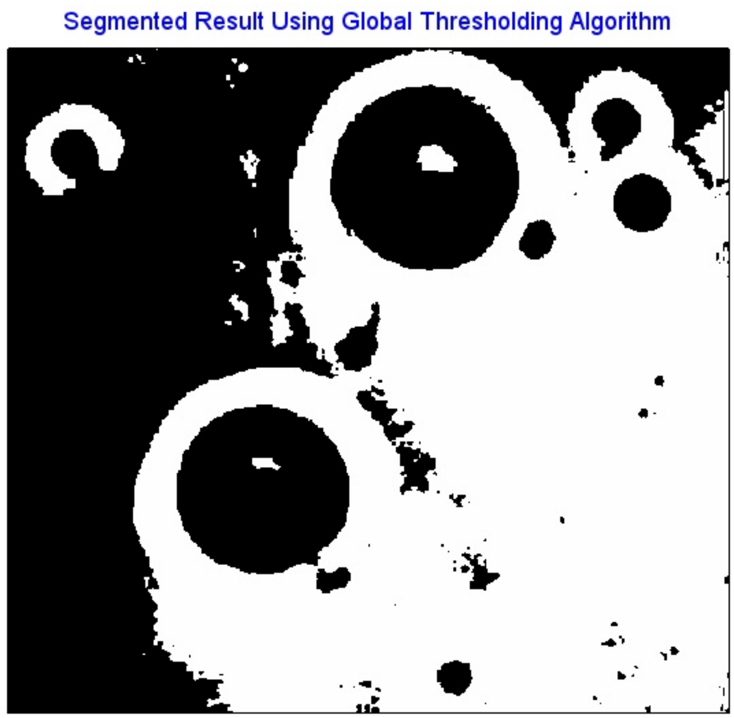


Figure 10.3: Optimum Global Thresholding using Otsu Method

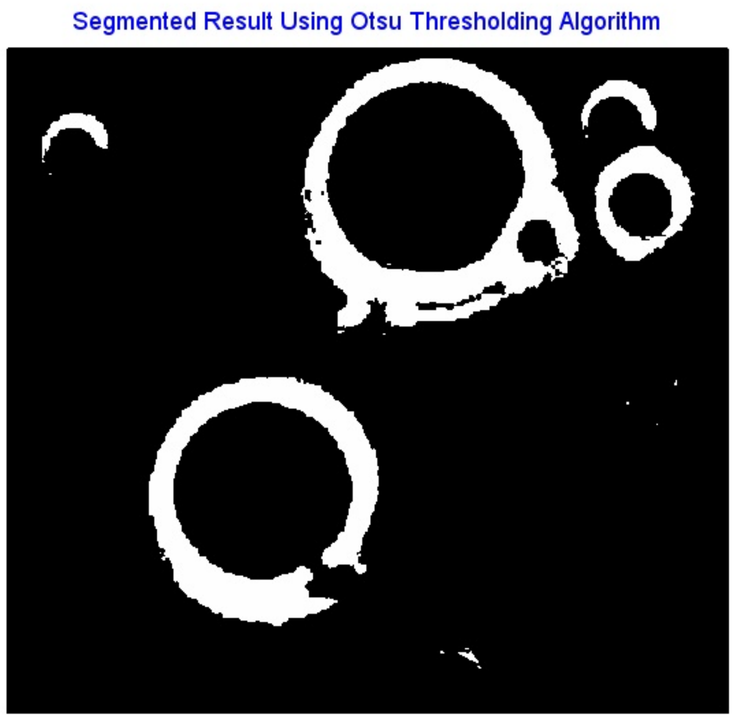


Figure 10.4: Optimum Global Thresholding using Otsu Method

```
//Ex10_18
// Using Edge Information Based on the Laplacian to Improve Global Thresholding
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).

a=imread("Ex10_18.tif");
a1=im2double(a);
figure,ShowImage(a,'Gray Image');
title('Original Image','color','blue','fontsize',4);
[M,N]=size(a);

[count cell]=imhist(a);
figure,plot2d3(cell,count);
title('Histogram','color','blue','fontsize',4);

//////////////////////////////// Otsu Method /////////////////////////////

normal_hist=count/(M*N);
Sum=0;
cumu_mean=0;
for k=1:max(cell)+1
    Sum=Sum+normal_hist(k);
    P1(k)=Sum;
    cumu_mean=cumu_mean+(k*normal_hist(k));
    m(k)=cumu_mean;
```

```
    Mg=cumu_mean;
    sigma_B(k)=(((Mg*P1(k))-m(k))^2)/(%eps+(P1(k)
       *(1-P1(k))));
end

th_Otsu=42;          //find(sigma_B==max(sigma_B));
b2=im2bw(a1,(th_Otsu/255));
figure,ShowImage(b2,'Binary Image');
title('Segmented Result Using Otsu Thresholding
    Algorithm','color','blue','fontsize',4);


mask=fspecial('laplacian');
c=abs(imfilter(a,mask));
figure,ShowImage(mat2gray(c),'Binary Image');
title('Laplacian Image','color','blue','fontsize',4)
    ;

//d=c.*a1;
//[count cell]=imhist(d);
//figure,plot2d3(cell,count);
//title('Histogram','color','blue','fontsize',4);

th_Otsu=115;         //find(sigma_B==max(sigma_B));
b3=im2bw(a1,(th_Otsu/255));
figure,ShowImage(b3,'Binary Image');
title('Segmented Result Using Otsu Thresholding
    Algorithm','color','blue','fontsize',4);
```

check Appendix ?? for dependency:

`Ex10_19.tif`

Scilab code Exa 10.19 `Multipal Global Thresholding`

```
//Ex10_19
```

```
// Multipal Global Thresholding
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E.
    Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s
    ).

a=imread("Ex10_19.tif");
a1=im2double(a);
figure,ShowImage(a,'Gray Image');
title('Original Image','color','blue','fontsize',4);
[M,N]=size(a);

[count cell]=imhist(a);
figure,plot2d3(cell,count);
title('Histogram','color','blue','fontsize',4);

//////////////////////////////// Otsu Method
    //////////////////////////////

normal_hist=count/(M*N);
Sum=0;
cumu_mean=0;
for k=1:max(cell)+1
    Sum=Sum+normal_hist(k);
    P1(k)=Sum;
    cumu_mean=cumu_mean+(k*normal_hist(k));
    m(k)=cumu_mean;
    Mg=cumu_mean;
    sigma_B(k)=(((Mg*P1(k))-m(k))^2)/(%eps+(P1(k)
```

```
*(1-P1(k))));
end

th_Otsu=[80 177];          //find(sigma_B==max(sigma_B));
b2=im2bw(a1,(th_Otsu(1)/255));
b3=im2bw(a1,(th_Otsu(2)/255));
b4=b2+b3;
figure,ShowImage(mat2gray(b4),'Binary Image');
title('Image Segmented into Three region using Dual Otsu Threshold','color','blue','fontsize',4);
```

check Appendix ?? for dependency:

`Ex10_20.tif`

Scilab code Exa 10.20 `Variable Thresholding Via Image Partitioning`

```
//Ex10_20
// Variable Thresholding Via Image Partitioning
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).


function th1=otsu(count,cell)
```

```
normal_hist=count/(M/2*N/3);
Sum=0;
cumu_mean=0;
for k=1:max(cell)+1
    Sum=Sum+normal_hist(k);
    P1(k)=Sum;
    cumu_mean=cumu_mean+(k*normal_hist(k));
    m(k)=cumu_mean;
    Mg=cumu_mean;
    sigma_B(k)=(((Mg*P1(k))-m(k))^2)/(%eps+(P1(k)*(1-P1(k))));
end

th1=find(sigma_B==max(sigma_B));
endfunction


a=imread("Ex10_20.tif");
a=imresize(a,[650 813],'bicubic');
a1=im2double(a);
figure,ShowImage(a,'Gray Image');
title('Original Image','color','blue','fontsize',4);
[M,N]=size(a);

[count cell]=imhist(a);
figure,plot2d3(cell,count);
title('Histogram','color','blue','fontsize',4);

//////////////////////////////// Iterative Thresholding ////////////////////
thr = iterthresh(a1);
b1=im2bw(a1,thr);
figure,ShowImage(b1,'Gray Image');
title('Segmentation Using Iterative Global Thresholding','color','blue','fontsize',4);

///////////////////////////////// Otsu Method //////////////////////////////
```

```

normal_hist=count/(M*N);
Sum=0;
cumu_mean=0;
for k=1:max(cell)+1
    Sum=Sum+normal_hist(k);
    P1(k)=Sum;
    cumu_mean=cumu_mean+(k*normal_hist(k));
    m(k)=cumu_mean;
    Mg=cumu_mean;
    sigma_B(k)=(((Mg*P1(k))-m(k))^2)/(%eps+(P1(k)
       *(1-P1(k))));
end

th=find(sigma_B==max(sigma_B));
b2=im2bw(a1,(th/255));
figure,ShowImage(mat2gray(b2),'Binary Image');
title('Image Segmented using Otsu Threshold','color'
   ,'blue','fontsize',4);

/////////////////////////// Otsu with Image
   Partitioning /////////////////////////////
count=[];
cell=[];
z=1;
th2=[40 50 70 40 50 70];
for i=1:M/2:M
    for j=1:N/3:N
        [count cell]=imhist(a(i:(i-1)+(M/2),j:(j-1)
           +(N/3)));
        th1=otsu(count,cell);
        b3(i:(i-1)+(M/2),j:(j-1)+(N/3))=im2bw(a1(i:(
           i-1)+(M/2),j:(j-1)+(N/3)),(th2(z)/255));
        z=z+1;
    end
end

figure,ShowImage(mat2gray(b3),'Binary Image');
```

```
title('Image Segmented using Otsu Threshold','color','blue','fontsize',4);
```

check Appendix ?? for dependency:

`Ex10_22.tif`

Scilab code Exa 10.22 `Document Thresholding Using Moving Averages`

```
//Ex10_22
// Document Thresholding Using Moving Averages
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s).

a=imread("Ex10_22.tif");
a1=im2double(a);
figure,ShowImage(a1,'Gray Image');
title('Original Image','color','blue','fontsize',4);
[M,N]=size(a);

Threshold = CalculateOtsuThreshold(a1);
Thresh_Image=im2bw(a1,Threshold);
figure,ShowImage(Thresh_Image,'Binary Image');
title('Thresholded Image with Otsu Method','color','blue','fontsize',4);
```

```
mask=zeros(1,20);
array=[];
for i=1:M
        if(pmodulo(i,2)==0)
            array=[array mtlb_fliplr(a1(i,:))];
        else
            array=[array a1(i,:)];
        end
end
disp('first');
for i=1:length(array)
    for j=1:length(mask)
        if(j<length(mask)) then
            mask(j)=mask(j+1);
        else
            mask(j)=array(i);
        end
    end
    avg(1,i)=sum(mask)/length(mask);
end
disp('Second');
len=1;
for i=1:M
        if(pmodulo(i,2)==0)
            b(i,:)=avg(len:len+N-1);
            len=len+N;
        else
            b(i,:)=avg(len:len+N-1);
            len=len+N;
        end
end
disp('Last');

b=0.5*b;
for i=1:M
    for j=1:N
```

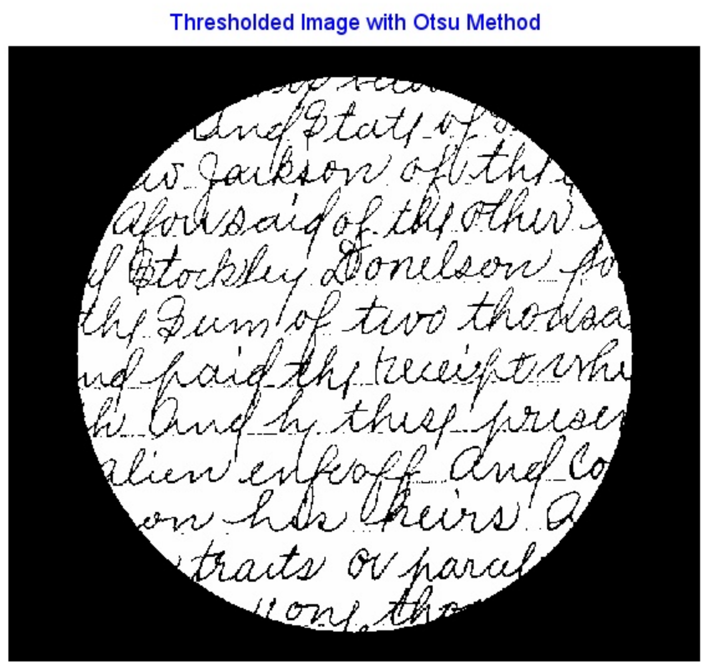


Figure 10.5: Document Thresholding Using Moving Averages

```
        if(b(i,j)>a1(i,j)) then
            c(i,j)=0;
        else
            c(i,j)=1;
        end
    end
end
figure,ShowImage(c,'Binary Image');
title('Local Thresholding Using Moving Average','
    color','blue','fontsize',4);
```

Scilab code Exa 10.23 Segmentation by Region Growing

```
//Ex10_23
// Segmentation by Region Growing
// Version : Scilab 5.4.1
// Operating System : Window-xp, Window-7
//Toolbox: Image Processing Design 8.3.1-1
//Toolbox: SIVP 0.5.3.1-2
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E.
    Woods

clc;
close;
clear;
xdel(winsid())//to close all currently open figure(s
    ).
a=imread("Ex10_23.tif");
a1=im2double(a);
figure,ShowImage(a1,'Gray Image');
title('Original Image','color','blue','fontsize',4);
[M,N]=size(a);

[count cell]=imhist(a);
figure,plot2d3(cell,count);
title('Histogram','color','blue','fontsize',4);

th=254/255;
Thresh_Image=im2bw(a1,th);
figure,ShowImage(Thresh_Image,'Gray Image');
title('Thresholded Image','color','blue','fontsize'
    ,4);
for i=1:M
    for j=1:N
```

```
        if(Thresh_Image(i,j)) then
            Thresh_Image1(i,j)=1;
        else
            Thresh_Image1(i,j)=0;
        end
    end
end

BlobImage = SearchBlobs(Thresh_Image);  // Connected Compoment Labelling
IsCalculated = CreateFeatureStruct(%f); // Feature struct is generated.
IsCalculated.Centroid = %t; // The bounding box shall be calculated for each blob.
BlobStatistics = AnalyzeBlobs(BlobImage, IsCalculated);
Seed_Image=zeros(M,N);
for i=1:max(BlobImage) // Centroid Calculation
    Seed_Image(BlobStatistics(i).Centroid(1,2), BlobStatistics(i).Centroid(1,1))=1;
end
figure,ShowImage(Seed_Image,'Gray Image');
title('Seed Point Image','color','blue','fontsize',4);

Diff=uint8(255*imsubtract(a1,Thresh_Image1));
figure,ShowImage(Diff,'Gray Image');
title('Seed Point Image','color','blue','fontsize',4);
[count cell]=imhist(Diff);
figure,plot2d3(cell,count);
title('Histogram','color','blue','fontsize',4);

Thresh_Image2=uint8(zeros(M,N))
for i=1:M
    for j=1:N
        if(Diff(i,j)<=68) then
            Thresh_Image2(i,j)=255;
```

```
        else if(Diff(i,j)>68 & Diff(i,j)<=165) then
            Thresh_Image2(i,j)=125;
        else
            Thresh_Image2(i,j)=0;
        end
    end
    end
end
figure,ShowImage(uint8(Thresh_Image2),'Gray Image');
title('Seed Point Image','color','blue','fontsize'
   ,4);
```

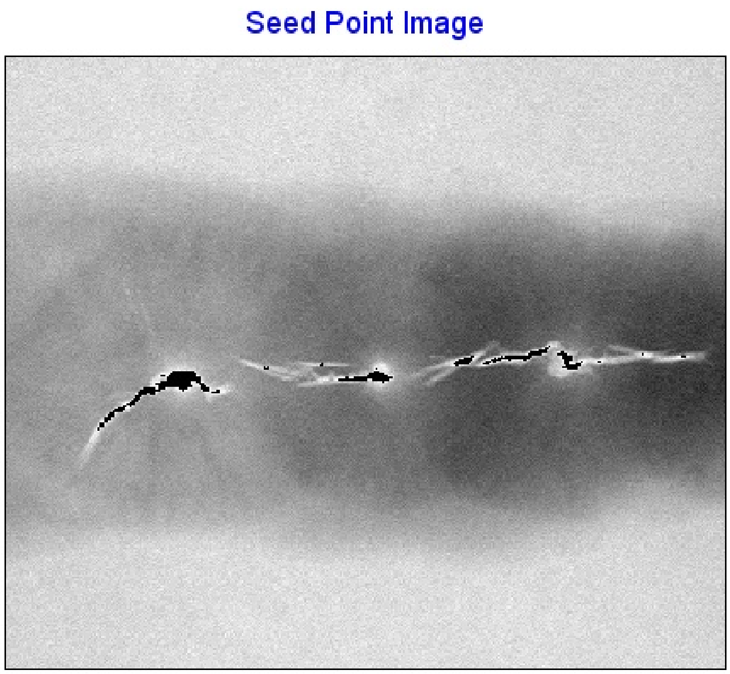


Figure 10.6: Segmentation by Region Growing

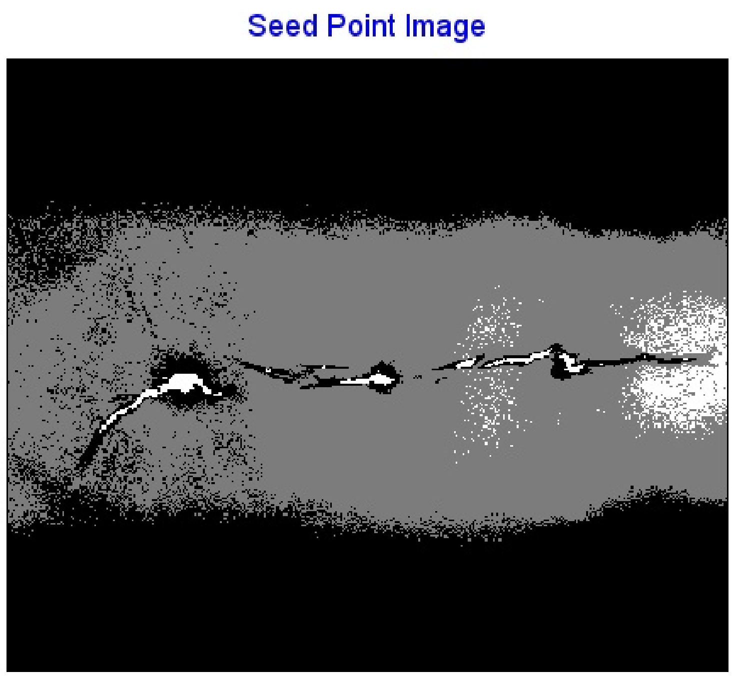


Figure 10.7: Segmentation by Region Growing

Appendix

Scilab code AP 1 Another illustration of the three principal Edge Detection Methods

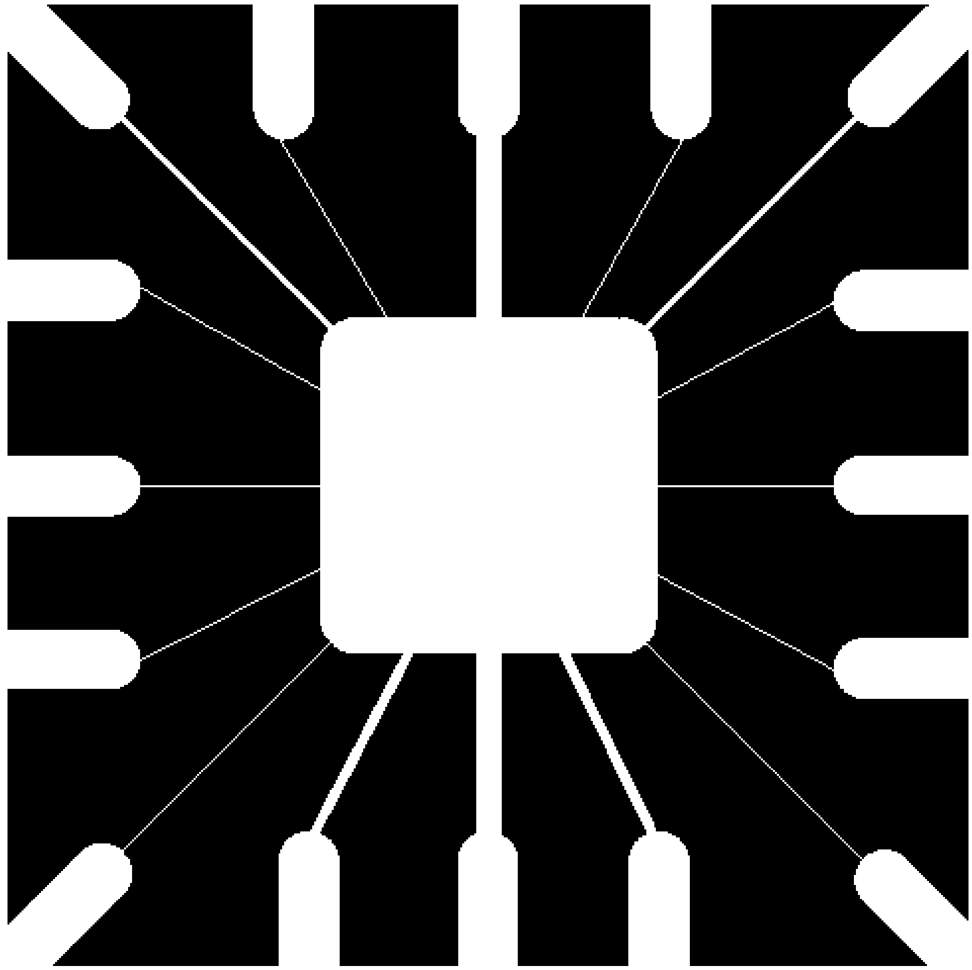
Scilab code AP 2
Illustration of the Canny Edge Detection Methods

Scilab code AP 3

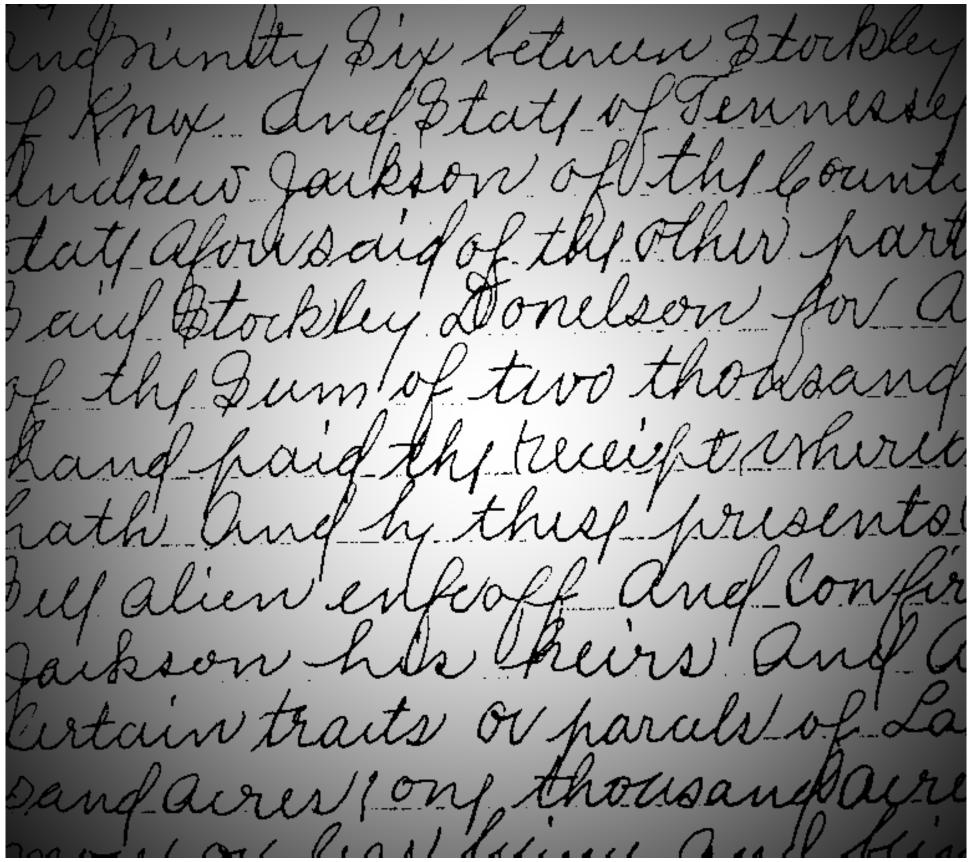
Illustration of the Marr Hildreth Edge Detection Methods

Scilab code AP 4
Illustration of the 2 D Gradient Magnitude and Angle

Scilab code AP 5
Behavior of the First and Second Derivative of a Noisy Edge

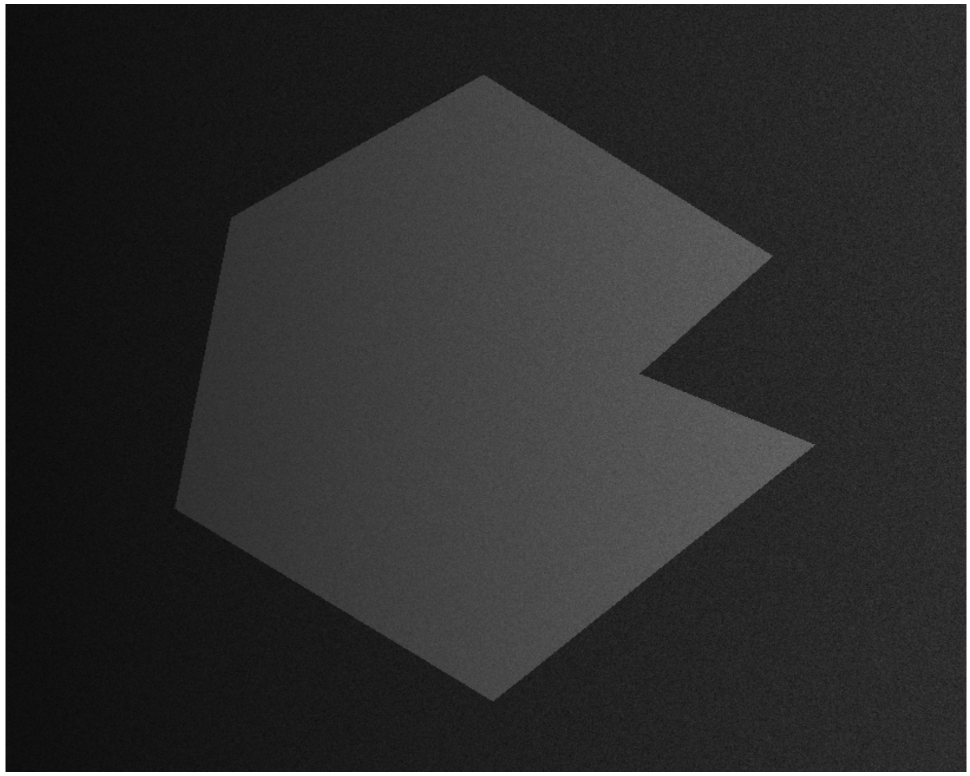


Scilab code AP 6
Detection of Lines in Specified Direction

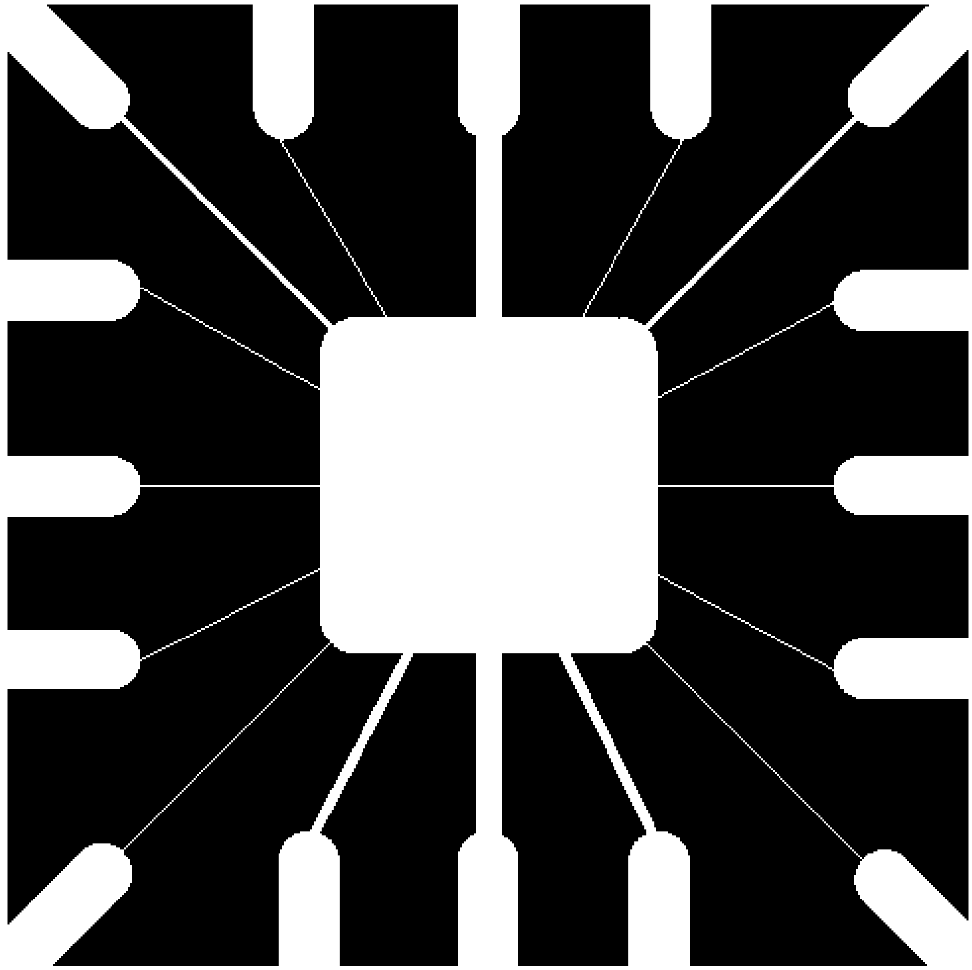


nd Ninety Six between Stockley
f Knox and State of Tennesse
Andrew Jackson of the Count
tate aforesaid of the other part
said Stockley Donelson for A
of the Sum of two thousand
hand paid the receipt where
hath and by these presents
sell alien enfeoff and Confir
Jackson his heirs and A
certain tracts or parcels of La
sand acres (one thousand acre

Scilab code AP 7
Document Thresholding Using Moving Averages

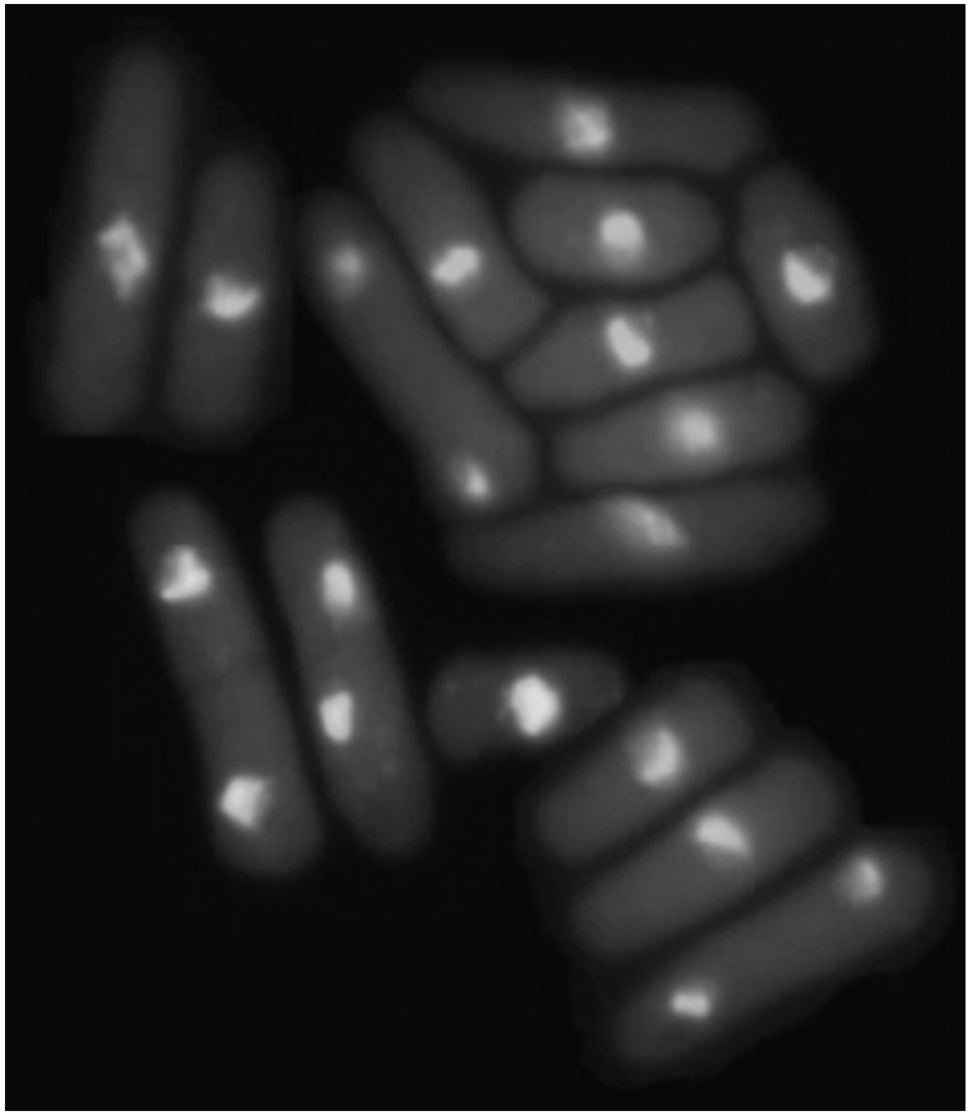


Scilab code AP 8
Variable Thresholding Via Image Partitioning

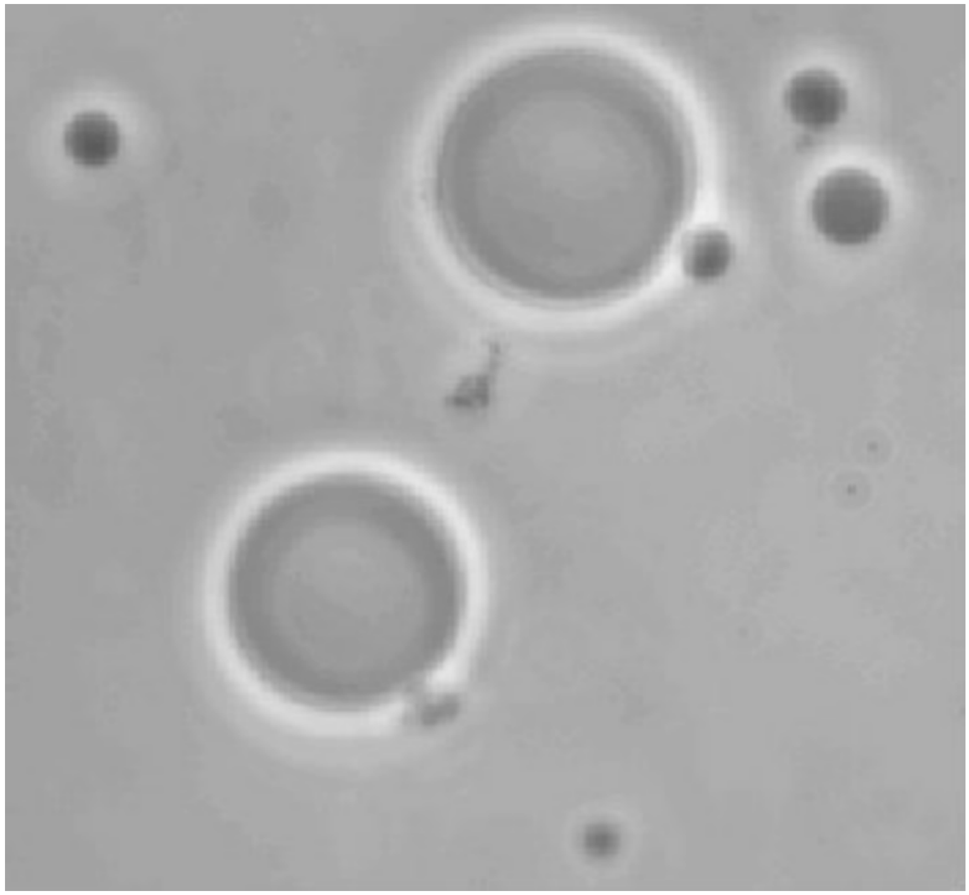


Scilab code AP 9
Using the Laplacian for the Detection

Scilab code AP 10 Multipal Global Thresholding

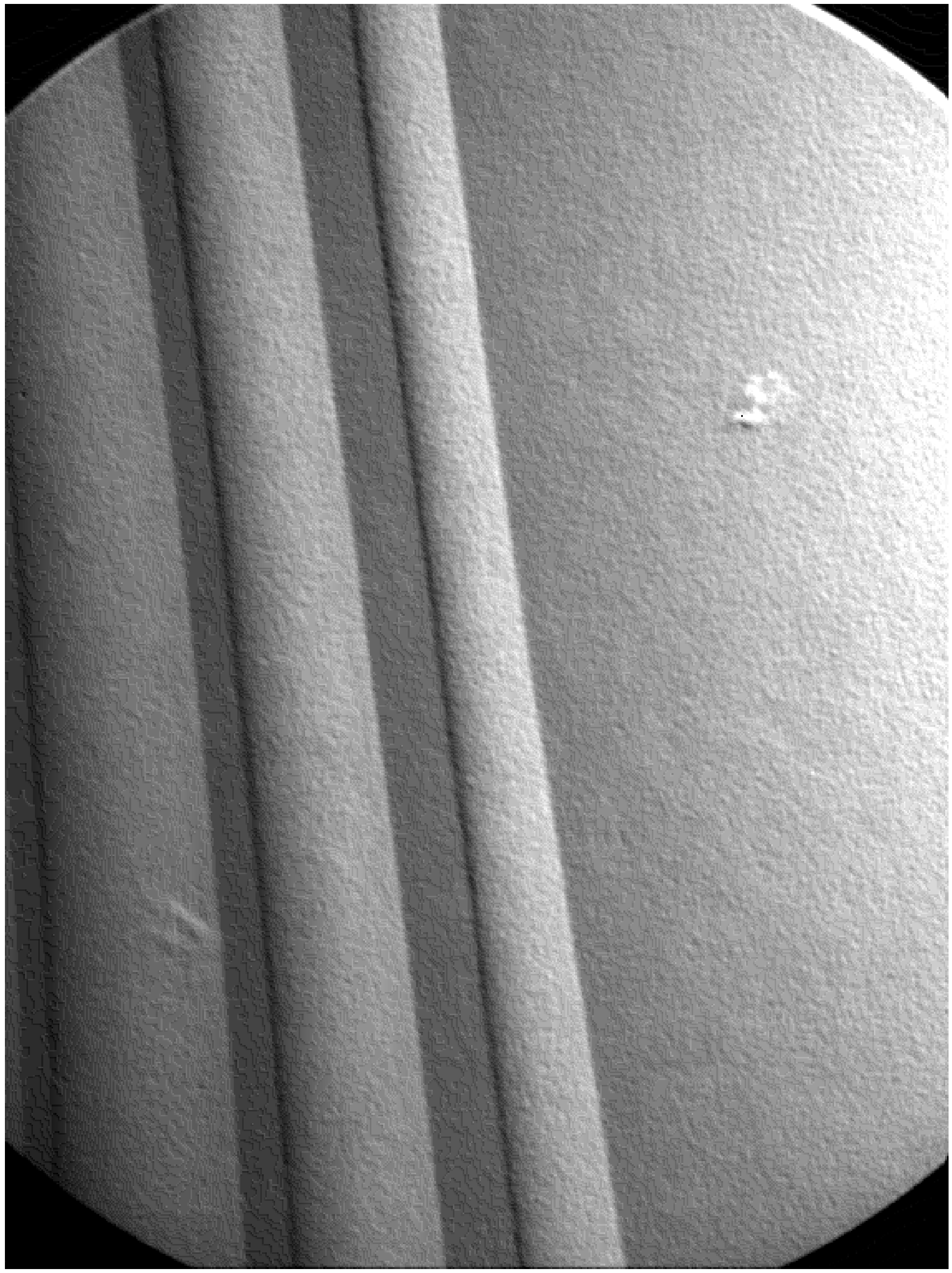


Scilab code AP 11
Using Edge Information Based on the Laplacian to Improve Global Thresholding



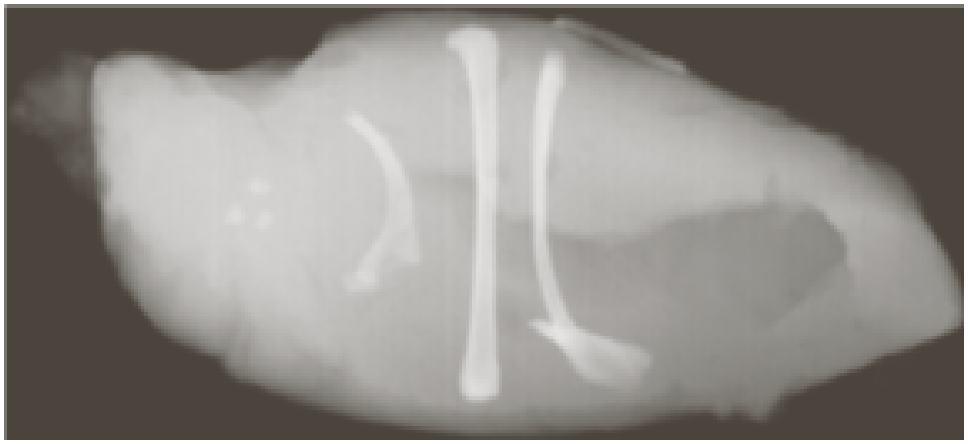
Scilab code AP 12
Optimum Global Thresholding using Otsu's Method

Scilab code AP 13
Global Thresholding

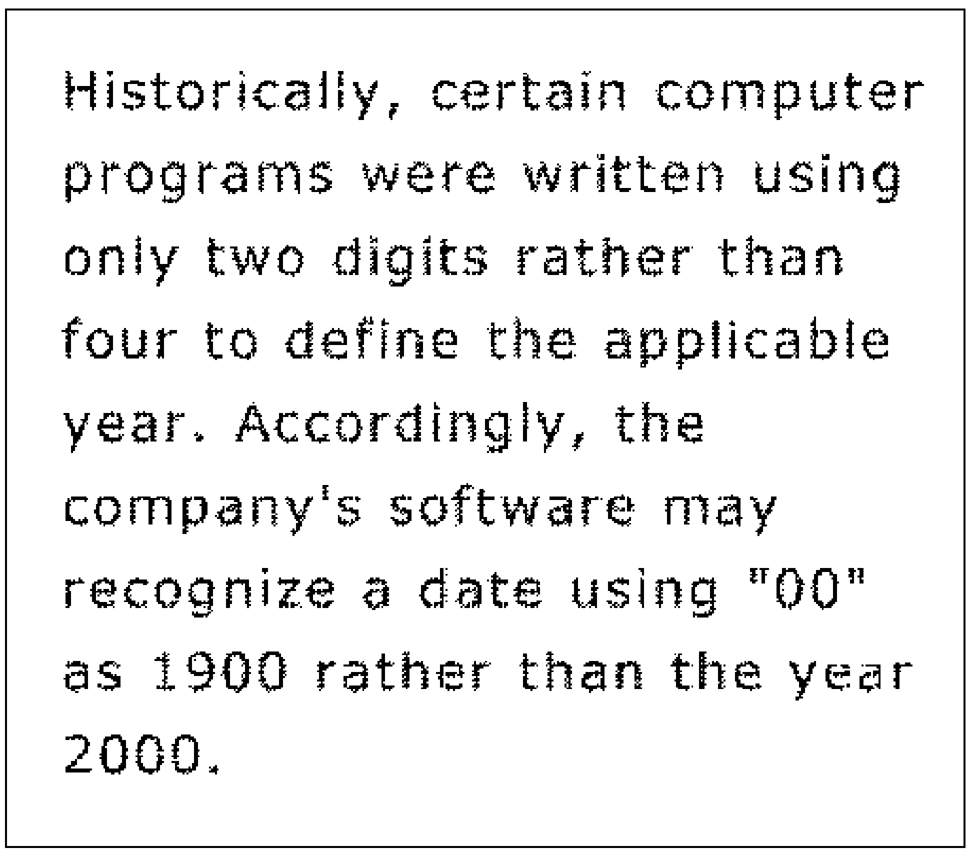


Scilab code AP 14
Detection of Isolated Point in an Image

 Illustration of Gray Scale Erosion and Dilation



Using Connected Components to Detect Foreign Object in Packaged Food

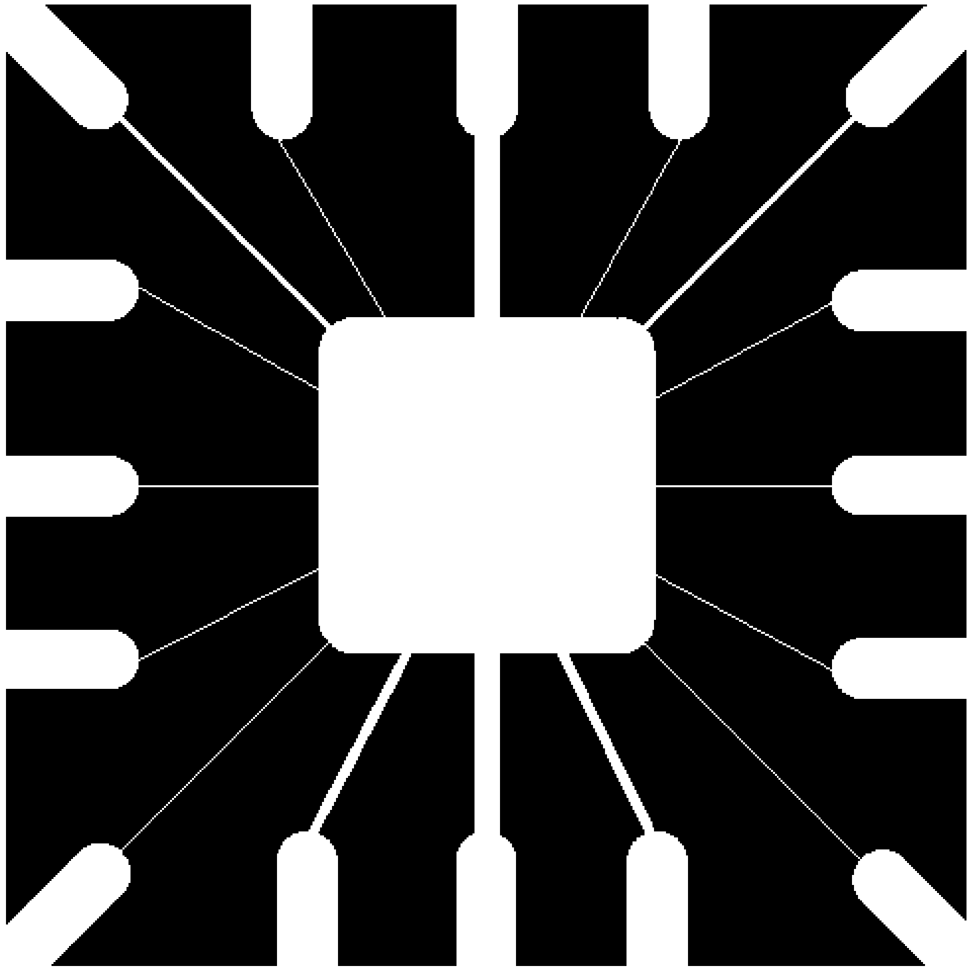


Scilab code AP 19
An Illustration of Dilation

Boundary Extraction by Morphological Processing

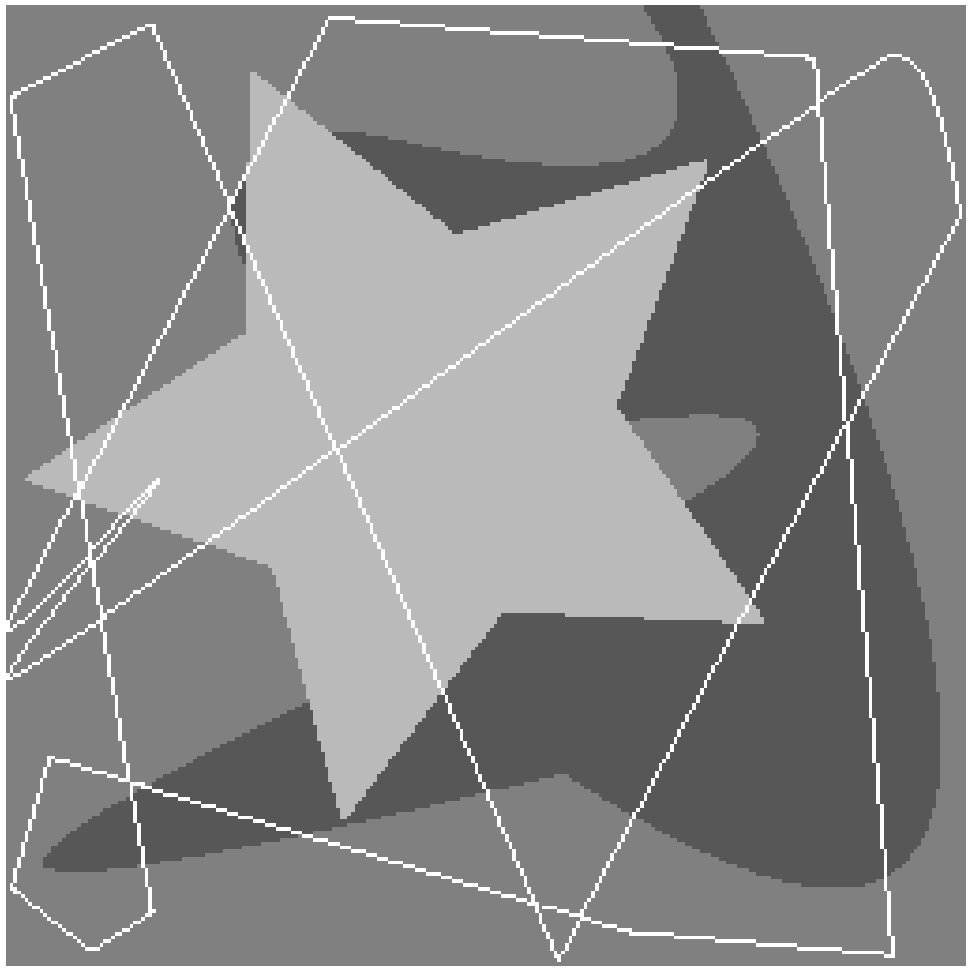

Use of opening and closing for Morphological Filtering

 Illustration of Gray Scale Opening and Closing



Scilab code AP 21

Using Erosion to remove image component



Scilab code AP 22 Image Entropy Estimation

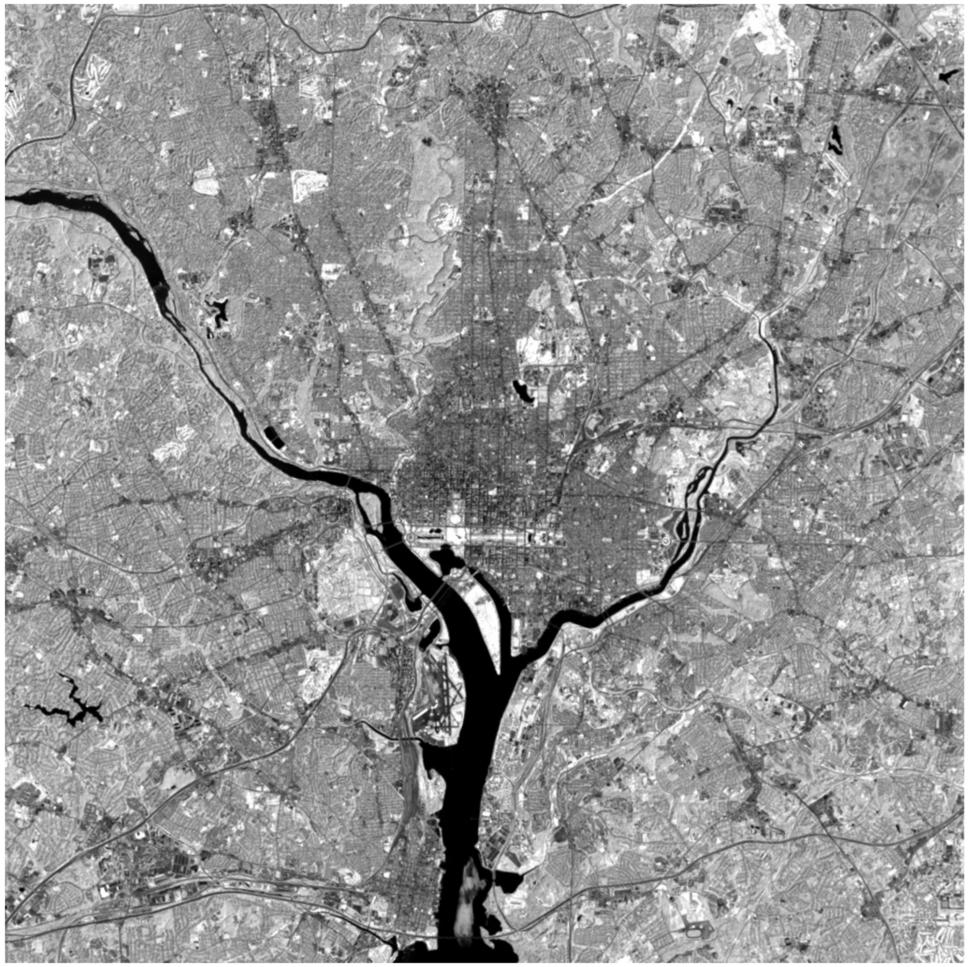
Scilab code AP 23
Tonal Transformations

Scilab code AP 24
Tonal Transformations

Scilab code AP 25

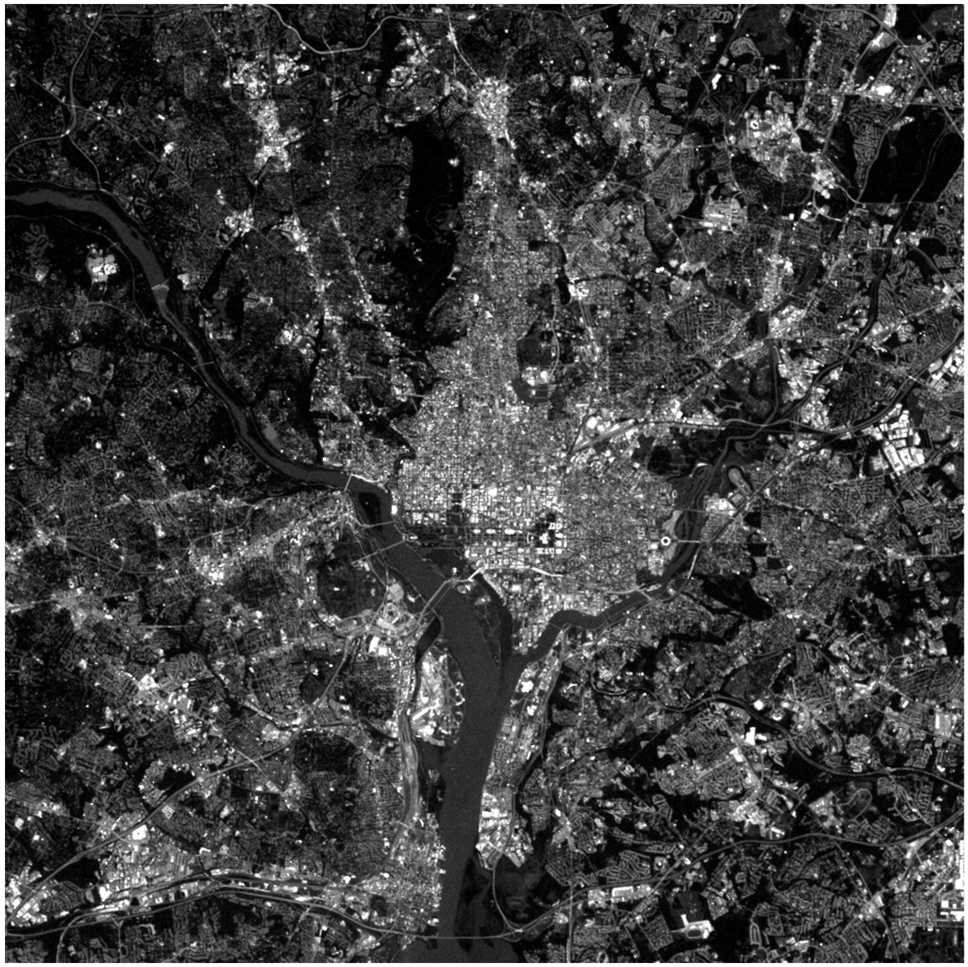
Tonal Transformations

Scilab code AP 26
Computing Color Image Components



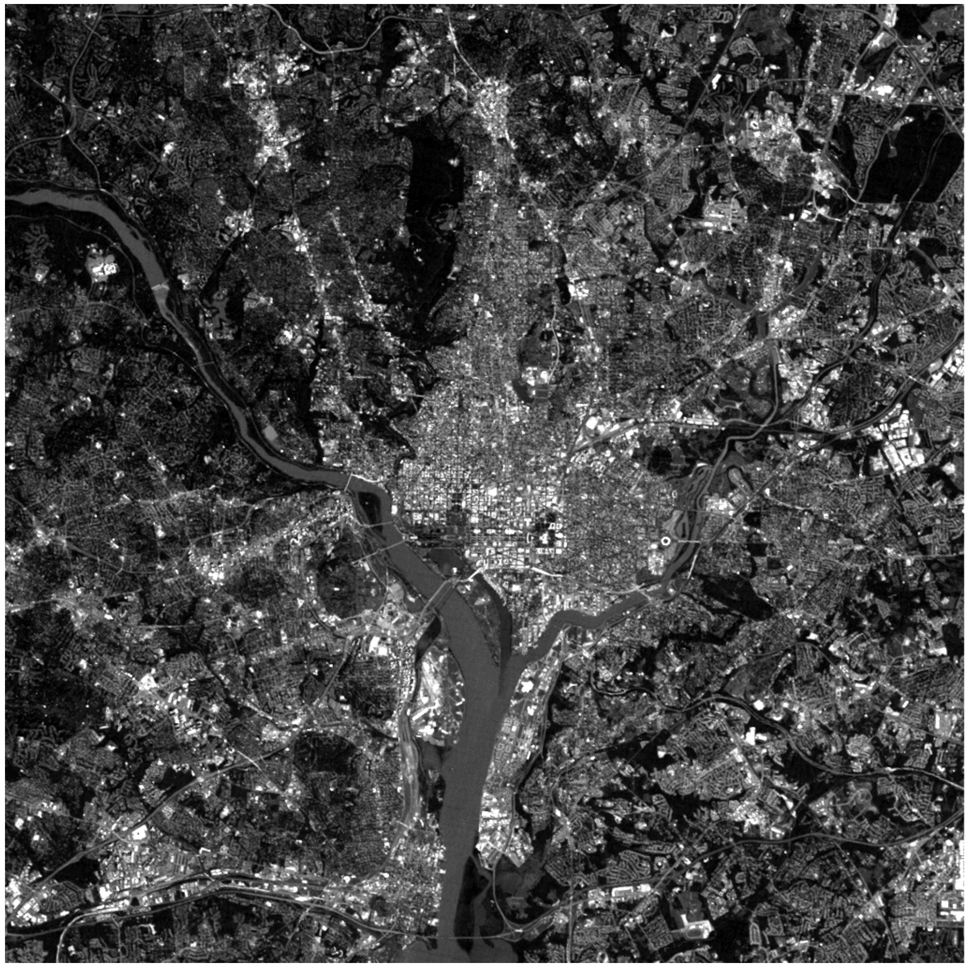
Scilab code AP 27

Color Coding of Multi Spectral Images

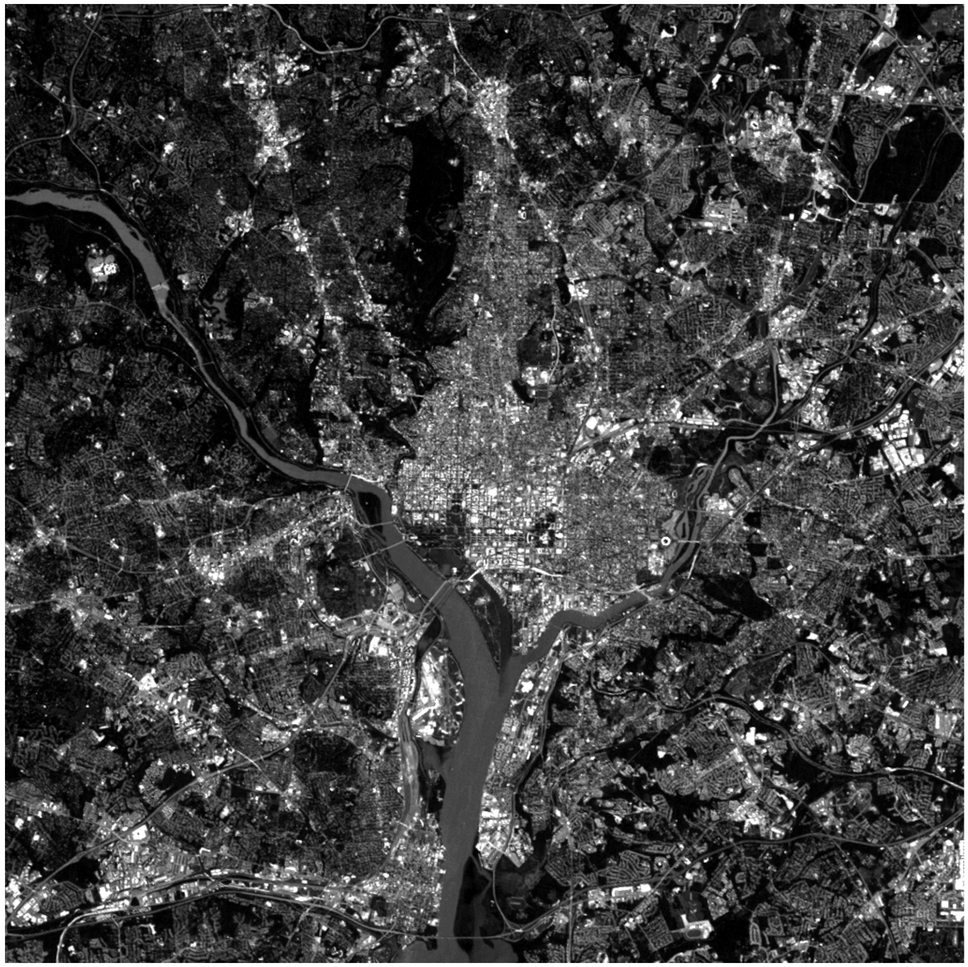


Scilab code AP 28

Color Coding of Multi Spectral Images

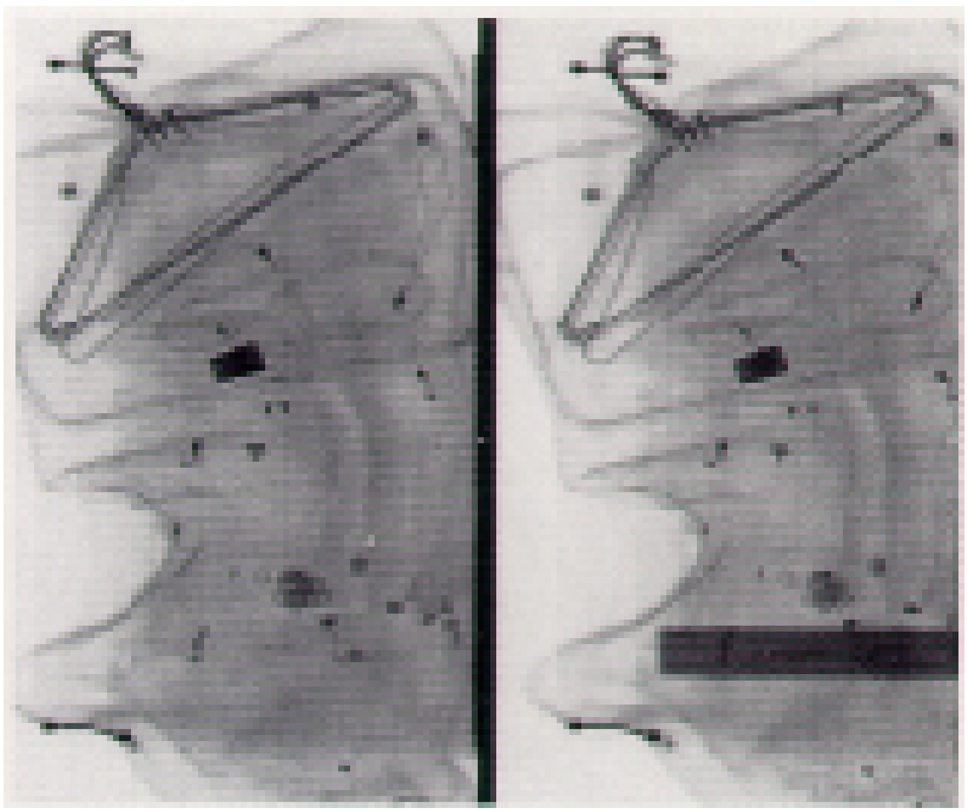


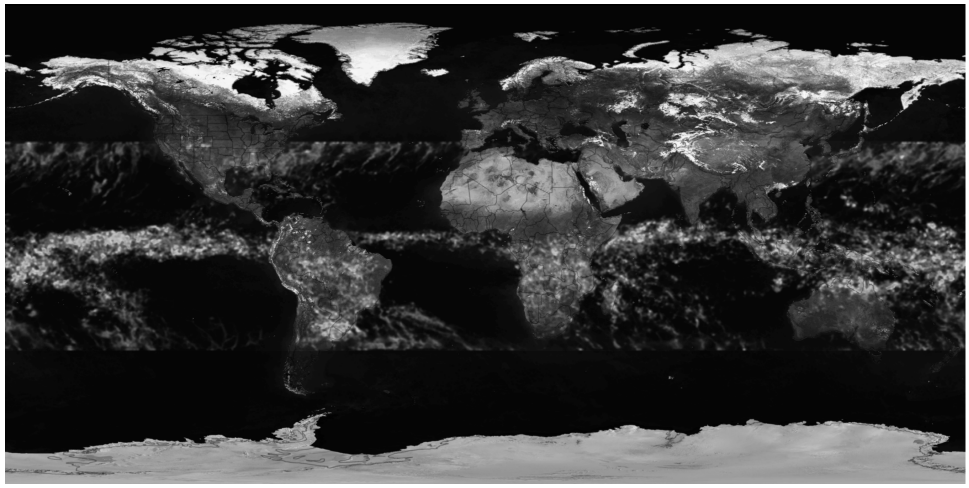
Scilab code AP 29 Color Coding of Multi Spectral Images



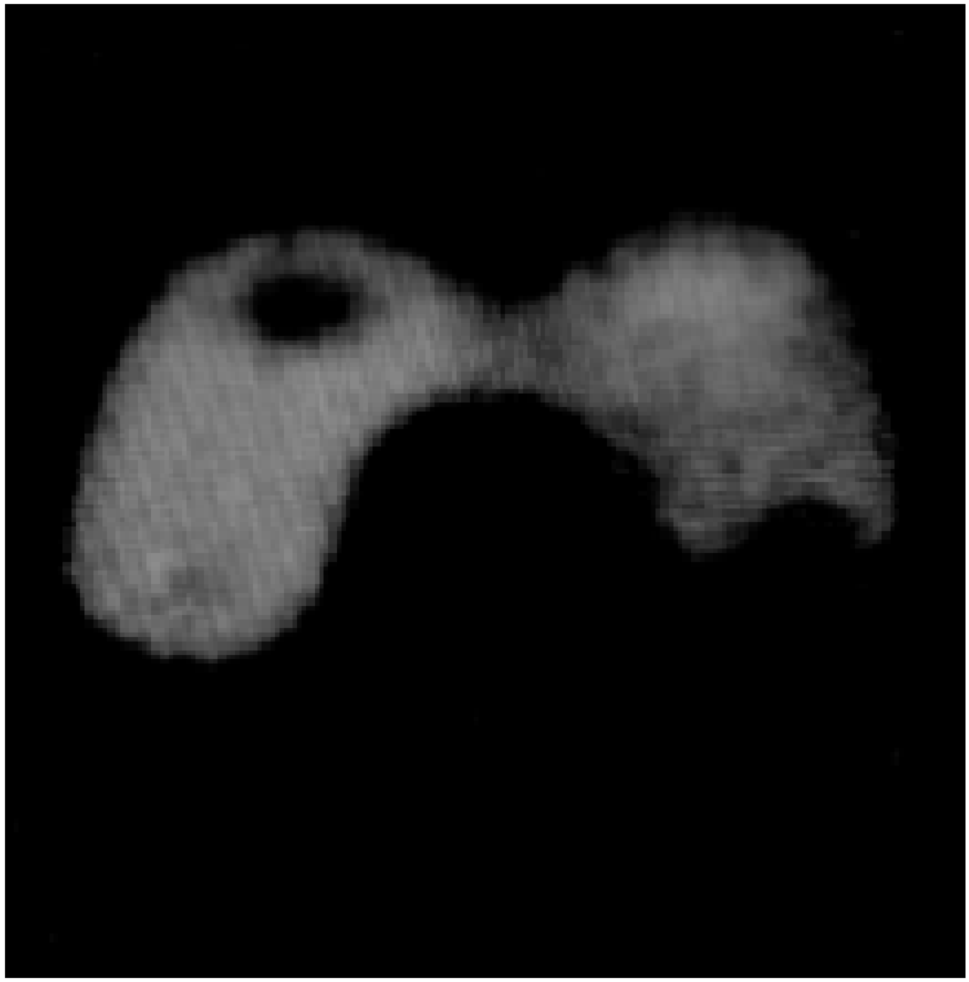
Scilab code AP 30

Color Coding of Multi Spectral Images


Use of Psedocolor for highlighting Exposives Contained in Luggage



Scilab code AP 32
Use of Color to Highlight Rainfall Levels



Scilab code AP 33
Intensity Slicing

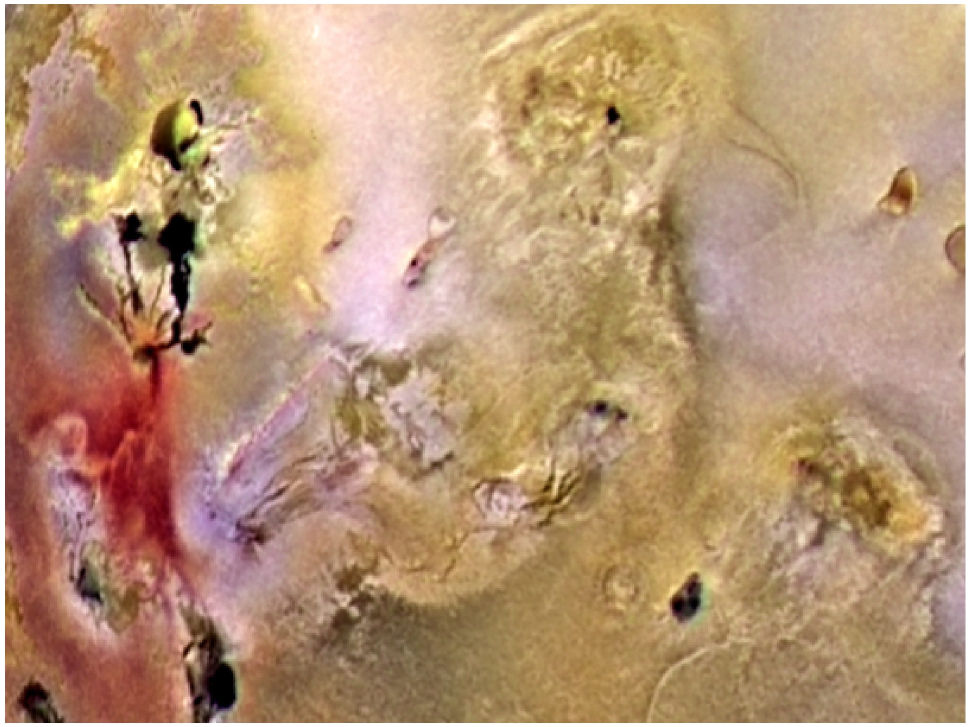
Scilab code AP 34 Illustration of the effects of converting noisy RGB Images to HSI

Scilab code AP 35
Illustration of the effects of converting noisy RGB Images to HSI

Scilab code AP 36

Illustration of the effects of converting noisy RGB Images to HSI

Scilab code AP 37
Edge Detection Vector Space

Scilab code AP 38 

egmentation in HSI Space

Scilab code AP 39 Sharpning with the Laplacian

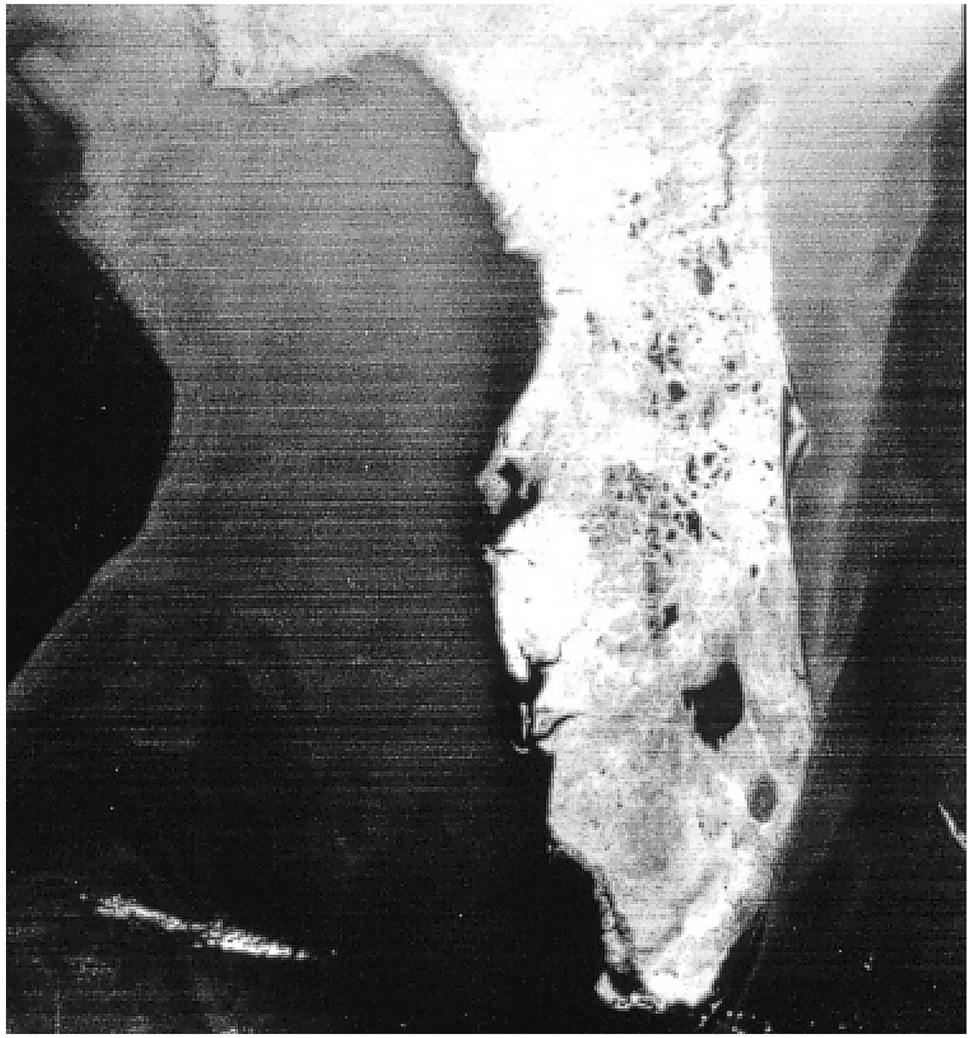
Scilab code AP 40

Color Image Smoothning by Neighbourhood Averaging

Scilab code AP 41

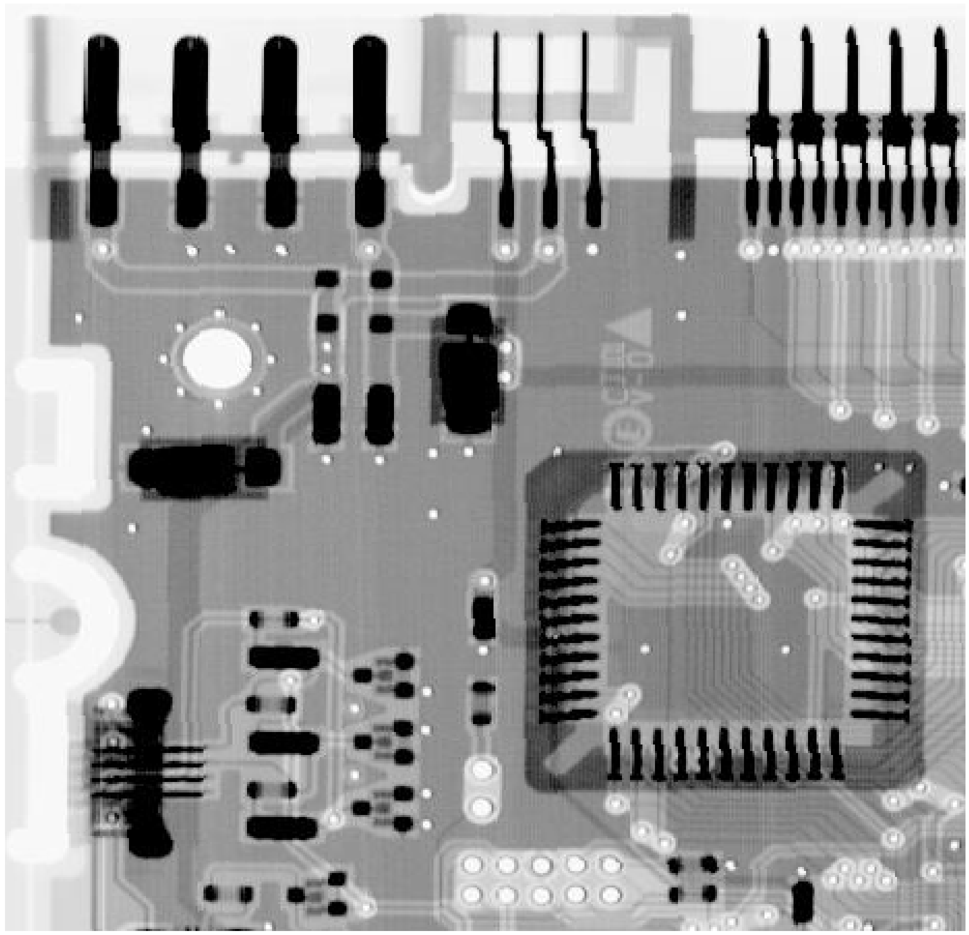
Histogram Equalization in the HSI Color Space

Scilab code AP 42
Color Balancing

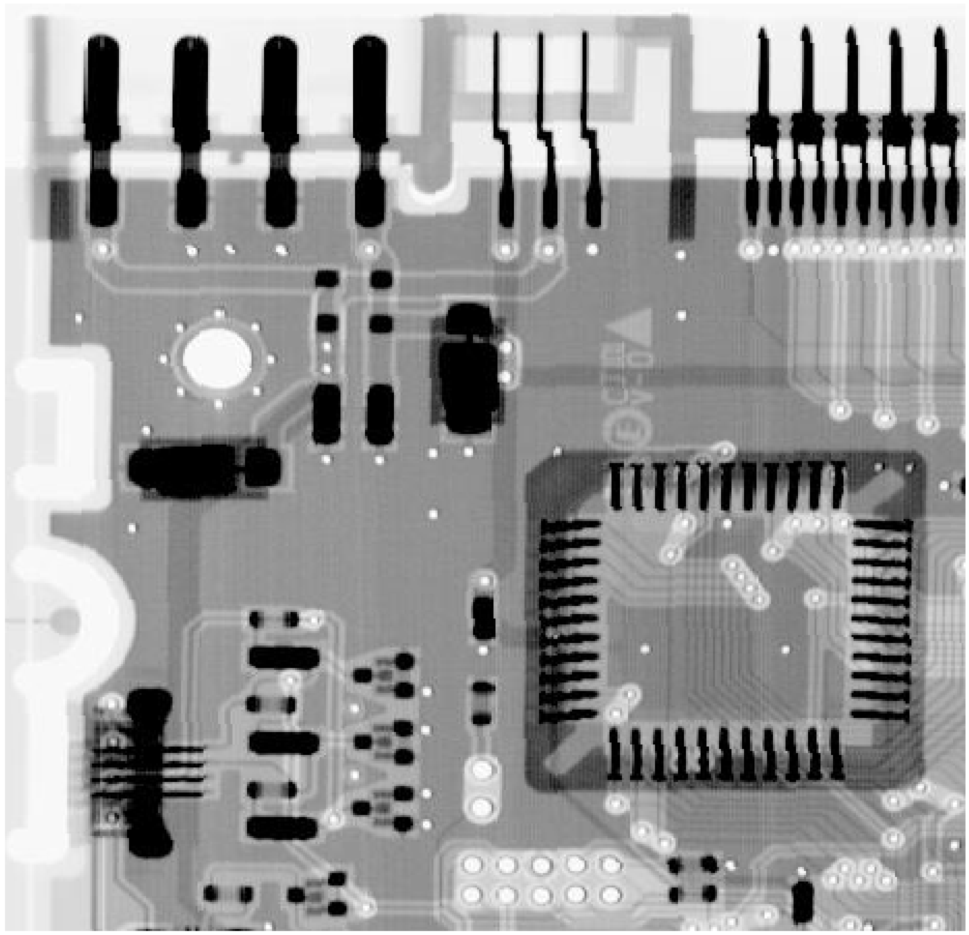


Scilab code AP 43

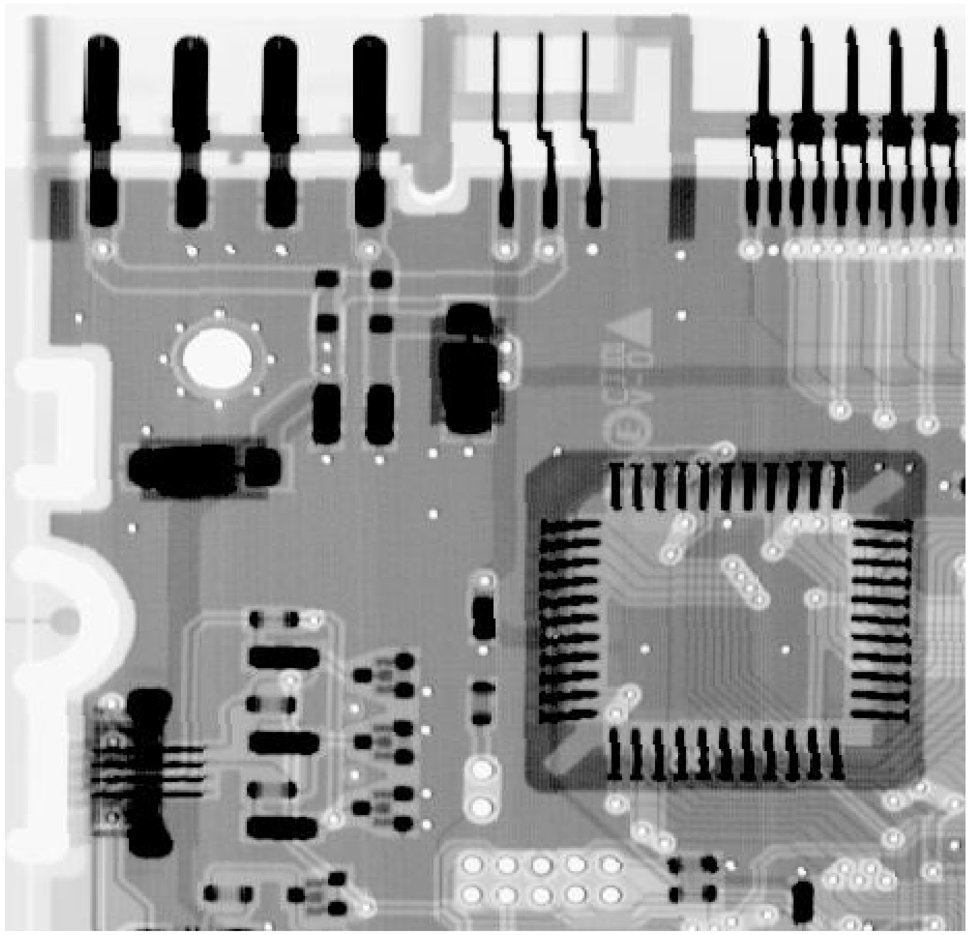
Removal of Periodic Noise by Notch Filtering



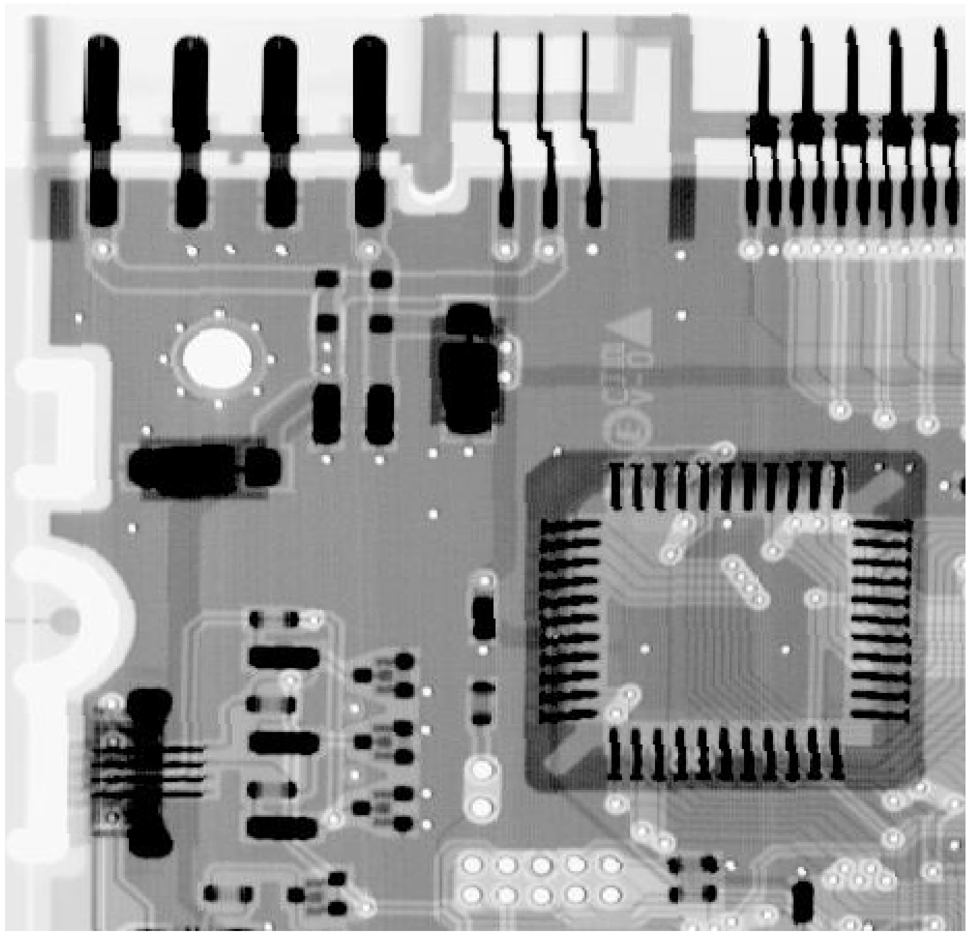
Scilab code AP 44
Illustration of Adaptive Median Filter



Scilab code AP 45
Illustration of Adaptive Local Noise Reduction Filtering



Scilab code AP 46
Illustration of Order Statistic filter



Scilab code AP 47 Illustration of Mean Filters

Comparision of Inverse Filtering and Wiener Filtering

Inverse Filtering

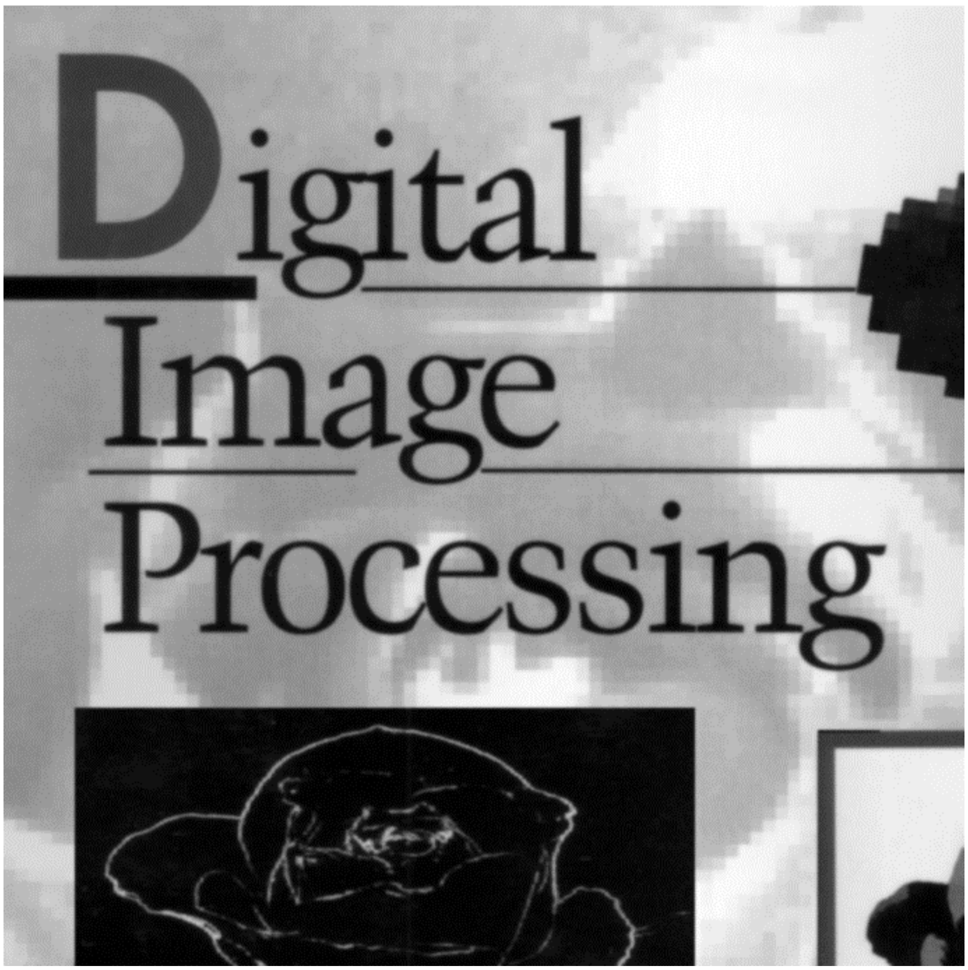
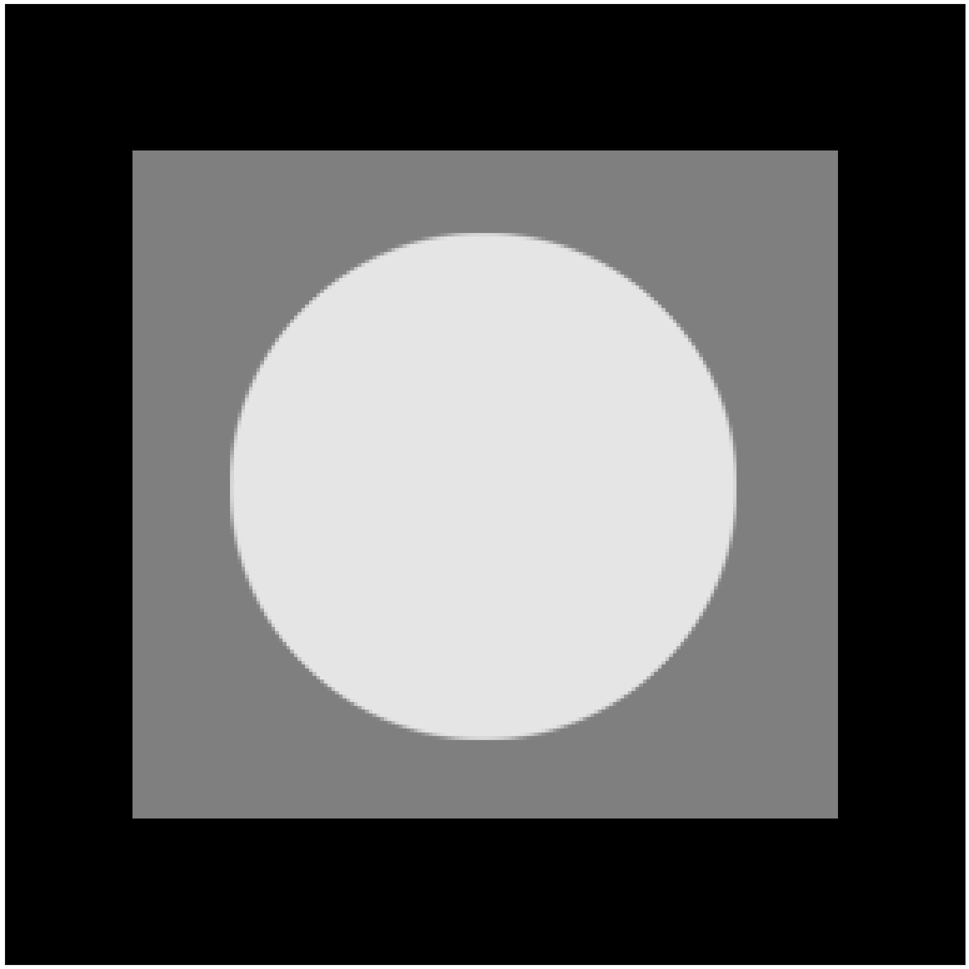
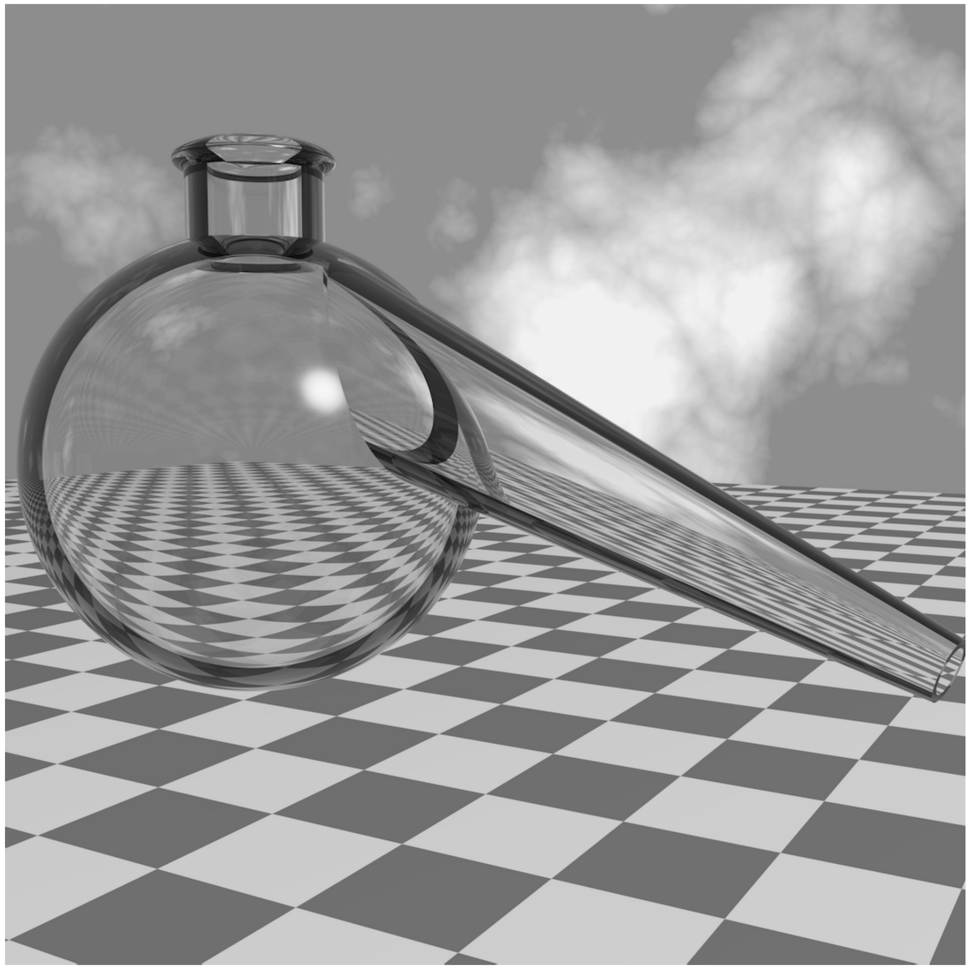


Image Bluring Due to Motion

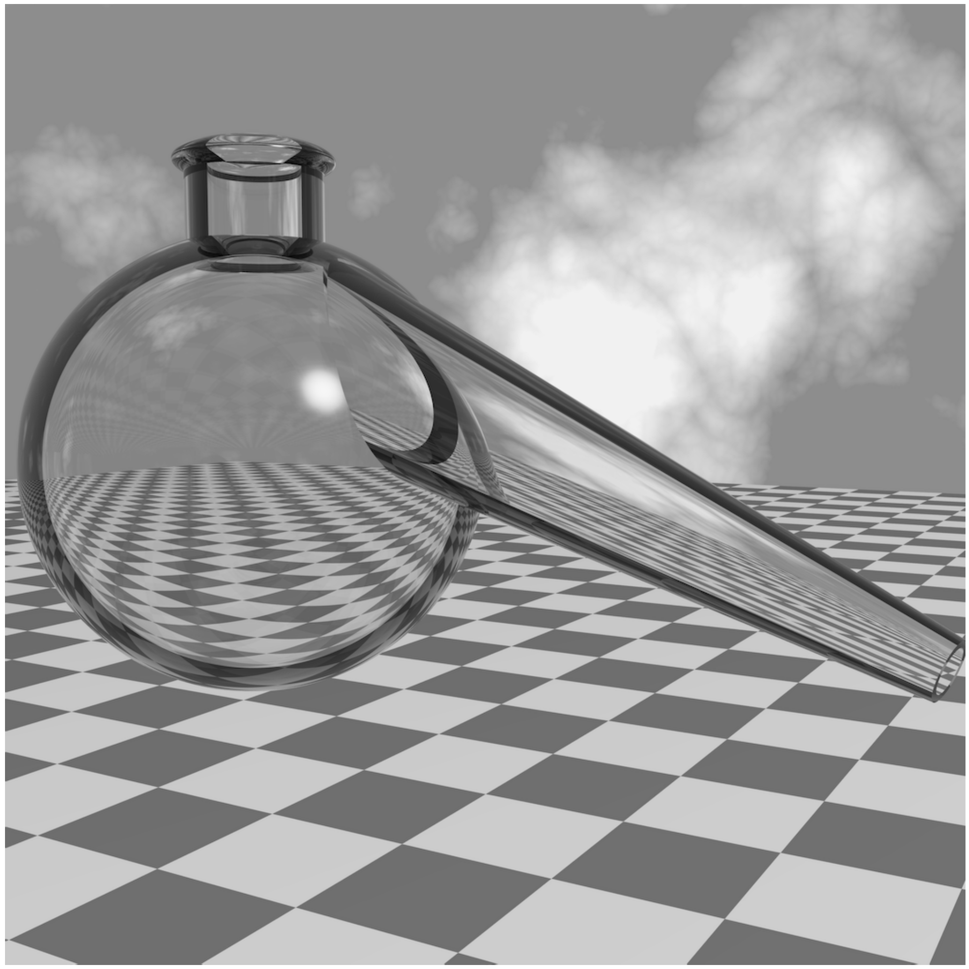


Scilab code AP 51

Noisy Images and their Histogram



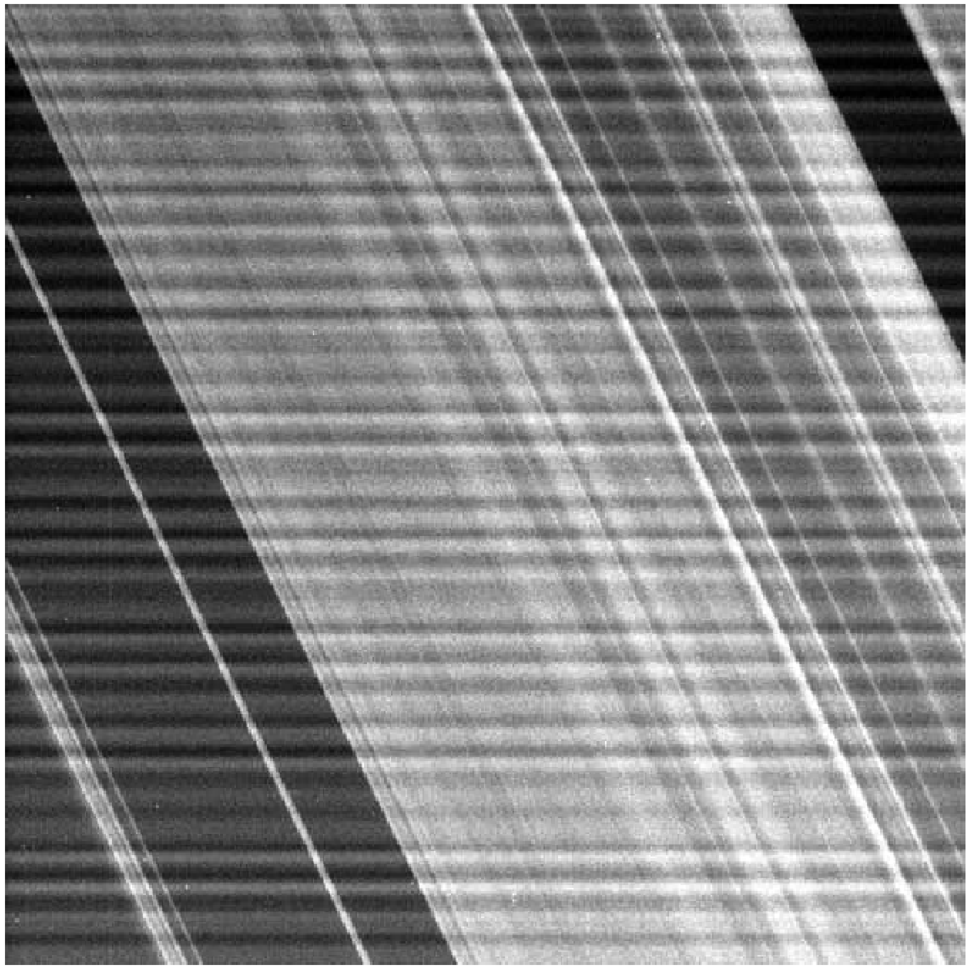
Scilab code AP 52 Illustration of Jaggies in Image Zooming



Scilab code AP 53
Illustration of Jaggies in Image Shrinking

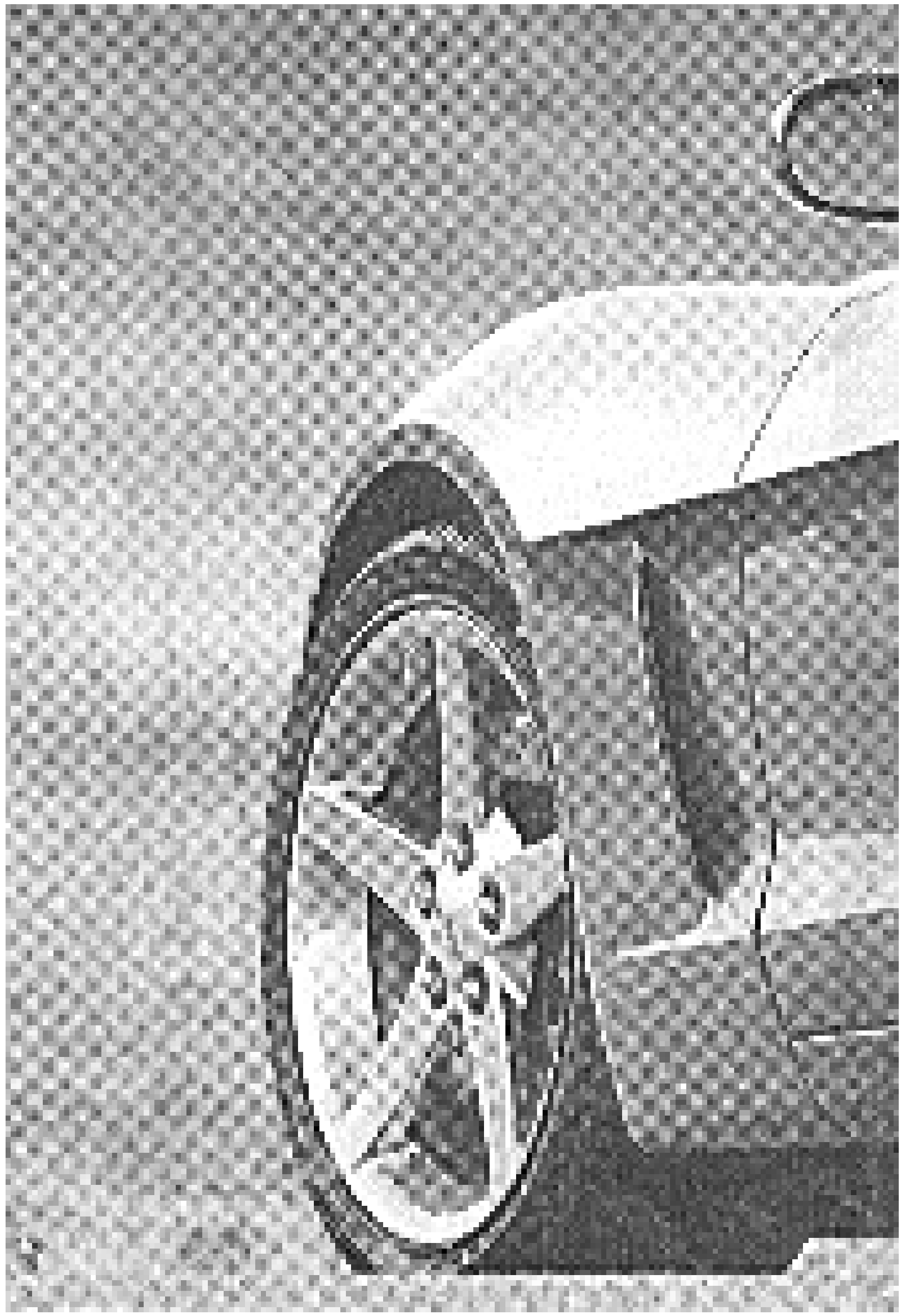
Scilab code AP 54

Illustration of Aliasing in Resampled Images

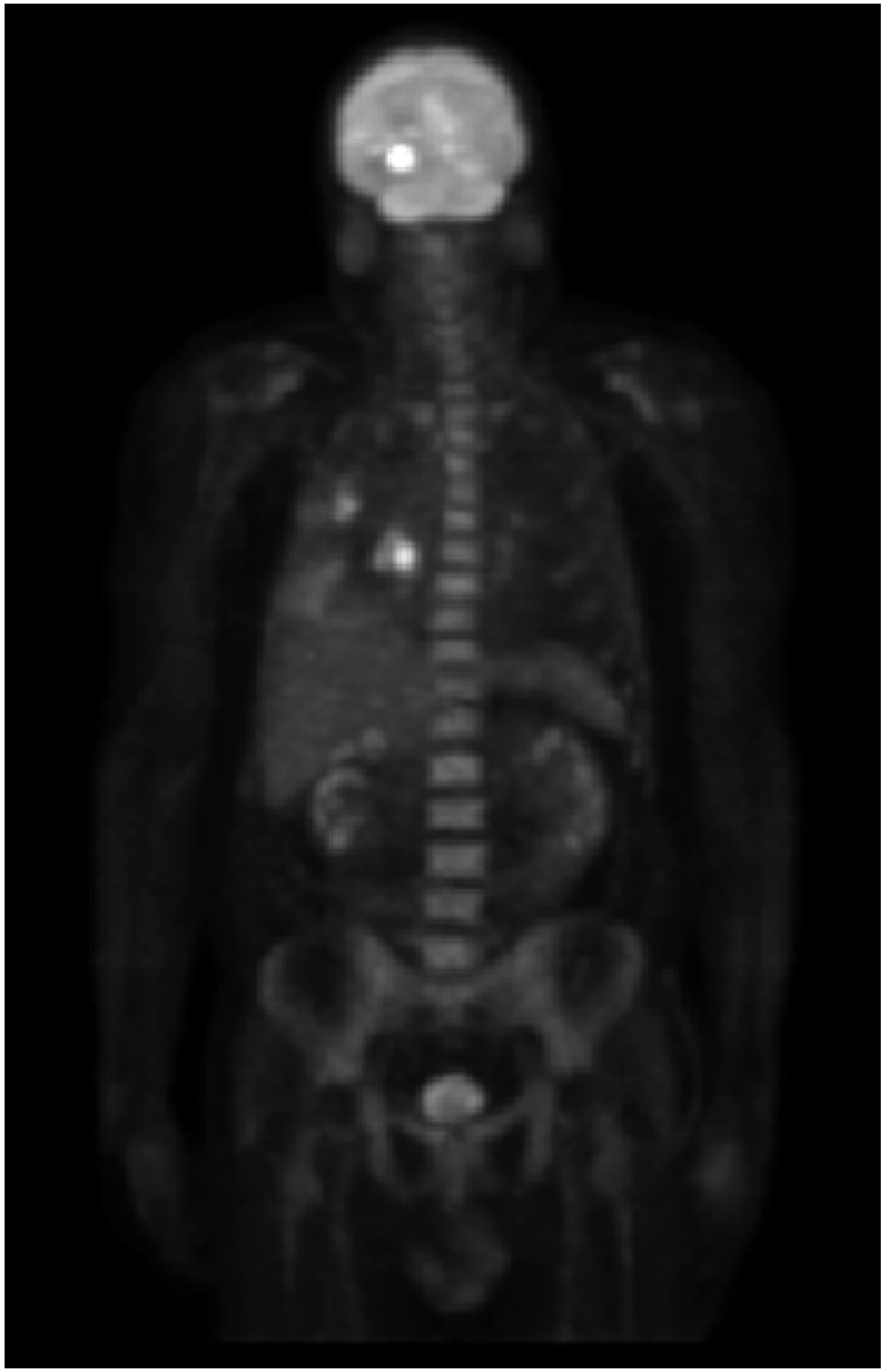


Scilab code AP 55

Enhancement of Corrupted Cassini Saturn Image by Notch Filtering

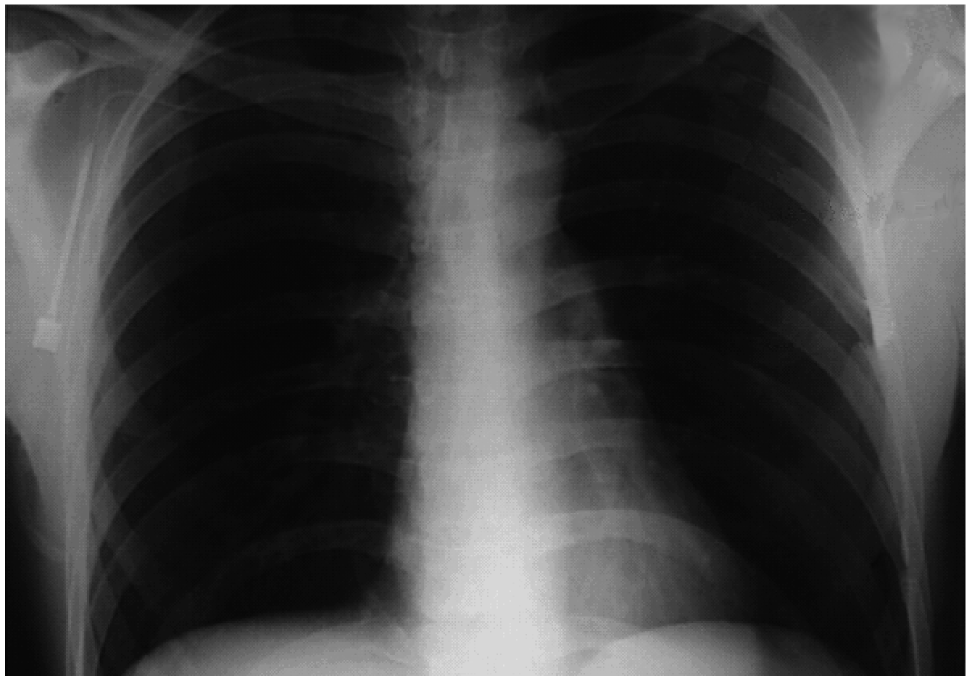


Scilab code AP 56
Reduction of Moire Patterns Using Notch Filtering



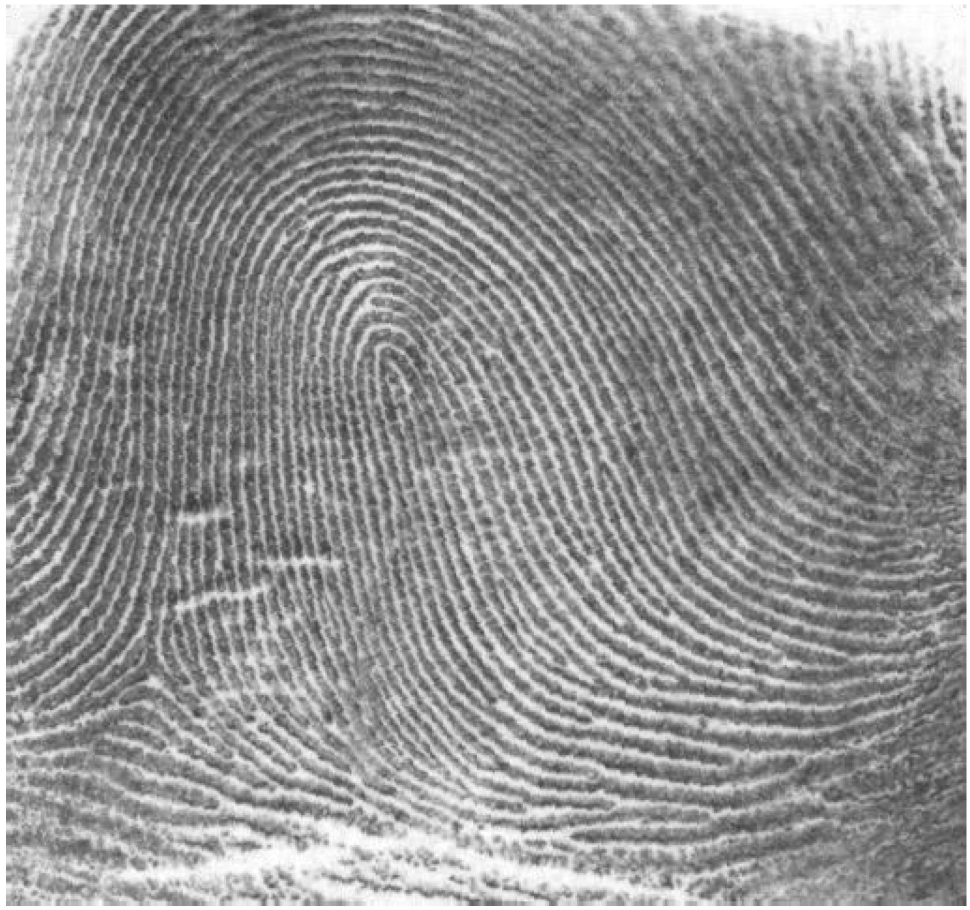
Scilab code AP 57

Image Enhancement using Homomorphic Filtering

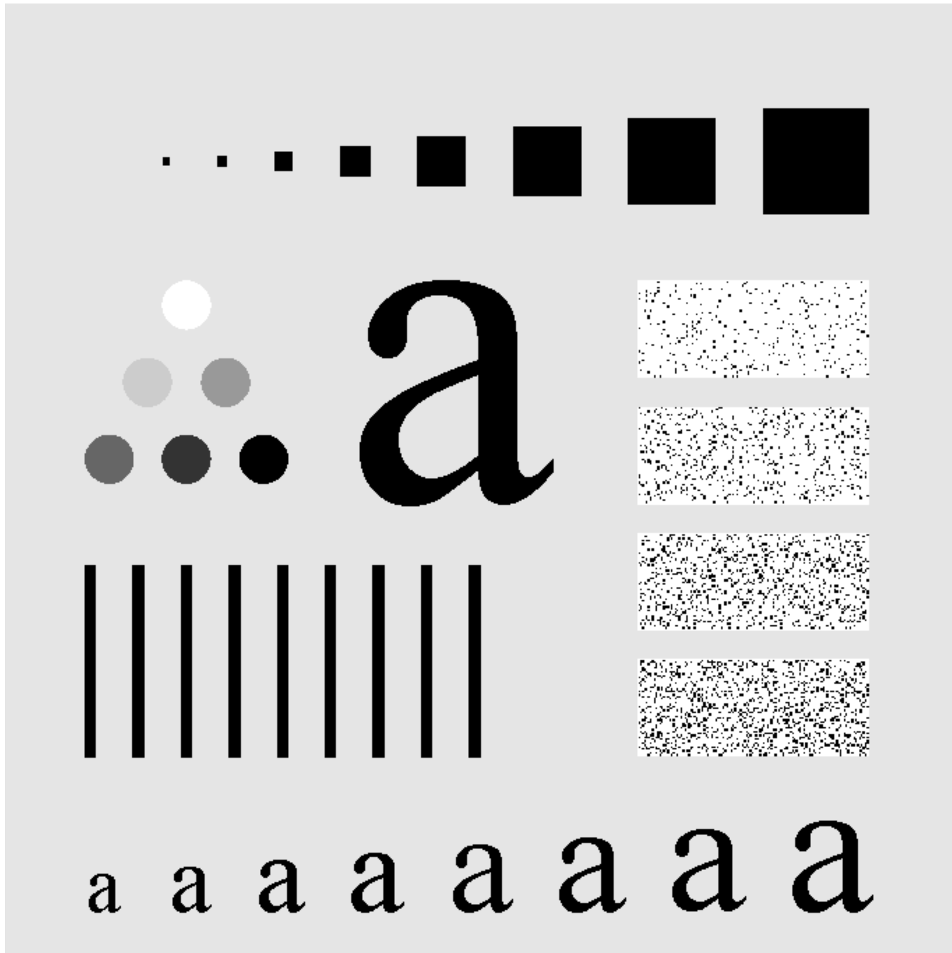


Scilab code AP 58
Image Enhancement using High Frequency Emphasis Filtering

Scilab code AP 59
Image Sharpning in the Frequency Domain using the Laplacian

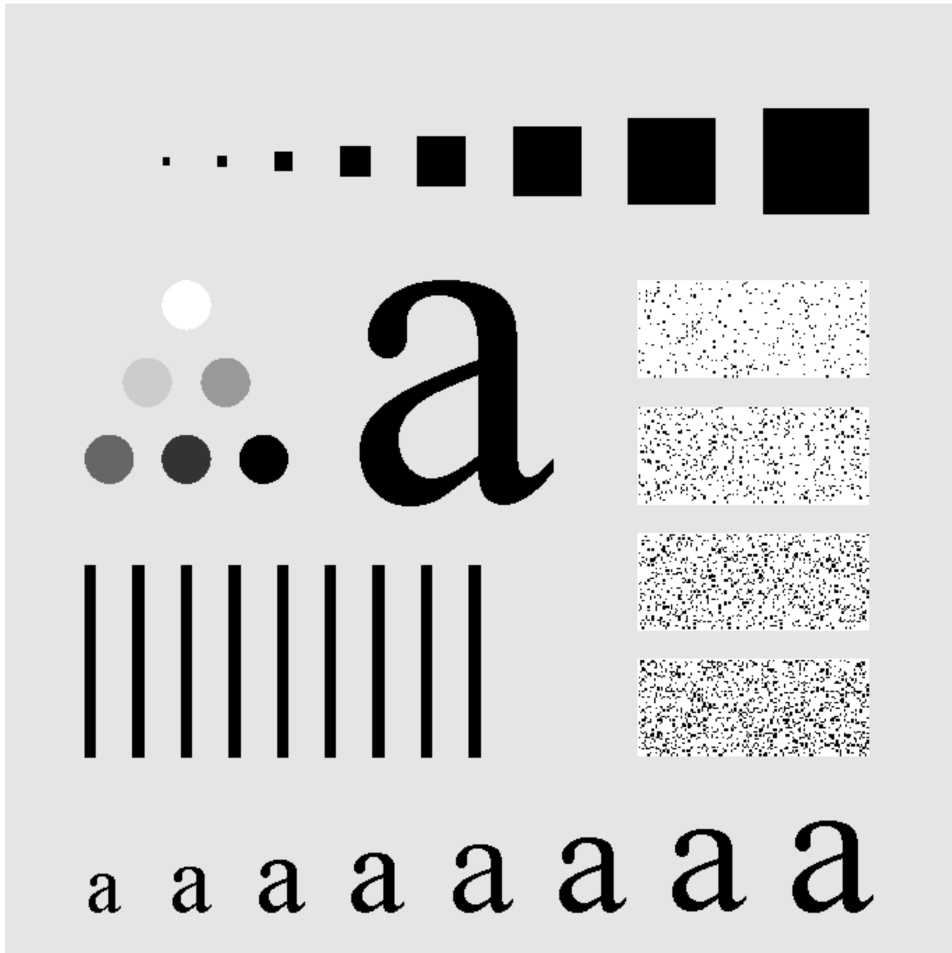


Scilab code AP 60
Using Highpass Filter and Thresholding for Image enhancement



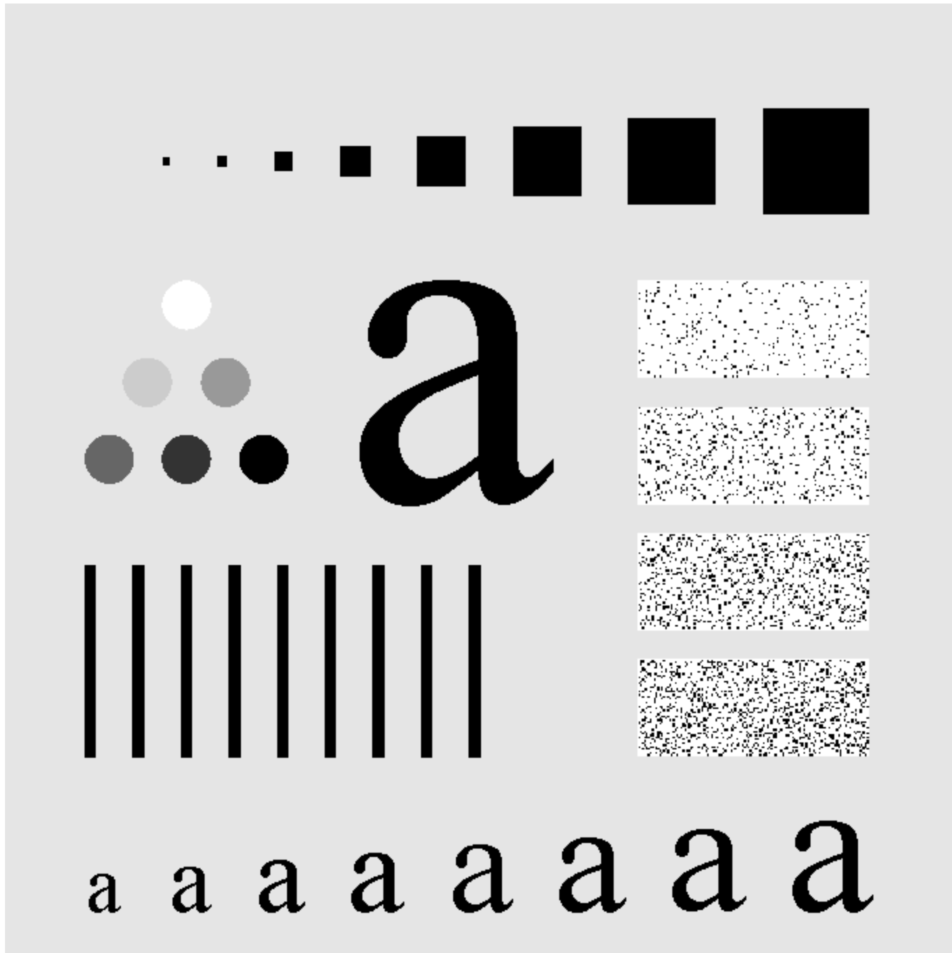
Scilab code AP 61

Image Smoothing using Gaussian Lowpass Filter



Scilab code AP 62

Image Smoothing with a Butterworth Lowoass Filter

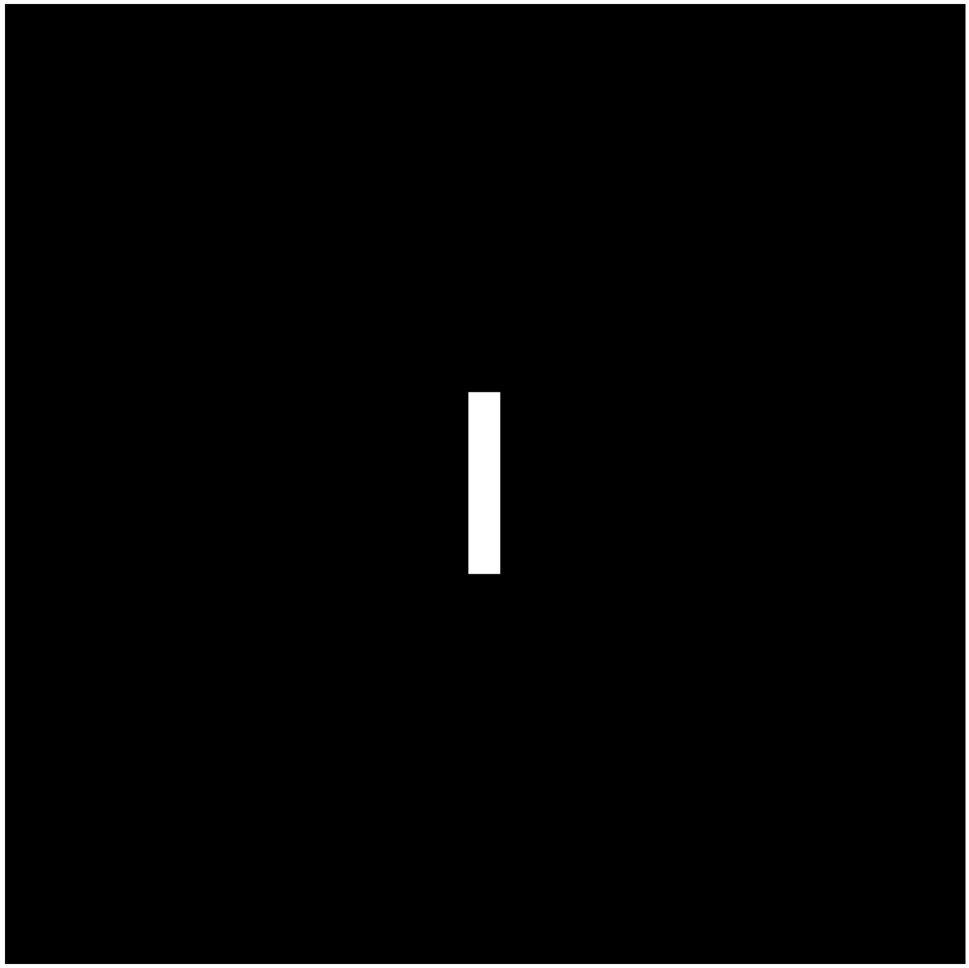


Scilab code AP 63

Image Smoothing using an ILPF

Scilab code AP 64

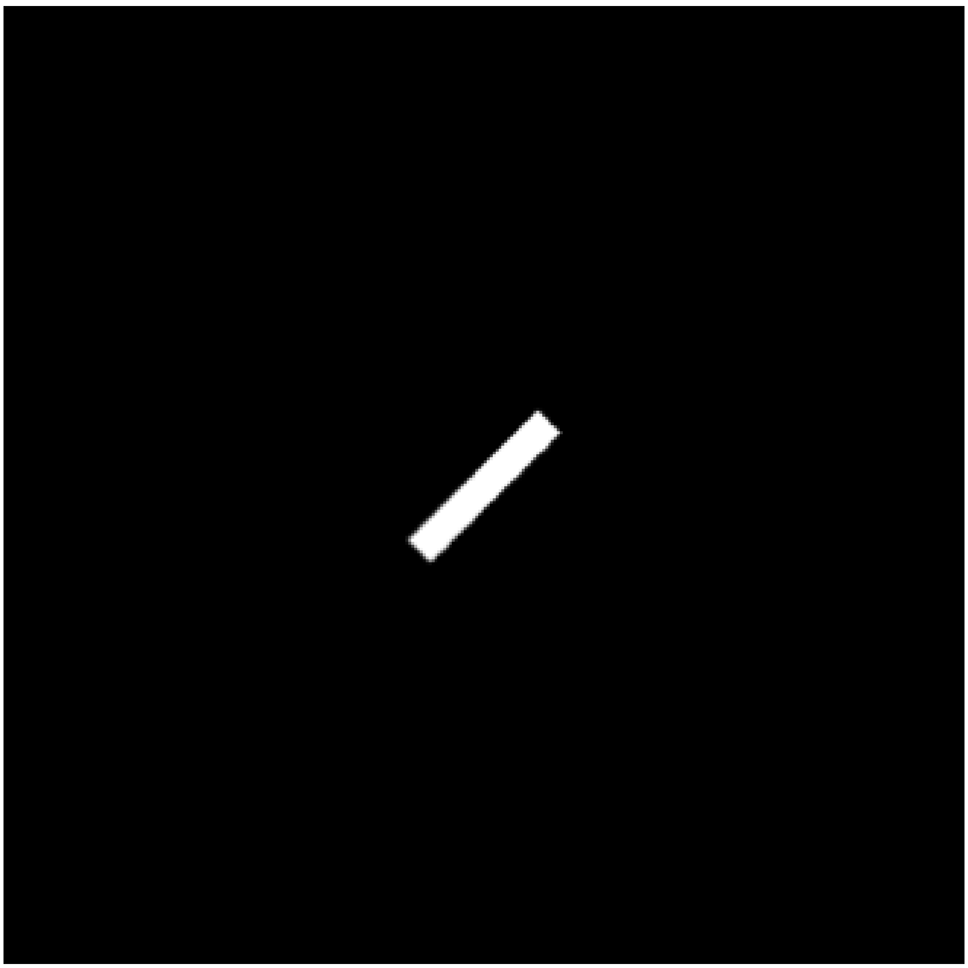
Obtaining a Frequency Domain Filtering from a Small Spatial Mask



Scilab code AP 65
Illustration of the Properties of the Fourier Spectrum and Phase Angle

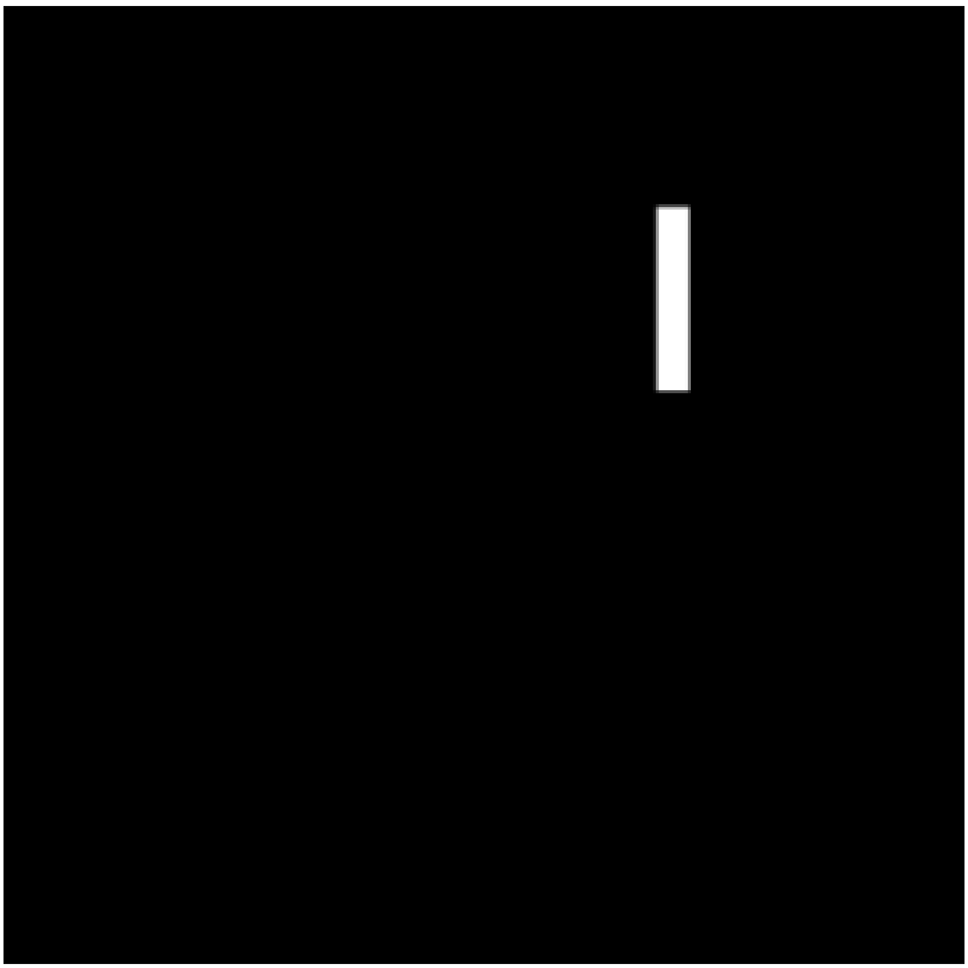
Scilab code AP 66

Illustration of the Properties of the Fourier Spectrum and Phase Angle



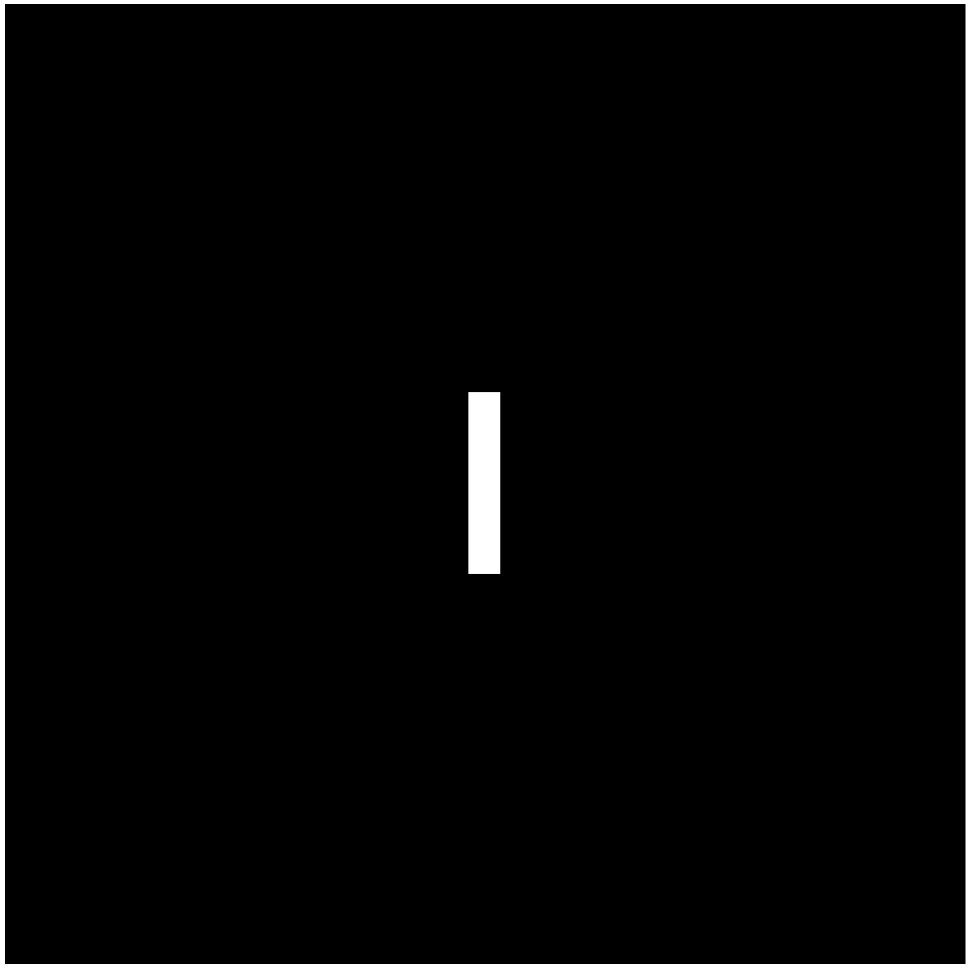
2 D

Fourier Spectrum of a Simple Function



2 D

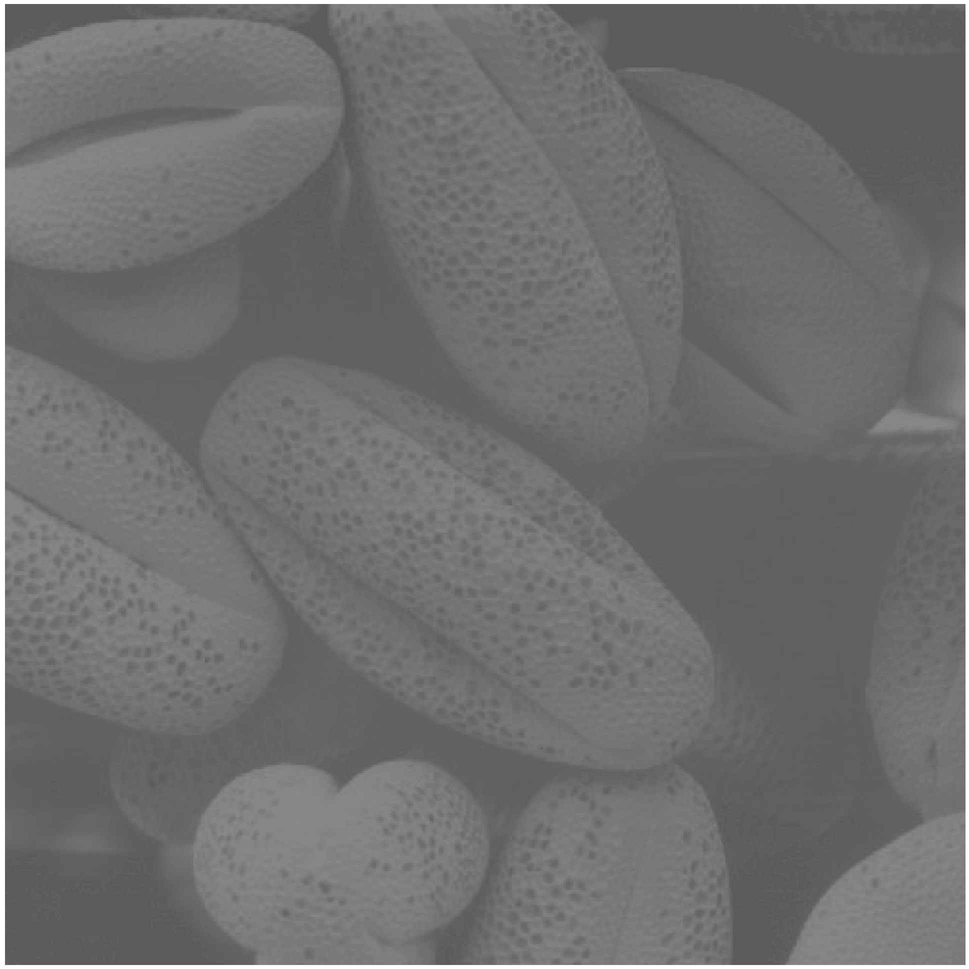
Fourier Spectrum of a Simple Function



Scilab code AP 69
2 D Fourier Spectrum of a Simple Function

Scilab code AP 70 Histogram Equalization

Scilab code AP 71
Histogram Equalization

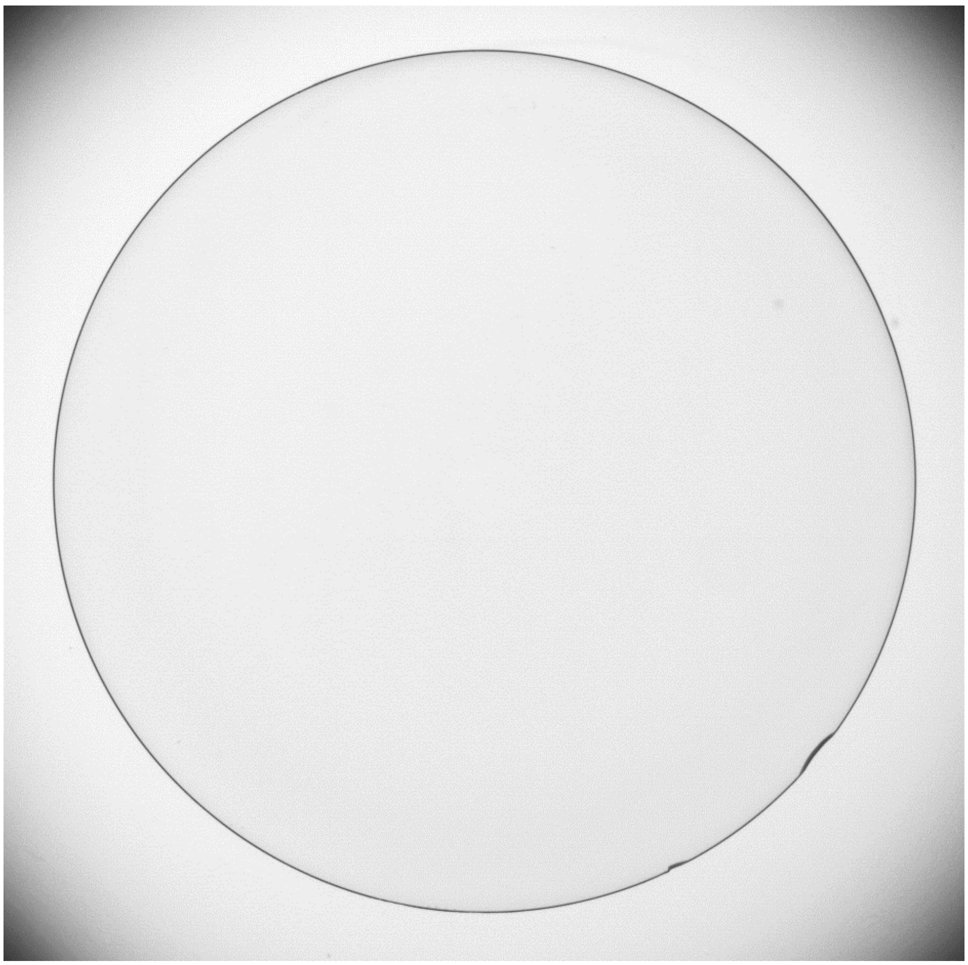


Scilab code AP 72

Histogram Equalization

Scilab code AP 73
Intensity Level Slicing

Scilab code AP 74 Illustration of Power Law Transformation

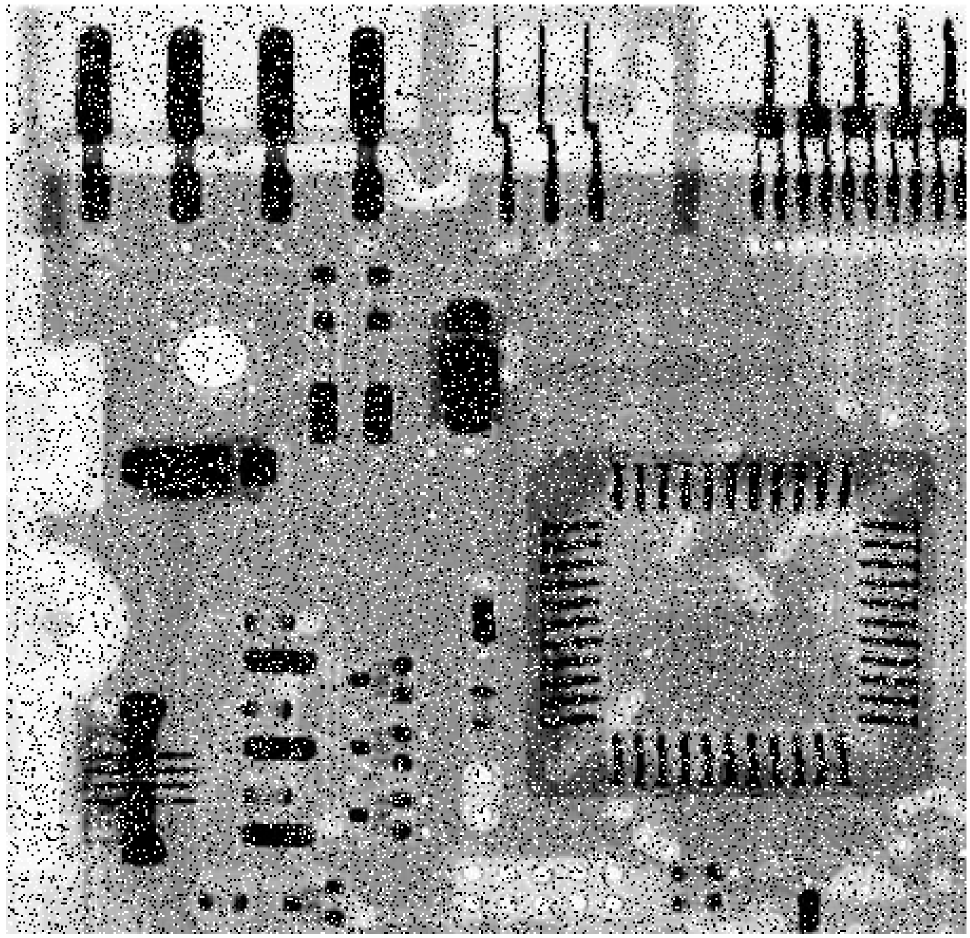


Use of gradient for Edge Enhancement

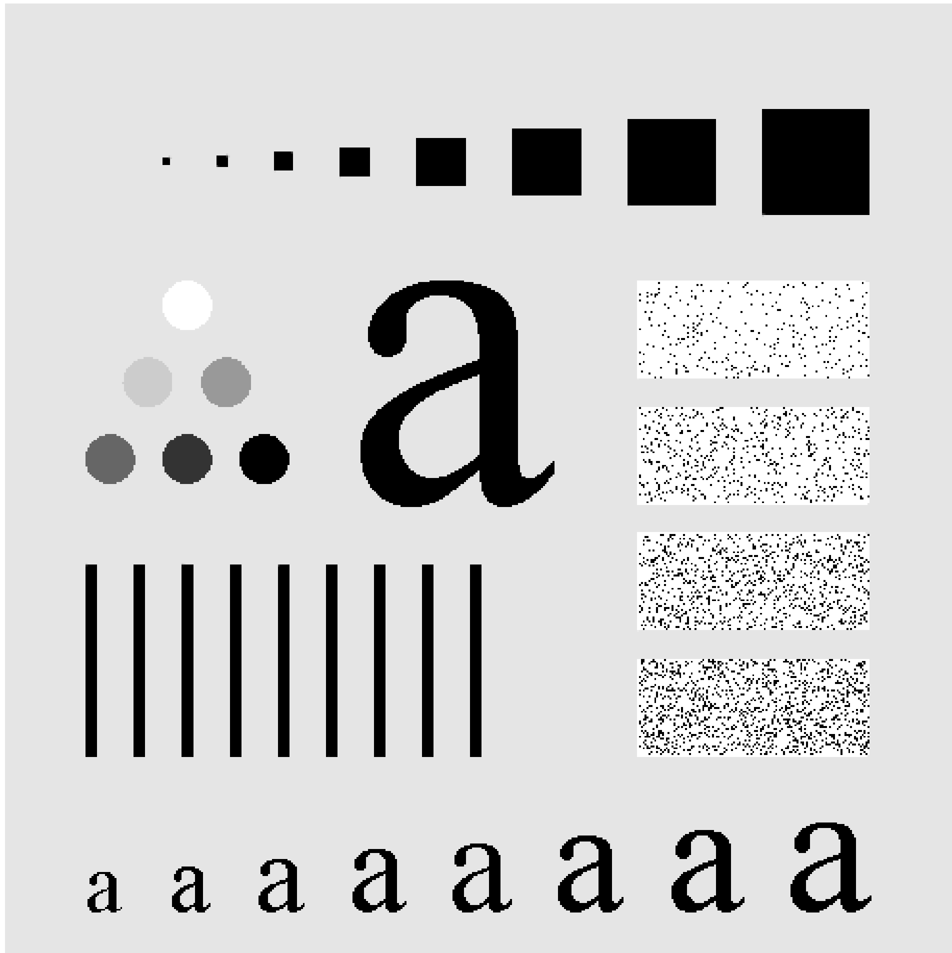
Scilab code AP 76

Image Sharpning using Un-Sharp Masking and High-Boost Filtering

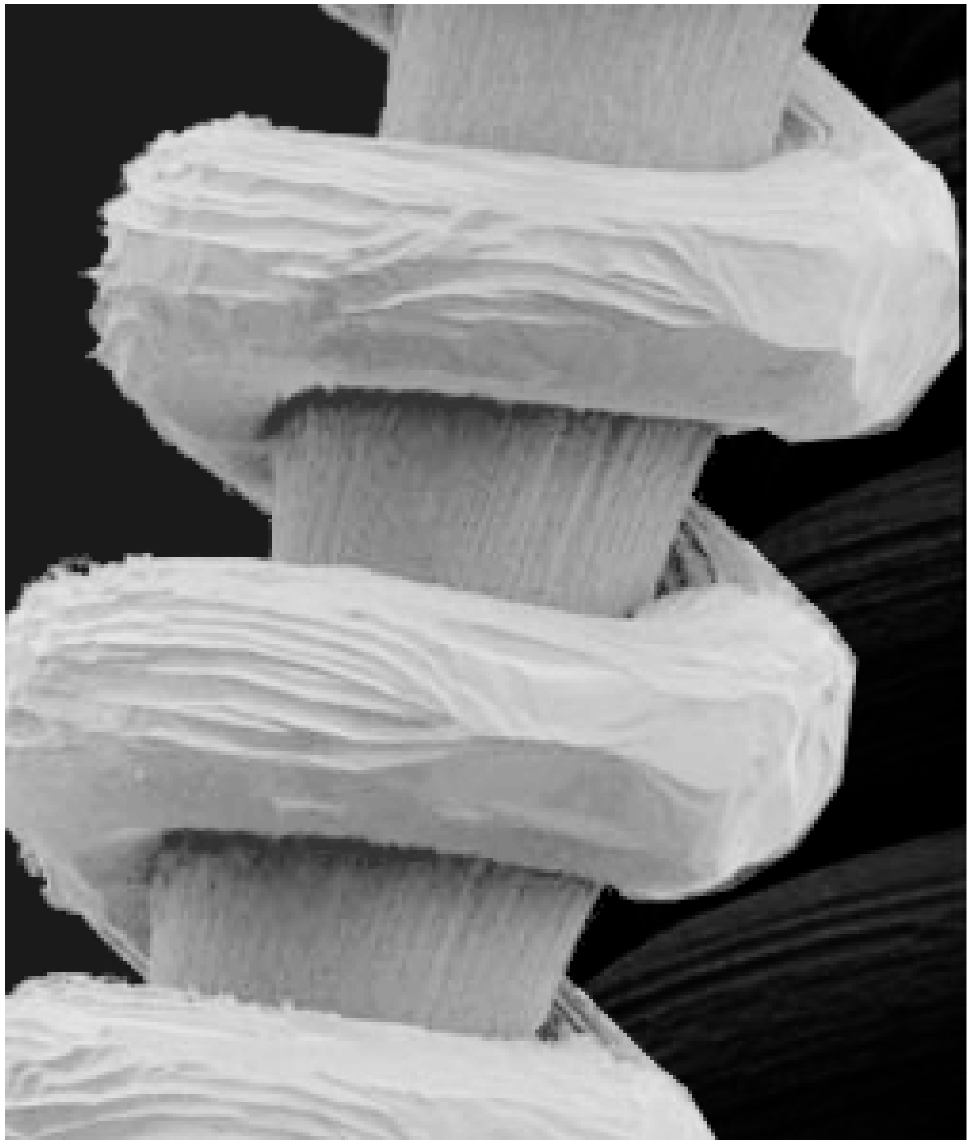
Scilab code AP 77 Image Sharpning using Laplacian



Scilab code AP 78 Median Filtering for Noise Reduction

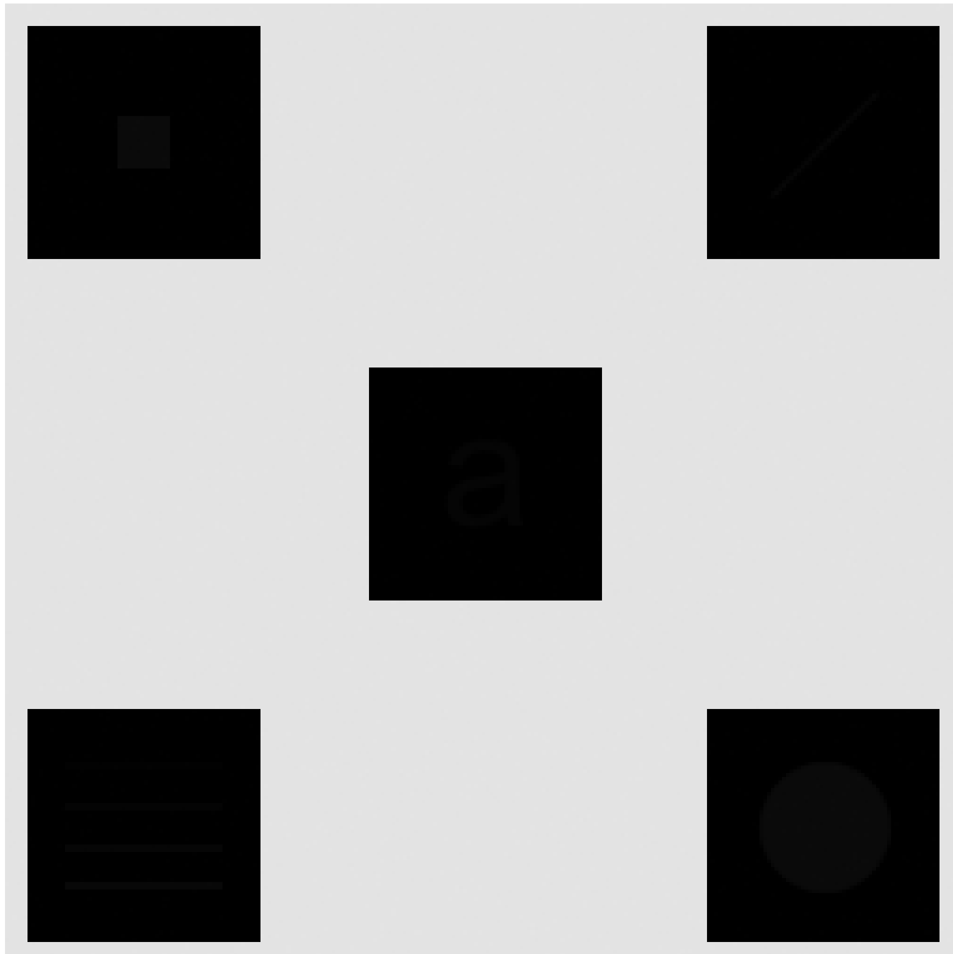


Scilab code AP 79
Image Smoothing



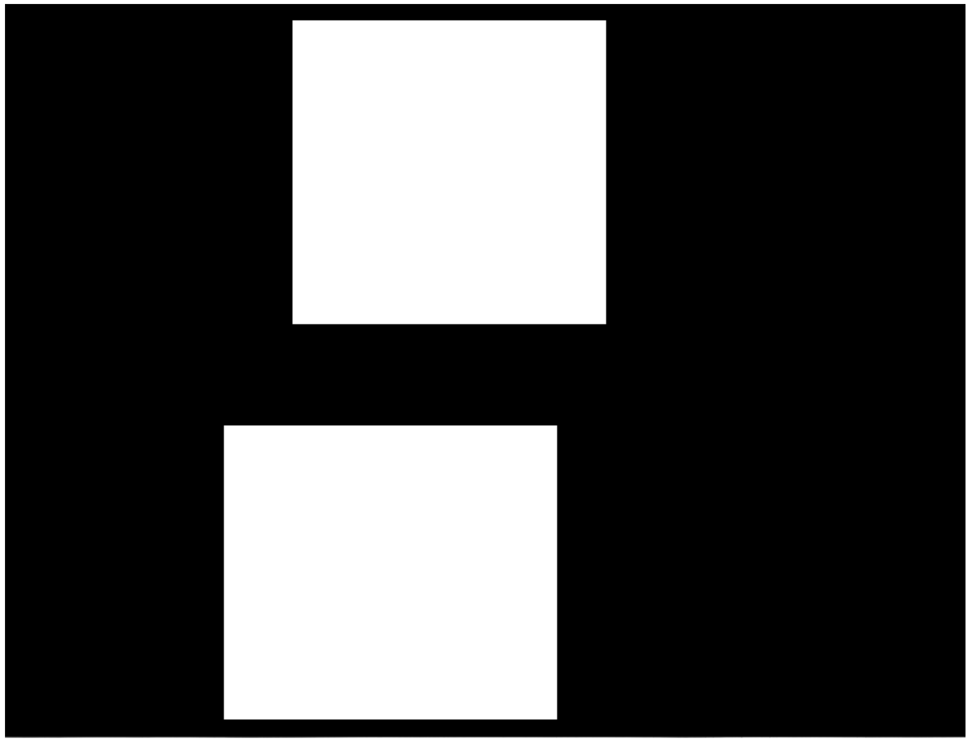
Scilab code AP 80

Local Enhancement using Histogram Statistic

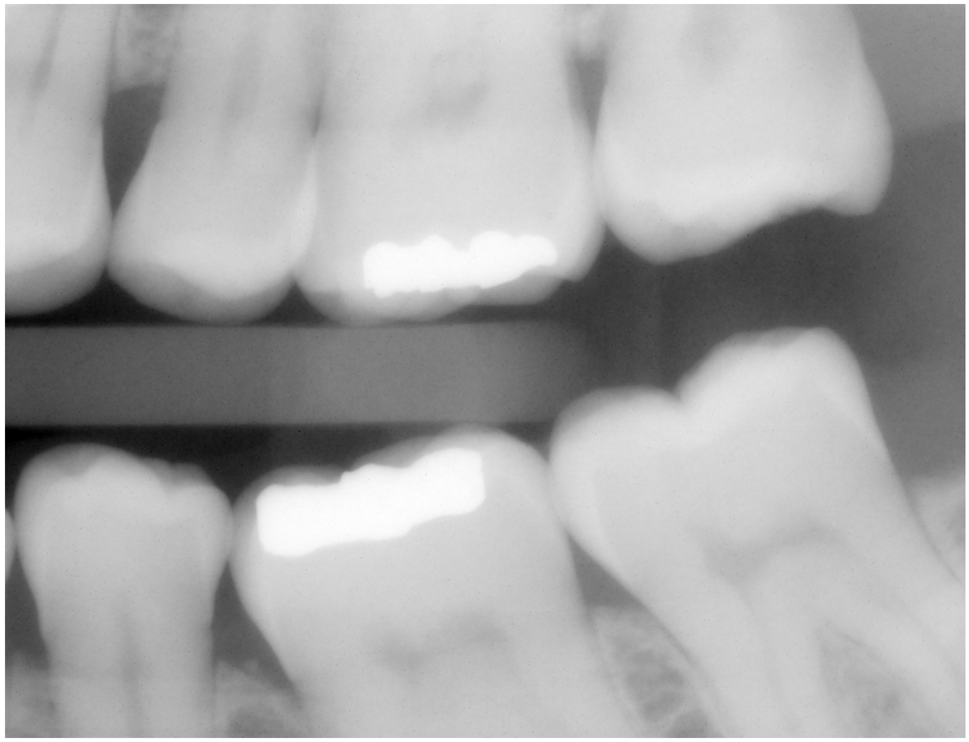


Scilab code AP 81
Local Histogram Equalization

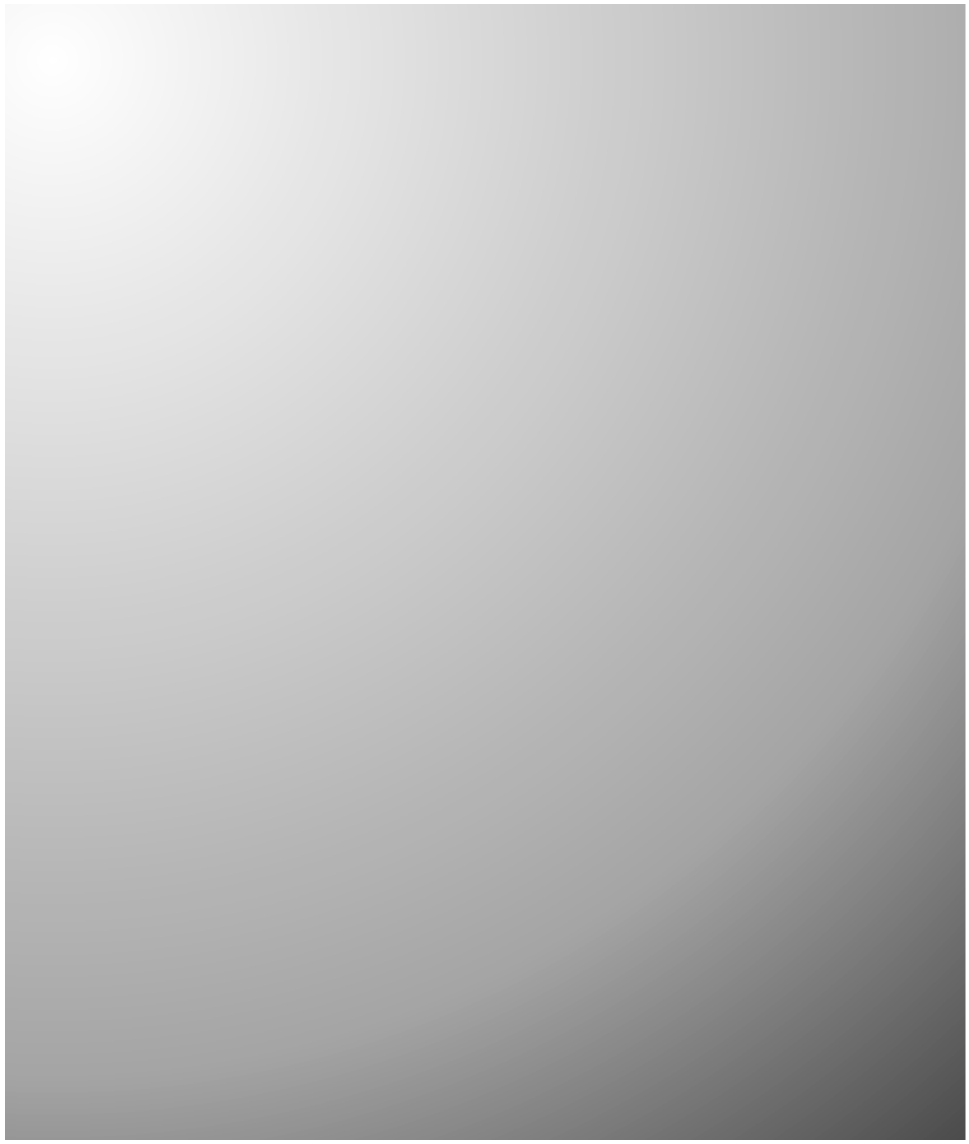
Scilab code AP 82
Gamma Intensity transformation



Scilab code AP 83
Image Multiplication for Shadding Correction

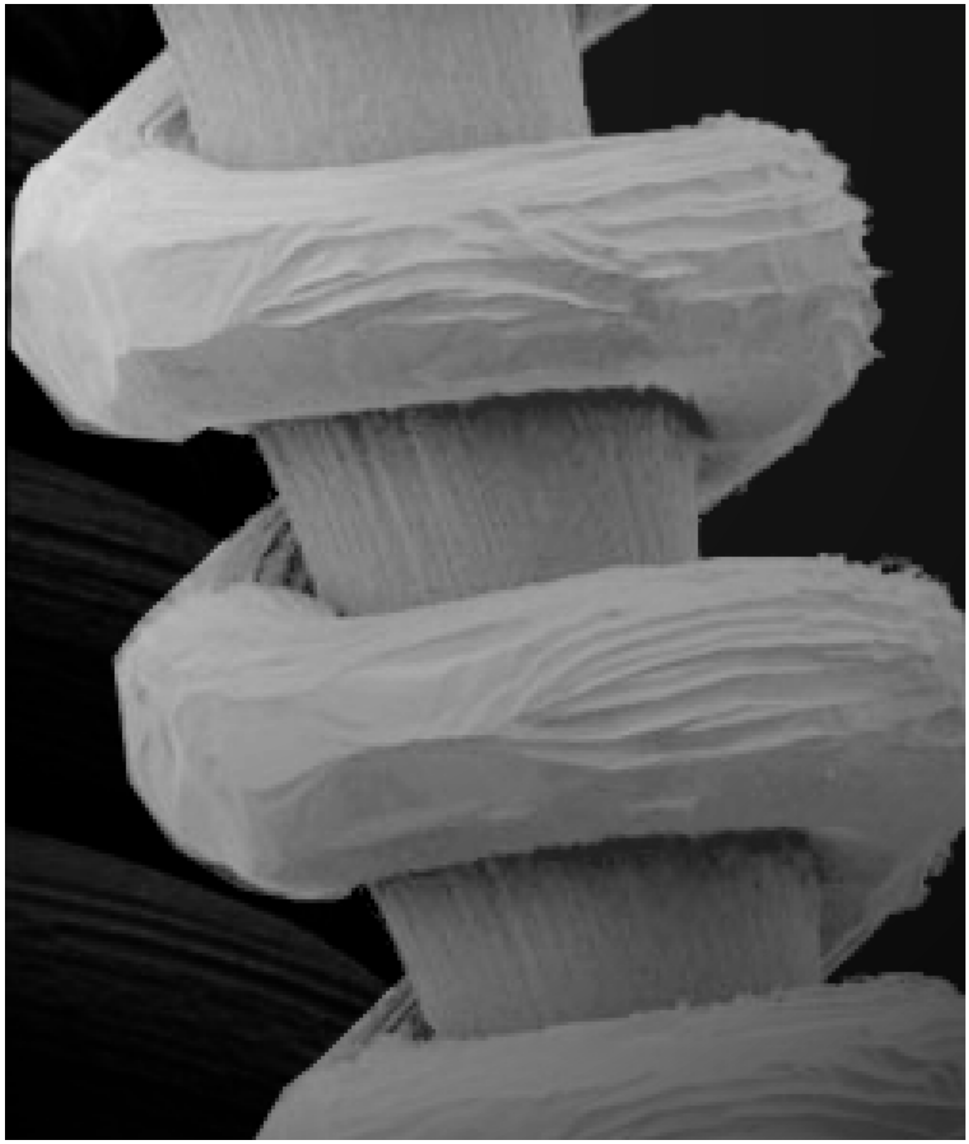


Scilab code AP 84 Image Multiplication for Shadding Correction



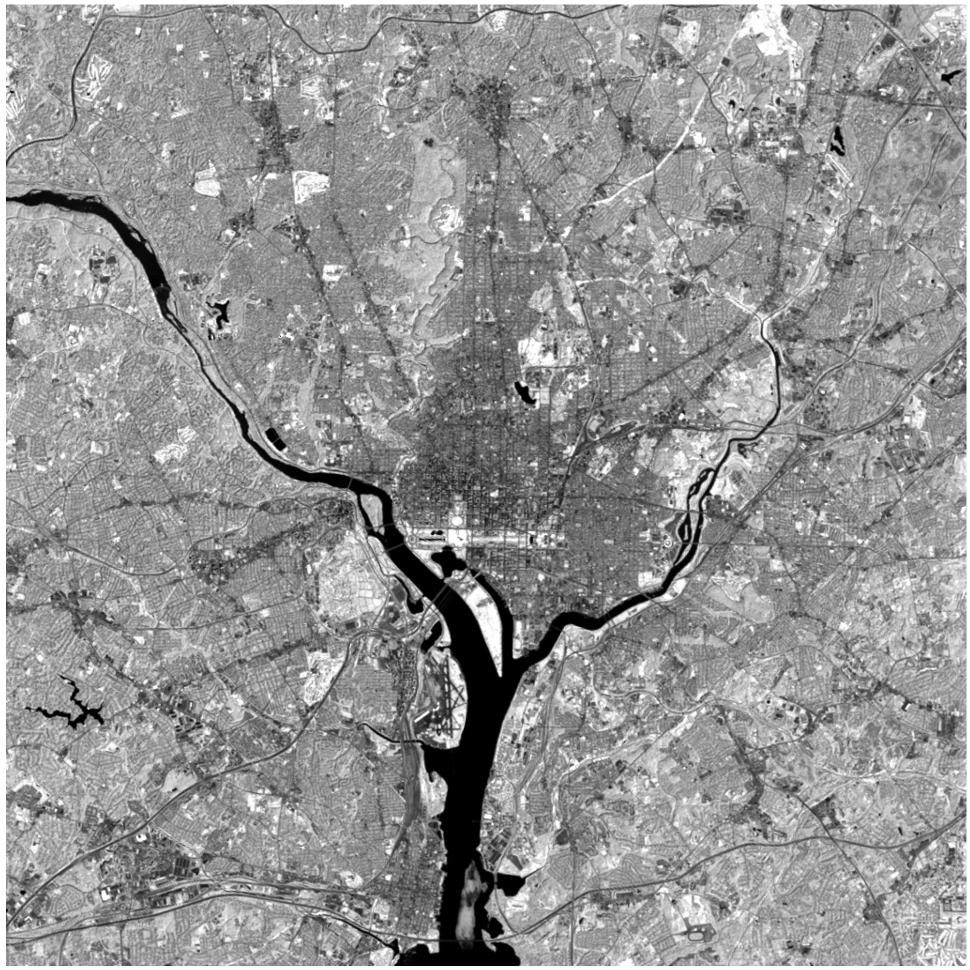
Scilab code AP 85

Image Multiplication for Shadding Correction



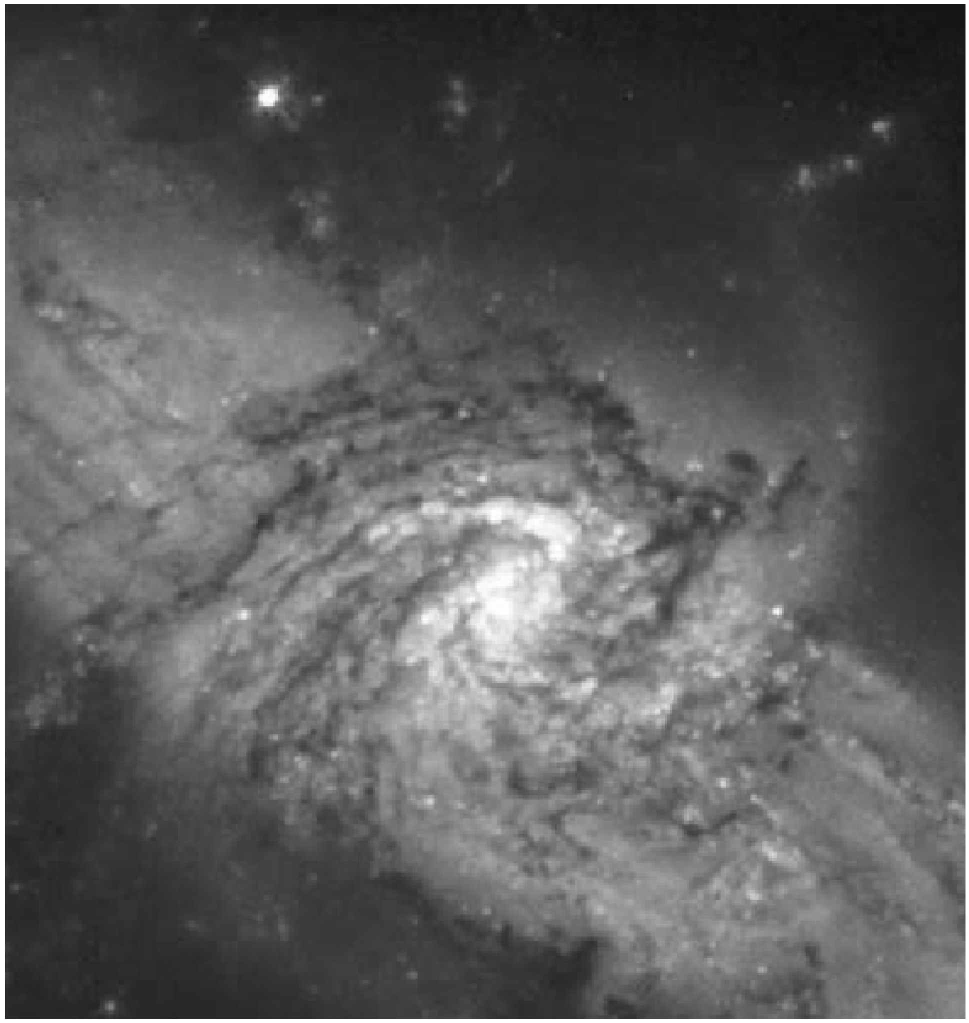
Scilab code AP 86

Image Multiplication for Shadding Correction



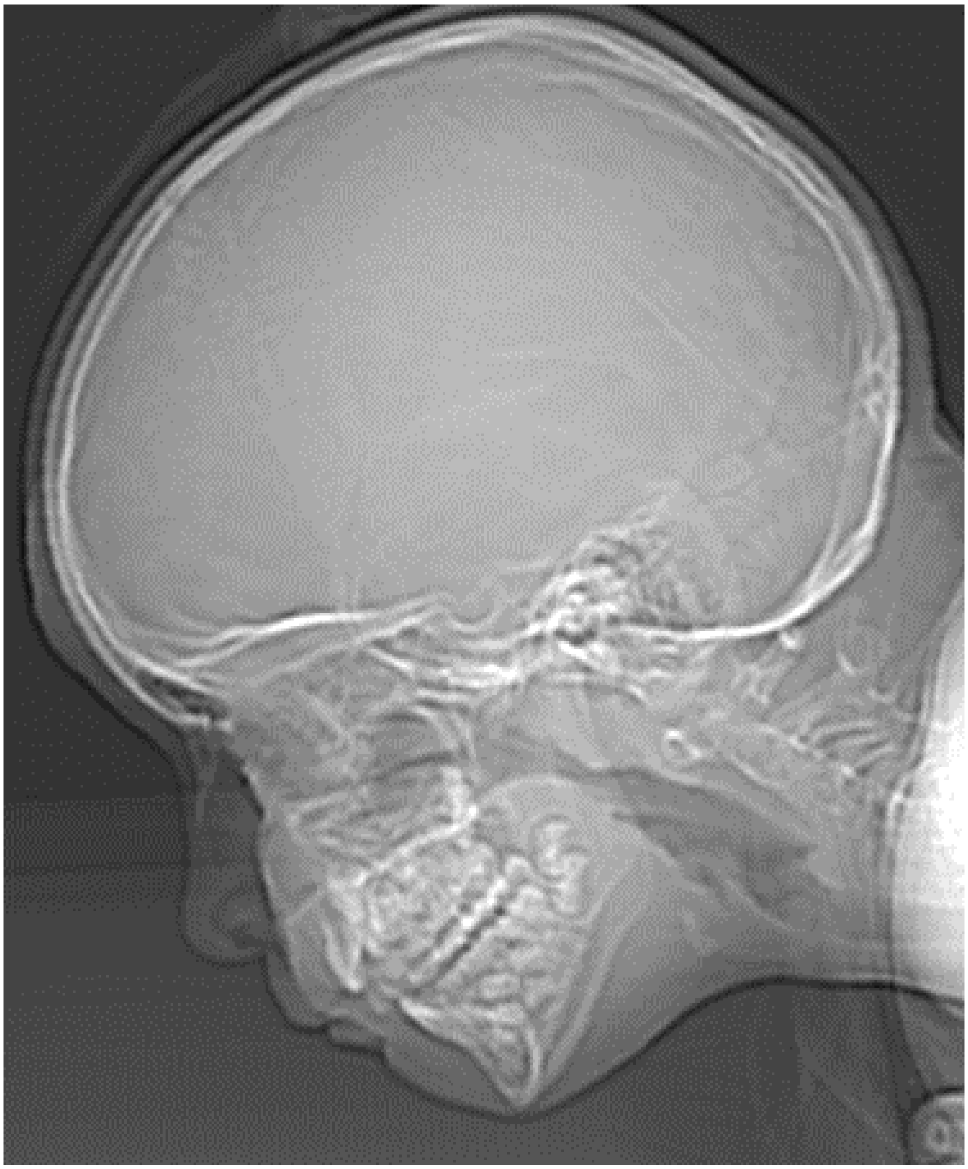
Scilab code AP 87

Image Subtraction for Enhancing differences

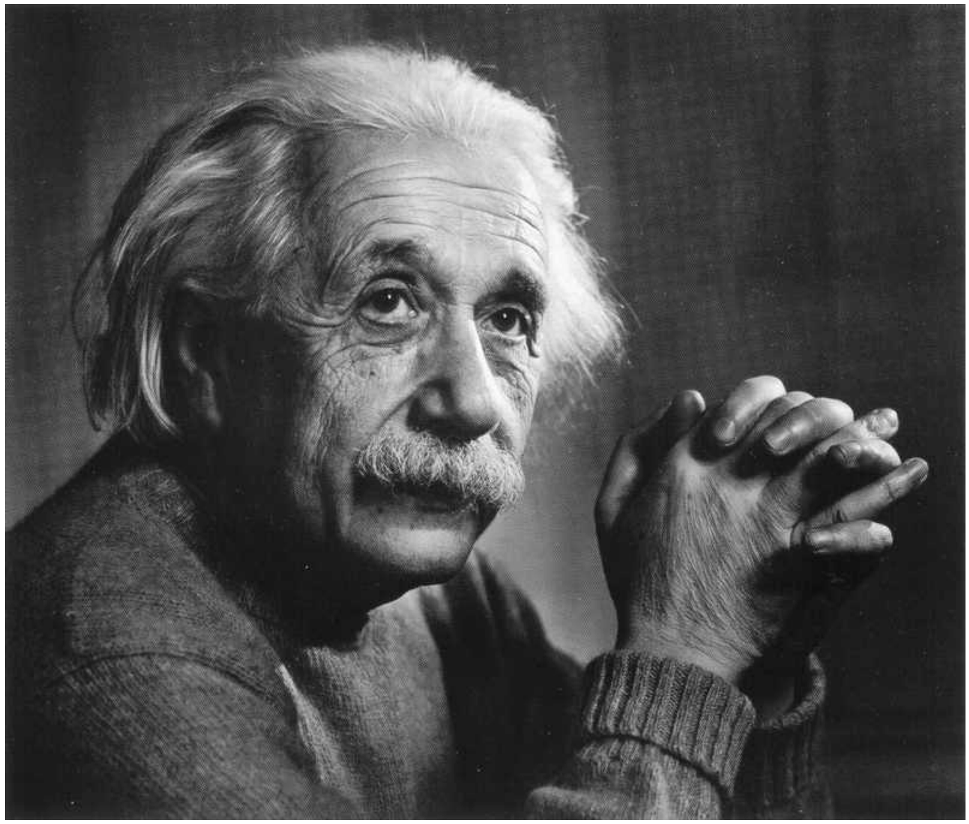


Scilab code AP 88
Addition of Noisy Images for Noise Reduction

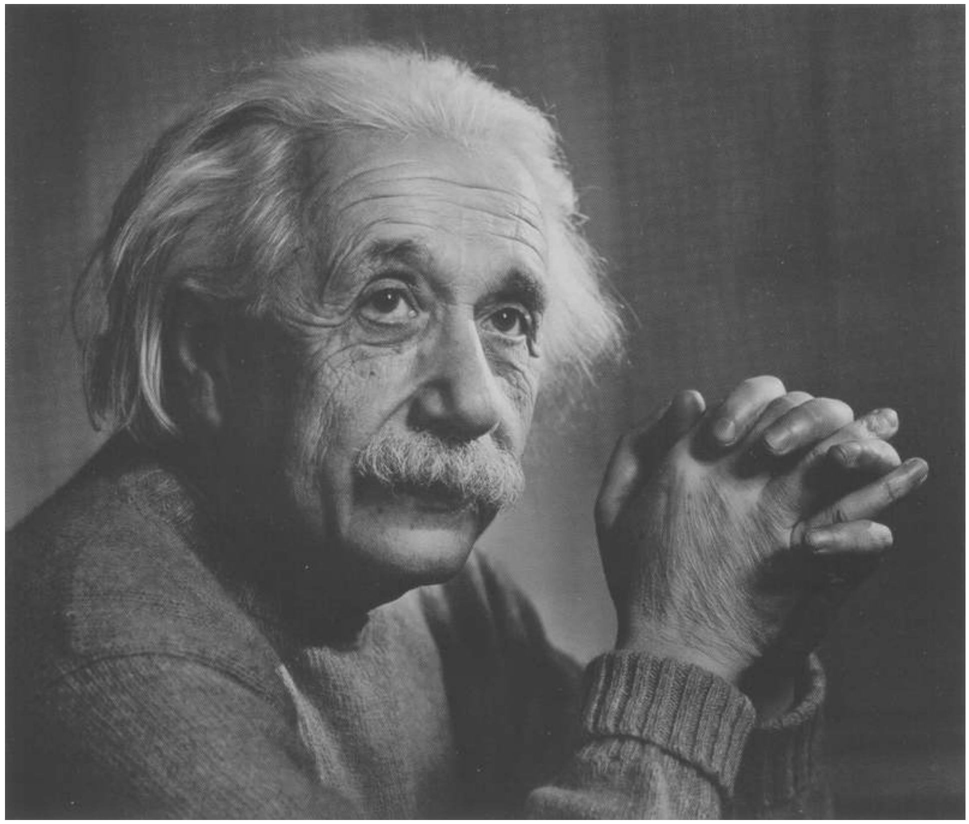
Scilab code AP 89
Comparision of Interpolation Approaches for Image Shrinking and Zooming


Typical Effects of Varying the Number of Intensity Levels in a digital Image

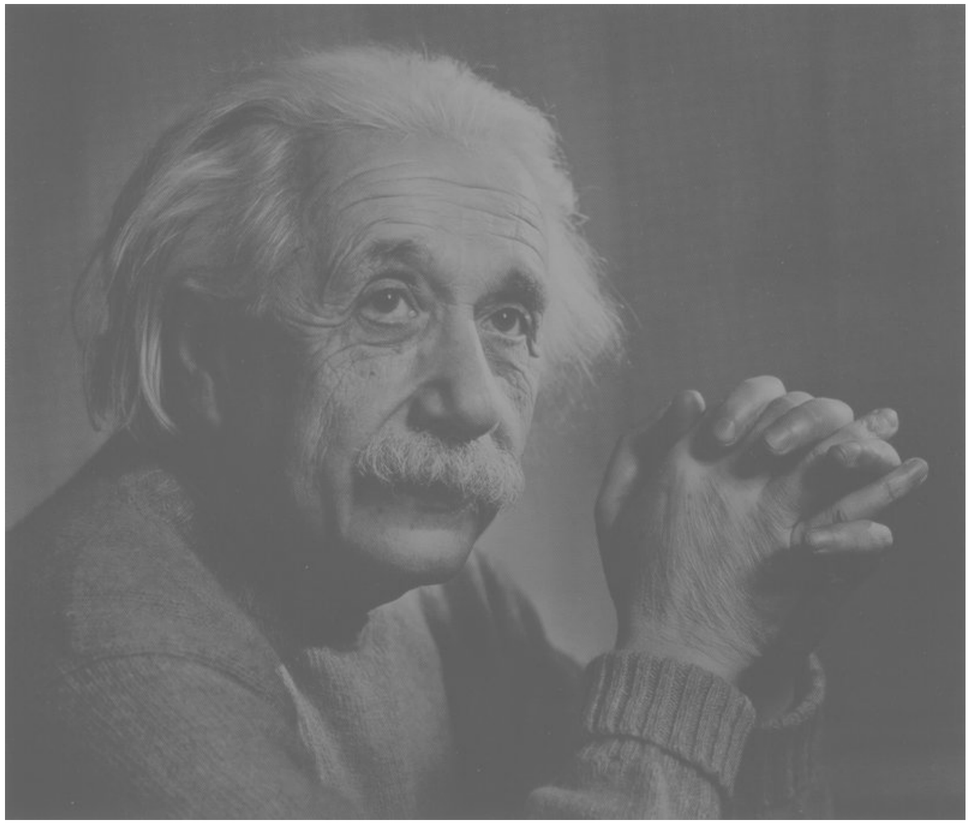
Scilab code AP 91
Illustration of the Effects of Reducing Image Spatial Resolution



Scilab code AP 92
Standard Deviation



Scilab code AP 93
Standard Deviation



Scilab code AP 94
Standard Deviation