

Scilab Textbook Companion for  
Digital Image Processing  
by R. C. Gonzalez and R. E. Woods<sup>1</sup>

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

July 2, 2020

<sup>1</sup>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	4
2 Digital Image Fundamental	6
3 Intensity Transformation and Spatial Filtering	17
4 Filtering in Frequency Domain	44
5 Image Restoration and Reconstruction	100
6 Color Image Processing	137
8 Image Compression	164
9 Morphological Image Processing	166
10 Image Segmentation	180

# List of Scilab Codes

Exa 2.2	Illustration of the Effects of Reducing Image Spatial Resolution . . . . .	6
Exa 2.3	Typical Effects of Varying the Number of Intensity Levels in a Digital Image . . . . .	7
Exa 2.4	Comparison of Interpolation Approaches for Image Shrinking and Zooming . . . . .	8
Exa 2.5	Addition of Noisy Images for Noise Reduction	10
Exa 2.6	Image Subtraction for Enhancing differences	12
Exa 2.7	Image Multiplication for Shadding Correction	13
Exa 2.12	Standard Deviation . . . . .	15
Exa 3.1	Gamma Intensity transformation . . . . .	17
Exa 3.2	Illustration of Power Law Transformation .	18
Exa 3.3	Intensity Level Slicing . . . . .	19
Exa 3.5	A Simple Illustration of Histogram Equalization . . . . .	20
Exa 3.6	Histogram Equalization with probability calculation . . . . .	22
Exa 3.8	Histogram Specification . . . . .	24
Exa 3.10	Local Histogram Equalization . . . . .	29
Exa 3.11	Computing Histogram Statistic . . . . .	32
Exa 3.12	Local Enhancement using Histogram Statistic	33
Exa 3.13	Image Smoothing . . . . .	36
Exa 3.14	Median Filtering for Noise Reduction . . . . .	38
Exa 3.15	Image Sharpning using Laplacian . . . . .	39
Exa 3.16	Image Sharpning using UnSharp Masking and HighBoost Filtering . . . . .	40
Exa 3.17	Use of gradient for Edge Enhancement . . . . .	42

Exa 4.1	Obtaining the Fourier Transform of a Simple Function . . . . .	44
Exa 4.4	The Mechanics of Computing the DFT . . .	45
Exa 4.7	Illustration of Aliasing in Resampled Images	47
Exa 4.8	Illustration of Jaggies in Image Shrinking . .	48
Exa 4.9	Illustration of Jaggies in Image Zooming . .	50
Exa 4.13	2 D Fourier Spectrum of a Simple Function	52
Exa 4.14	Illustration of the Properties of the Fourier Spectrum and Phase Angle . . . . .	55
Exa 4.15	Obtaining a Frequency Domain Filtering from a Small Spatial Mask . . . . .	57
Exa 4.16	Image Smoothing using an ILPF . . . . .	60
Exa 4.17	Image Smoothing with a Butterworth Lowpass Filter . . . . .	66
Exa 4.18	Image Smoothing using Gaussian Lowpass Filter . . . . .	72
Exa 4.19	Using Highpass Filter and Thresholding for Image enhancement . . . . .	78
Exa 4.20	Image Sharpening in the Frequency Domain using the Laplacian . . . . .	80
Exa 4.21	Image Enhancement using High Frequency Emphasis Filtering . . . . .	85
Exa 4.22	Image Enhancement using Homomorphic Filtering . . . . .	88
Exa 4.23	Reduction of Moire Patterns Using Notch Filtering . . . . .	92
Exa 4.24	Enhancement of Corrupted Cassini Saturn Image by Notch Filtering . . . . .	95
Exa 5.1	Noisy Images and their Histogram . . . . .	100
Exa 5.2	Illustration of Mean Filters . . . . .	106
Exa 5.3	Illustration of Order Statistic filter . . . . .	113
Exa 5.4	Illustration of Adaptive Local Noise Reduction Filtering . . . . .	120
Exa 5.5	Illustration of Adaptive Median Filter . . .	123
Exa 5.8	Removal of Periodic Noise by Notch Filtering	125
Exa 5.10	Image Blurring Due to Motion . . . . .	128
Exa 5.11	Inverse Filtering . . . . .	129

Exa 5.12	Comparison of Inverse Filtering and Wiener Filtering . . . . .	133
Exa 6.3	Intensity Slicing . . . . .	137
Exa 6.4	Use of Color to Highlight Rainfall Levels . .	138
Exa 6.5	Use of Psuedocolor for highlighting Explosives Contained in Luggage . . . . .	140
Exa 6.6	Color Coding of Multi Spectral Images . . .	142
Exa 6.7	Computing Color Image Components . . . .	143
Exa 6.9	Tonal Transformations . . . . .	145
Exa 6.10	Color Balancing . . . . .	147
Exa 6.11	Histogram Equalization in the HSI Color Space	151
Exa 6.12	Color Image Smoothning by Neighbourhood Averaging . . . . .	153
Exa 6.13	Sharpning with the Laplacian . . . . .	156
Exa 6.14	Segmentation in HSI Space . . . . .	158
Exa 6.16	Edge Detection Vector Space . . . . .	160
Exa 6.17	Illustration of the effects of converting noisy RGB Images to HSV . . . . .	162
Exa 8.2	Image Entropy Estimation . . . . .	164
Exa 9.1	Using Erosion to remove image component .	166
Exa 9.2	An Illustration of Dilation . . . . .	167
Exa 9.4	Use of opening and closing for Morphological Filtering . . . . .	168
Exa 9.5	Boundary Extraction by Morphological Processing . . . . .	170
Exa 9.7	Using Connected Components to Detect Foreign Object in Packaged Food . . . . .	171
Exa 9.9	Illustration of Gray Scale Erosion and Dilation	173
Exa 9.10	Illustration of Gray Scale Opening and Closing	176
Exa 10.1	Detection of Isolated Point in an Image . . .	180
Exa 10.2	Using the Laplacian for the Detection . . . .	181
Exa 10.3	Detection of Lines in Specified Direction . .	182
Exa 10.4	Behavior of the First and Second Derivative of a Noisy Edge . . . . .	185
Exa 10.6	Illustration of the 2 D Gradient Magnitude and Angle . . . . .	188
Exa 10.7	Illustration of the Marr Hildreth Edge Detection Methods . . . . .	190

Exa 10.8	Illustration of the Canny Edge Detection Methods . . . . .	191
Exa 10.9	Another illustration of the three principal Edge Detection Methods . . . . .	194
Exa 10.15	Global Thresholding . . . . .	198
Exa 10.16	Optimum Global Thresholding using Otsu Method	199
Exa 10.18	Using Edge Information Based on the Laplacian to Improve Global Thresholding . . . . .	200
Exa 10.19	Multipal Global Thresholding . . . . .	204
Exa 10.20	Variable Thresholding Via Image Partitioning	206
Exa 10.22	Document Thresholding Using Moving Averages . . . . .	209
Exa 10.23	Segmentation by Region Growing . . . . .	212
AP 1	Another illustration of the three principal Edge Detection Methods . . . . .	217
AP 2	Illustration of the Canny Edge Detection Methods . . . . .	218
AP 3	Illustration of the Marr Hildreth Edge Detection Methods . . . . .	219
AP 4	Illustration of the 2 D Gradient Magnitude and Angle . . . . .	220
AP 5	Behavior of the First and Second Derivative of a Noisy Edge . . . . .	220
AP 6	Detection of Lines in Specified Direction . . . . .	221
AP 7	Document Thresholding Using Moving Averages . . . . .	222
AP 8	Variable Thresholding Via Image Partitioning	223
AP 9	Using the Laplacian for the Detection . . . . .	224
AP 10	Multipal Global Thresholding . . . . .	225
AP 11	Using Edge Information Based on the Laplacian to Improve Global Thresholding . . . . .	226
AP 12	Optimum Global Thresholding using Otsu's Method . . . . .	227
AP 13	Global Thresholding . . . . .	228
AP 14	Detection of Isolated Point in an Image . . . . .	229
AP 15	Illustration of Gray Scale Erosion and Dilation	230
AP 16	Using Connected Components to Detect Foreign Object in Packaged Food . . . . .	231



AP 19	An Illustration of Dilation . . . . .	231
AP 17	Boundary Extraction by Morphological Processing . . . . .	232
AP 18	Use of opening and closing for Morphological Filtering . . . . .	233
AP 20	Illustration of Gray Scale Opening and Closing	234
AP 21	Using Erosion to remove image component .	235
AP 22	Image Entropy Estimation . . . . .	236
AP 23	Tonal Transformations . . . . .	237
AP 24	Tonal Transformations . . . . .	238
AP 25	Tonal Transformations . . . . .	239
AP 26	Computing Color Image Components . . . .	240
AP 27	Color Coding of Multi Spectral Images . . .	241
AP 28	Color Coding of Multi Spectral Images . . .	242
AP 29	Color Coding of Multi Spectral Images . . .	243
AP 30	Color Coding of Multi Spectral Images . . .	244
AP 31	Use of Pseudocolor for highlighting Explosives Contained in Luggage . . . . .	245
AP 32	Use of Color to Highlight Rainfall Levels . .	246
AP 33	Intensity Slicing . . . . .	247
AP 34	Illustration of the effects of converting noisy RGB Images to HSI . . . . .	248
AP 35	Illustration of the effects of converting noisy RGB Images to HSI . . . . .	249
AP 36	Illustration of the effects of converting noisy RGB Images to HSI . . . . .	250
AP 37	Edge Detection Vector Space . . . . .	251
AP 38	Segmentation in HSI Space . . . . .	252
AP 39	Sharpening with the Laplacian . . . . .	253
AP 40	Color Image Smoothing by Neighbourhood Averaging . . . . .	254
AP 41	Histogram Equalization in the HSI Color Space	255
AP 42	Color Balancing . . . . .	256
AP 43	Removal of Periodic Noise by Notch Filtering	257
AP 44	Illustration of Adaptive Median Filter . . .	258
AP 45	Illustration of Adaptive Local Noise Reduction Filtering . . . . .	259
AP 46	Illustration of Order Statistic filter . . . . .	260

AP 47	Illustration of Mean Filters . . . . .	261
AP 48	Comparison of Inverse Filtering and Wiener Filtering . . . . .	262
AP 49	Inverse Filtering . . . . .	263
AP 50	Image Blurring Due to Motion . . . . .	264
AP 51	Noisy Images and their Histogram . . . . .	265
AP 52	Illustration of Jaggies in Image Zooming . . . . .	266
AP 53	Illustration of Jaggies in Image Shrinking . . . . .	267
AP 54	Illustration of Aliasing in Resampled Images . . . . .	268
AP 55	Enhancement of Corrupted Cassini Saturn Image by Notch Filtering . . . . .	269
AP 56	Reduction of Moire Patterns Using Notch Filtering . . . . .	270
AP 57	Image Enhancement using Homomorphic Filtering . . . . .	272
AP 58	Image Enhancement using High Frequency Emphasis Filtering . . . . .	272
AP 59	Image Sharpning in the Frequency Domain using the Laplacian . . . . .	273
AP 60	Using Highpass Filter and Thresholding for Image enhancement . . . . .	274
AP 61	Image Smoothing using Gaussian Lowpass Filter . . . . .	275
AP 62	Image Smoothing with a Butterworth Lowpass Filter . . . . .	276
AP 63	Image Smoothing using an ILPF . . . . .	277
AP 64	Obtaining a Frequency Domain Filtering from a Small Spatial Mask . . . . .	278
AP 65	Illustration of the Properties of the Fourier Spectrum and Phase Angle . . . . .	279
AP 66	Illustration of the Properties of the Fourier Spectrum and Phase Angle . . . . .	280
AP 67	2 D Fourier Spectrum of a Simple Function . . . . .	281
AP 68	2 D Fourier Spectrum of a Simple Function . . . . .	282
AP 69	2 D Fourier Spectrum of a Simple Function . . . . .	283
AP 70	Histogram Equalization . . . . .	284
AP 71	Histogram Equalization . . . . .	285
AP 72	Histogram Equalization . . . . .	286

AP 73	Intensity Level Slicing . . . . .	287
AP 74	Illustration of Power Law Transformation . . . . .	288
AP 75	Use of gradient for Edge Enhancement . . . . .	289
AP 76	Image Sharpning using Un-Sharp Masking and High-Boost Filtering . . . . .	290
AP 77	Image Sharpning using Laplacian . . . . .	291
AP 78	Median Filtering for Noise Reduction . . . . .	292
AP 79	Image Smoothing . . . . .	293
AP 80	Local Enhancement using Histogram Statistic	294
AP 81	Local Histogram Equalization . . . . .	295
AP 82	Gamma Intensity transformation . . . . .	296
AP 83	Image Multiplication for Shadding Correction	297
AP 84	Image Multiplication for Shadding Correction	298
AP 85	Image Multiplication for Shadding Correction	299
AP 86	Image Multiplication for Shadding Correction	300
AP 87	Image Subtraction for Enhancing differences	301
AP 88	Addition of Noisy Images for Noise Reduction	302
AP 89	Comparision of Interpolation Approaches for Image Shrinking and Zooming . . . . .	303
AP 90	Typical Effects of Varying the Number of Intensity Levels in a digital Image . . . . .	304
AP 91	Illustration of the Effects of Reducing Image Spatial Resolution . . . . .	305
AP 92	Standard Deviation . . . . .	306
AP 93	Standard Deviation . . . . .	307
AP 94	Standard Deviation . . . . .	308

# List of Figures

3.1	Histogram Specification	27
3.2	Histogram Specification	28
3.3	Local Histogram Equalization	29
3.4	Local Histogram Equalization	30
4.1	Image Smoothing using an ILPF	64
4.2	Image Smoothing using an ILPF	65
4.3	Image Smoothing with a Butterworth Lowpass Filter	70
4.4	Image Smoothing with a Butterworth Lowpass Filter	71
4.5	Image Smoothing using Gaussian Lowpass Filter	76
4.6	Image Smoothing using Gaussian Lowpass Filter	77
4.7	Using Highpass Filter and Thresholding for Image enhancement	81
4.8	Using Highpass Filter and Thresholding for Image enhancement	82
4.9	Image Enhancement using High Frequency Emphasis Filtering	89
4.10	Image Enhancement using High Frequency Emphasis Filtering	90
4.11	Reduction of Moire Patterns Using Notch Filtering	96
4.12	Reduction of Moire Patterns Using Notch Filtering	96
10.1	Using the Laplacian for the Detection	183
10.2	Using the Laplacian for the Detection	184
10.3	Optimum Global Thresholding using Otsu Method	201
10.4	Optimum Global Thresholding using Otsu Method	202

10.5 Document Thresholding Using Moving Averages . . . . .	211
10.6 Segmentation by Region Growing . . . . .	215
10.7 Segmentation by Region Growing . . . . .	216

# 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

```
1 //Ex2_2
2 //Illustration of the Effects of Reducing Image
   Spatial Resolution
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
   Woods
9
10 clc;
11 close;
12 clear;
13 xdel(winsid())//to close all currently open figure(s
   ).
14 gray=imread("Ex2_2.tif");
```

```

15 figure,ShowImage(gray,'Gray Image');
16 title('Original Image (1250 DPI)');
17 [M,N]=size(gray);
18 a1=imresize(gray,[443 337],'nearest');
19 figure,ShowImage(a1,'Resize Image');
20 title('Resize Image (300 DPI)');
21
22 a2=imresize(gray,[886 675],'nearest');
23 figure,ShowImage(a2,'Resize Image');
24 title('Resize Image (150 DPI)');
25
26 a3=imresize(gray,[213 162],'nearest');
27 figure,ShowImage(a3,'Resize Image');
28 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

```

1 //Ex2_3
2 //Typical Effects of Varying the Number of Intensity
   Levels in a digital Image
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
   Woods
9
10 clc;
11 close;
12 clear;

```

```

13 xdel(winsid())//to close all currently open figure(s
    ).
14 gray=rgb2gray(imread("Ex2_3.png"));
15 figure,ShowImage(gray,'Gray Image');
16 title('Original Image');
17 [nr nc]=size(gray);
18 b=gray;
19 level=[128 64 32 16 8 4 2];
20 for x=1:length(level)
21 k=level(x);
22 for y=1:k
23 for i=1:nr
24     for j=1:nc
25         if(gray(i,j)>=((255/k)*(y-1)) & gray(i,j)
                <((255/k)*y))
26             b(i,j)=((255/k)*(y-1))+((255/k)/2);
27         end
28     end
29 end
30 end
31 figure,ShowImage(b,'OutPut Image');
32 title('Image With Less Number of Gray Level');
33 end

```

---

check Appendix ?? for dependency:

Ex2\_4.tif

**Scilab code Exa 2.4** Comparison of Interpolation Approaches for Image Shrinking and

```

1 //Ex2_4
2 //Comparison of Interpolation Approaches for Image
    Shrinking and Zooming
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1

```



```

6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
  Woods
9
10 clc;
11 close;
12 clear;
13 xdel(winsid())//to close all currently open figure(s
  ).
14 gray=imread("Ex2_4.tif");
15 figure,ShowImage(gray,'Gray Image');
16 title('Original Image (1250 DPI)');
17 [M,N]=size(gray);
18
19 a2=imresize(gray,[213 162],'nearest'); //nearest
  neigubour Interpolation
20 figure,ShowImage(a2,'Resize Image');
21 title('Resize Image (72 DPI) nearest neigubour
  Interpolation');
22 a2=imresize(gray,[213 162],'bilinear'); ///
  bilinear Interpolation
23 figure,ShowImage(a2,'Resize Image');
24 title('Resize Image (72 DPI) with bilinear
  Interpolation');
25 a2=imresize(gray,[213 162],'bicubic'); //bicubic
  Interpolation
26 figure,ShowImage(a2,'Resize Image');
27 title('Resize Image (72 DPI) with bicubic
  Interpolation');
28
29 a3=imresize(gray,[886 675],'nearest'); //nearest
  neigubour Interpolation
30 figure,ShowImage(a3,'Resize Image');
31 title('Resize Image (150 DPI) with nearest neigubour
  Interpolation');
32 a3=imresize(gray,[886 675],'nearest'); ///bilinear
  Interpolation

```

```

33 figure,ShowImage(a3,'Resize Image');
34 title('Resize Image (150 DPI) with bilinear
      Interpolation');
35 a3=imresize(gray,[886 675],'nearest'); //bicubic
      Interpolation
36 figure,ShowImage(a3,'Resize Image');
37 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

```

1 //Ex2_5
2 //Addition of Noisy Images for Noise Reduction
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
      Woods
9
10 clc;
11 close;
12 clear;
13 xdel(winsid())//to close all currently open figure(s
      ).
14 gray=imread("Ex2_5.tif");
15 //gray=rgb2gray(a);
16 gray=im2double(gray);
17
18 figure,ShowImage(gray,'Gray Image');
19 title('Original Image');

```

```

20 [nr nc]=size(gray);
21 noise_image=gray;
22
23 out_image=double(zeros(nr,nc));
24 level=[5 10 20 50 100];
25 for i=1:length(level)
26 No=level(i);
27 disp(No);
28 for k=1:No
29     noisy_image=imnoise(noise_image,'gaussian'
        ,0,0.02);
30 // figure,ShowImage(noisy_image,'Image corrupted
    by salt & pepper noise');//ShowImage() is used to
    show image, figure is command to view
    images in separate window.
31 // title('Image corrupted by Gaussian noise');//
    title() is used for providing a title to an
    image.
32 // disp(size(noise_image));
33 out_image=imadd(out_image,noisy_image);
34 end
35 out_image=out_image/No;
36 out_image=mat2gray(out_image);
37
38 figure,ShowImage(out_image,'Image Recoverd from the
    Noise');//ShowImage() is used to show image,
    figure is command to view images in separate
    window.
39 title('Image Recoverd from the Noise');//title()
    is used for providing a title to an image.
40 //Recoverd_Image=0.5*out_image.^0.15;//Gamma
    Transformation
41 //figure,ShowImage(Recoverd_Image,'Recoverd Image
    after Gamma Transormation');//ShowImage() is used
    to show image, figure is command to view images
    in separate window.
42 //title('Image Recoverd from the Noise');//title()
    is used for providing a title to an image.

```

43 **end**

---

check Appendix ?? for dependency:

Ex2\_6.tif

### Scilab code Exa 2.6 Image Subtraction for Enhancing differences

```
1 //Ex2_6
2 //Image Subtraction for Enhancing differences
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
  Woods
9
10 clc;
11 close;
12 clear;
13 xdel(winsid())//to close all currently open figure(s
  ).
14 gray=imread("Ex2_6.tif");
15 gray=imresize(gray,0.25,'bicubic'); //Resize the
  Image with bicubic Interpolation
16 figure,ShowImage(gray,'Gray Image');
17 title('Original Image');
18 [nr nc]=size(gray);
19 for i=1:8
20     c(:, :, i)=mtlb_logical(bitget(gray,9-i)); //
  Separate bit Planes from the Gray Scale Image
21
22 end
23 c(:, :, 8)=zeros(nr,nc); // Set Zeros to LSB
24 //c(:, :, 7)=zeros(nr,nc); // Set Zeros to LSB
```

```

25
26 for i=1:nr
27     for j=1:nc
28         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);
29     end
30 end
31 figure; ShowImage(mask, 'Modified Image');
32 title('Image Obtained by Setting Zeros to LSB ');
33 mask=uint8(mask); //Convert the Image to uint8 Data
    Type
34 Diff_image=imsubtract(gray,mask); //Subtract two
    Images
35 Diff_image=mat2gray(Diff_image);
36 figure; ShowImage(Diff_image, 'Modified Image');
37 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

```

1 //Ex2_7
2 // Image Multiplication for Shadding Correction.
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
    Woods
9
10 clc;
11 close;
12 clear;
13 xdel(winsid())//to close all currently open figure(s
    ).
14
15 /////////////////////////////////////////////////// Image Division
    //////////////////////////////////////
16 gray=imread(" Ex2_7.tif");
17 shade=imread(" Ex2_7_1.tif");
18 gray=im2double(imresize(gray,0.5,'bicubic')); //
    Resize the Image with Bicubic Interpolation
19 shade=im2double(imresize(shade,0.5,'bicubic')); //
    Resize the Image with Bicubic Interpolation
20 figure,ShowImage(gray,'Gray Image');
21 title('Original Image');
22 figure,ShowImage(shade,'Sahde Image');
23 title('Shading Pattern Image');
24 [nr nc]=size(gray);
25 Enhance_image=imdivide(gray,shade);
26 Enhance_image=mat2gray(Enhance_image);
27 figure,ShowImage(Enhance_image,'Enhance Image');
28 title('Enhance Image after Shading Correction');
29
30 /////////////////////////////////////////////////// Image Multiplication
    //////////////////////////////////////
31 gray=imread(" Ex2_7_2.tif");
32 mask=imread(" Ex2_7_3.tif");

```

```

33 gray=im2double(imresize(gray,0.5,'bicubic')); //
    Resize the Image with Bicubic Interpolation
34 mask=im2double(imresize(mask,0.5,'bicubic')); //
    Resize the Image with Bicubic Interpolation
35 figure,ShowImage(gray,'Gray Image');
36 title('Original Image');
37 figure,ShowImage(mask,'mask Image');
38 title('mask Pattern Image(ROI)');
39 [nr nc]=size(gray);
40 Enhance_image=immultiply(gray,mask);
41 Enhance_image=mat2gray(Enhance_image);
42 figure,ShowImage(Enhance_image,'Enhance Image');
43 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

```

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

```

```

9
10 clc;
11 close;
12 clear;
13 xdel(winsid()) //to close all currently open figure(s
    ).
14
15 ////////// Image Rotation
    //////////
16 gray1=imread("Ex2_12.tif");
17 gray1=im2double(gray1); //Convert the data type into
    double range
18 figure,ShowImage(gray1,'Gray Image');
19 title('Original Image');
20 gray2=imread("Ex2_12_1.tif");
21 gray2=im2double(gray2); //Convert the data type
    into double range
22 figure,ShowImage(gray2,'Gray Image');
23 title('Original Image');
24 gray3=imread("Ex2_12_2.tif");
25 gray3=im2double(gray3); //Convert the data type into
    double range
26 figure,ShowImage(gray3,'Gray Image');
27 title('Original Image');
28
29 y=variance(gray1); // calculate variance
30 disp('Variance of Image 1:')
31 disp(y);
32 y=variance(gray2); // calculate variance
33 disp('Variance of Image 2:')
34 disp(y);
35 y=variance(gray3); // calculate variance
36 disp('Variance of Image 3:')
37 disp(y);

```

---



# Chapter 3

## Intensity Transformation and Spatial Filtering

check Appendix ?? for dependency:

Ex3\_1.tif

**Scilab code Exa 3.1** Gamma Intensity transformation

```
1 //Ex3_1
2 // Gamma Intensity transformation
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
  Woods
9
10 clc;
11 close;
12 clear;
13 xdel(winsid())//to close all currently open figure(s
  ).
```

```

14 gray=imread("Ex3_1.tif");
15 figure,ShowImage(gray,'Gray Image');
16 title('Original Image','color','blue','fontsize',4);
17 [M,N]=size(gray);
18 c=1;
19 gamma=[0.6 0.4 0.3];
20 for i=1:length(gamma)
21     b=c.*(gray).^gamma(i); //Gamma transformation
22     b=mat2gray(b);
23     figure,ShowImage(b,'Gray Image');
24     title('Enhance Image after Gamma transformation',
25           'color','blue','fontsize',4);
25 end

```

---

check Appendix ?? for dependency:

Ex3\_2.tif

### Scilab code Exa 3.2 Illustration of Power Law Transformation

```

1 //Ex3_2
2 // Illustration of Power Law Transformation
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
9     Woods
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

```

15 gray=im2double(gray);
16 figure,ShowImage(gray,'Gray Image');
17 title('Original Image','color','blue','fontsize',4);
18 [M,N]=size(gray);
19 c=1;
20 gamma=[3 4 5];
21 for i=1:length(gamma)
22     b=c.*(gray).^gamma(i); //Gamma transformation
23     b=mat2gray(b);
24     figure,ShowImage(b,'Gray Image');
25     title('Enhance Image after Gamma transformation',
           'color','blue','fontsize',4);
26 end

```

---

check Appendix ?? for dependency:

Ex3\_3.tif

### Scilab code Exa 3.3 Intensity Level Slicing

```

1 //Ex3_3
2 // Intensity Level Slicing
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
   Woods
9
10 clc;
11 close;
12 clear;
13 xdel(winsid())//to close all currently open figure(s
   ).
14 gray=imread("Ex3_3.tif");

```

```

15 //gray=im2double(gray);
16 figure,ShowImage(gray,'Gray Image');
17 title('Original Image','color','blue','fontsize',4);
18 [M,N]=size(gray);
19 A=145;
20 B=245;
21 for i=1:M
22     for j=1:N
23         if(gray(i,j)>A & gray(i,j)<=B)
24             b(i,j)=255;
25             c(i,j)=255;
26         else
27             b(i,j)=0;
28             c(i,j)=gray(i,j);
29         end
30     end
31 end
32 figure,ShowImage(b,'Gray Image');
33 title('Image after Intensity Slicing transformation',
34       'color','blue','fontsize',4);
35 figure,ShowImage(c,'Gray Image');
36 title('Image after Intensity Slicing transformation(
37       Linear)','color','blue','fontsize',4);

```

---

### Scilab code Exa 3.5 A Simple Illustration of Histogram Equalization

```

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

```

## Woods

```
9
10
11
12 clc;
13 close;
14 clear;
15 xdel(winsid()) //to close all currently open figure(s
    ).
16 r=[0 1 2 3 4 5 6 7]; // Intensity
17 nk=[790 1023 850 656 329 245 122 81]; //Total No.
    of Pixels having Same Intensity
18 M=sum(nk);
19 probability_r=nk/M; // Probability calculation
20 for i=1:length(r)
21     sum_1=0;
22     for j=1:i
23         sum_1=sum_1+probability_r(j);
24     end
25     s(i)=max(r)*sum_1;
26 end
27 s=round(s); // Rounding Approach
28 disp(s);
29 [nr nc]=size(s);
30
31
32 for i=0:max(r)
33     [row col]=find(s==i);
34     len=length(row);
35     if(len>0)
36         sum_1=0;
37         for j=1:len
38             sum_1=sum_1+probability_r(row(j)); //
                Addition of Probability of same
                intensity after Equalization
39         end
40         Hist_equ(i+1)=sum_1;
41     else
```

```

42         Hist_equ(i+1)=0;
43     end
44 end
45 disp('Histogram Equalization:')
46 disp(Hist_equ);
47
48 figure,bar(r,probability_r,0.1);
49 title('Original Histogram','color','blue','fontsize'
50       ,4);
51 xlabel('Intensity');
52 ylabel('Probability of Same Intensity');
53
54 figure,bar(r,Hist_equ,0.1);
55 title('Equalized Histogram','color','blue','fontsize'
56       ,4);
57 xlabel('Intensity');
58 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

```

1 //Ex3-6
2 // Histogram Equalization
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1

```

```

6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
    Woods
9
10
11
12 clc;
13 close;
14 clear;
15 xdel(winsid())//to close all currently open figure(s
    ).
16 a=imread('Ex3_6.tif');
17 [P Q]=size(a);
18 [count cell]=imhist(a);
19 figure,ShowImage(a,'Original Image');
20 title('Original Image','color','blue','fontsize',4);
21
22 r=cell'; // Transpose of matrix
23 nk=round(count)'; // Transpose of matrix
24
25 //r=[0 1 2 3 4 5 6 7]; // Intensity
26 //nk=[790 1023 850 656 329 245 122 81]; //Total No.
    of Pixels having Same Intensity
27 M=sum(nk);
28 probablity_r=nk/M; // Probablity calculation
29 for i=1:length(r)
30     sum_1=0;
31     for j=1:i
32         sum_1=sum_1+probablity_r(j);
33     end
34     s(i)=max(r)*sum_1;
35 end
36 s=round(s); // Rounding Approach
37 disp(s);
38 [nr nc]=size(s);
39 temp=s'; // Transpose of matrix
40 for i=1:P // Intensity Replacement in Original

```

```

    Image
41     for j=1:Q
42         b(i,j)=temp(double(a(i,j))+1);
43     end
44 end
45 figure,ShowImage(b,'histogram Equalized Image');
46 title('histogram Equalized Image','color','blue','
    fontsize',4);
47
48 for i=0:max(r)
49     [row col]=find(s==i);
50     len=length(row);
51     if(len>0)
52         sum_1=0;
53         for j=1:len
54             sum_1=sum_1+probeblity_r(row(j)); //
                    Addition of Probability of same
                    intensity after Equqlization
55         end
56         Hist_equ(i+1)=sum_1;
57     else
58         Hist_equ(i+1)=0;
59     end
60 end
61 disp('Histogram Equalization:');
62 disp(Hist_equ);
63
64 figure,bar(r,Hist_equ,0.1);
65 title('Equalized Histogram','color','blue','fontsize
    ',4);
66 xlabel('Intensity');
67 ylabel('Probability of Same Intensity');

```

---

Scilab code Exa 3.8 Histogram Specification



```

1 //Ex3_8
2 // Histogram Specification
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
   Woods
9
10
11
12 clc;
13 close;
14 clear;
15 xdel(winsid())//to close all currently open figure(s
   ).
16 r=[0 1 2 3 4 5 6 7]; // Intensity
17 nk=[790 1023 850 656 329 245 122 81]; //Total No.
   of Pixels having Same Intensity
18 probability_Specified=[0.00 0.00 0.00 0.15 0.20 0.30
   0.20 0.15]; // Histogram Specification
19 M=sum(nk);
20 probability_r=nk/M; // Probablity calculation
21 for i=1:length(r)
22     sum_1=0;
23     sum_2=0;
24     for j=1:i
25         sum_1=sum_1+probability_r(j); //
   Histogram Equalization
26         sum_2=sum_2+probability_Specified(j); //
   Histogram Specification
27     end
28     s(i)=max(r)*sum_1;
29     G(i)=max(r)*sum_2;
30 end
31
32 s=round(s); // Rounding Approach

```

```

33 disp('Histogram Equalization:')
34 disp(s);
35 G=round(G); // Rounding Approach
36 disp('Histogram Specification G(Zq):')
37 disp(G);
38 [nr nc]=size(s);
39
40 for i=0:max(r)
41     [row col]=find(G(i+1)==s);
42     len=length(row);
43     if(len>0)
44         sum_1=0;
45         for j=1:len
46             sum_1=sum_1+probability_r(row(j));
47         end
48         Hist_Spe(i+1)=sum_1;
49     end
50     if(len==0)
51         if(G(i+1)==0)
52             Hist_Spe(i+1)=0;
53         else
54             Hist_Spe(i+1)=probability_r(G(i+1));
55         end
56 end
57 end
58 disp('Histogram After Matching:')
59 disp(Hist_Spe);
60
61 figure,bar(r,probability_r,0.1);
62 title('Original Histogram','color','blue','fontsize
    ',4);
63 xlabel('Intensity');
64 ylabel('Probability of Same Intensity');
65
66 figure,bar(r,probability_Specified,0.1);
67 title('Specified Histogram','color','blue','fontsize
    ',4);
68 xlabel('Intensity');

```

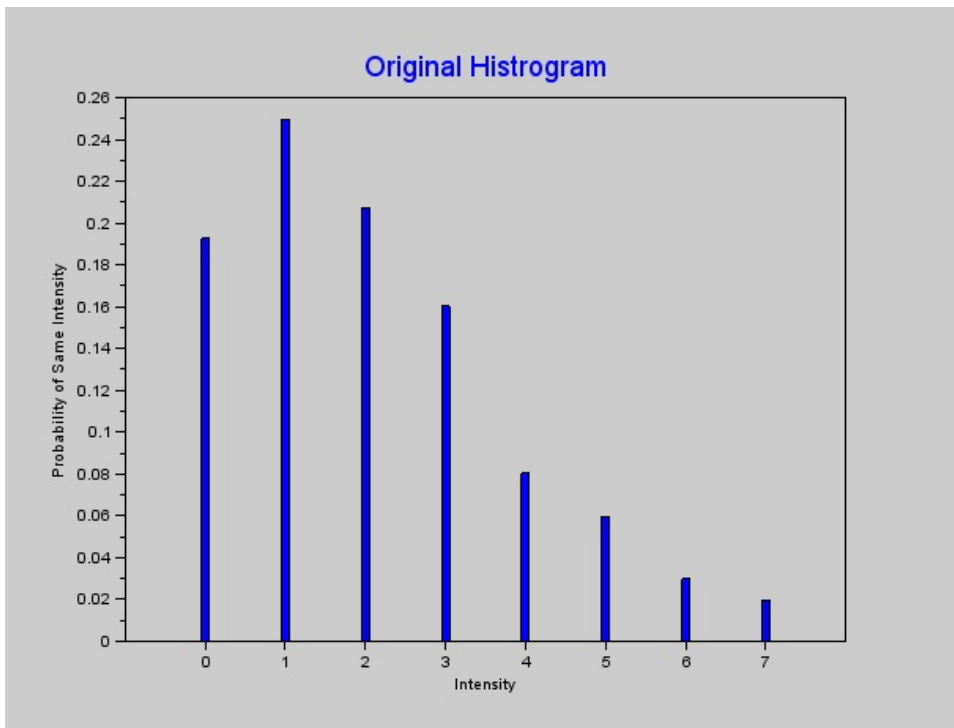


Figure 3.1: Histogram Specification

```

69 ylabel('Probability of Same Intensity');
70
71 figure, bar(r, Hist_Spe, 0.1);
72 title('Histogram matching', 'color', 'blue', 'fontsize',
73       ,4);
73 xlabel('Intensity');
74 ylabel('Probability of Same Intensity');

```

---

check Appendix ?? for dependency:

Ex3\_10.tif

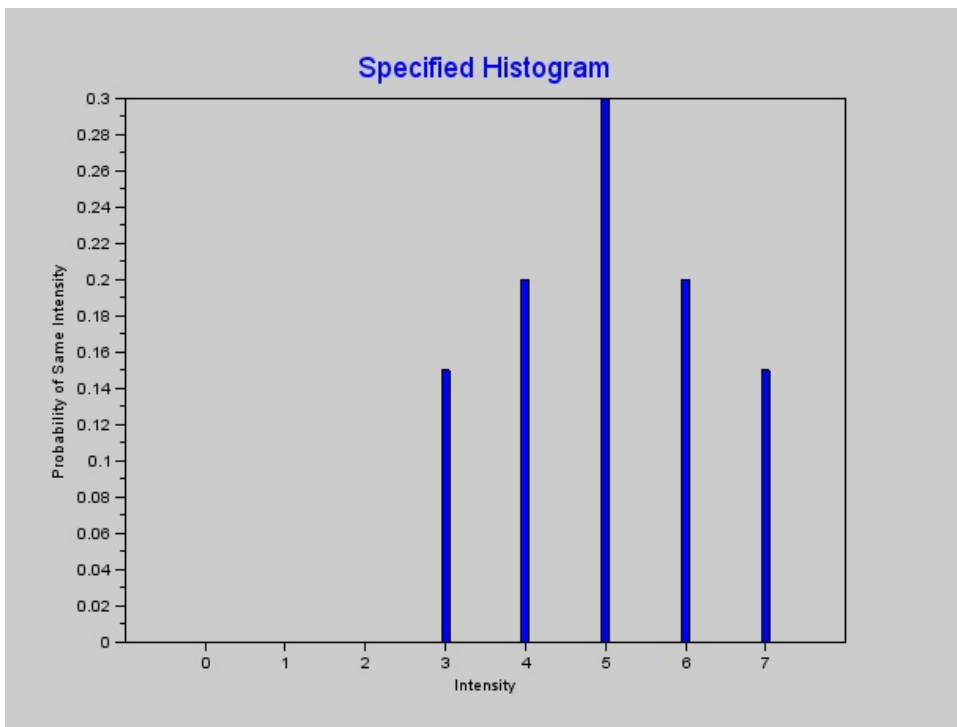


Figure 3.2: Histogram Specification

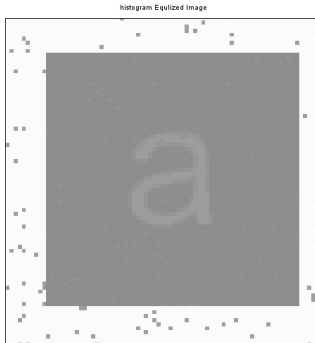


Figure 3.3: Local Histogram Equalization

**Scilab code Exa 3.10** Local Histogram Equalization

```
1 //Ex3_10
2 // Local Histogram Equalization
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
   Woods
9
10
11
12 clc;
13 close;
```

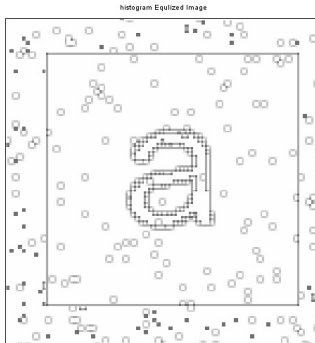


Figure 3.4: Local Histogram Equalization

```

14 clear;
15 xdel(winsid())//to close all currently open figure(s
    ).
16 p1=imread('Ex3_10.tif');
17 a=imcrop(p1,[175 178 155 160]);
18 //a=imresize(a,0.5,'bicubic');
19 [P Q]=size(a);
20
21 ////////////////////////////////// Global Histogram
    Equalization //////////////////////////////////
22 [count cell]=imhist(a);
23 figure,ShowImage(a,'Original Image');
24 title('Original Image','color','blue','fontsize',4);
25
26 r=cell'; // Transpose of matrix
27 nk=round(count)'; // Transpose of matrix
28 M=sum(nk);
29 probablity_r=nk/M; // Probablity calculation
30 for i=1:length(r)
31     sum_1=0;
32     for j=1:i
33         sum_1=sum_1+proablity_r(j);
34     end

```

```

35     s(i)=max(r)*sum_1;
36 end
37 s=round(s); // Rounding Approach
38 //disp(s);
39 [nr nc]=size(s);
40 temp=s'; // Transpose of matrix
41 for i=1:P // Intensity Replacement in Original
    Image
42     for j=1:Q
43         b(i,j)=temp(double(a(i,j))+1);
44     end
45 end
46 figure,ShowImage(b,'histogram Equalized Image');
47 title('Image Enhancement using Global Histogram
    Equalization ','color ','blue ','fontsize ',4);
48
49 ////////////////////////////////// Local Histogram
    Equalization //////////////////////////////////
50 mask=3;
51 for i=1+floor(mask/2):P-floor(mask/2)
52     for j=1+floor(mask/2):Q-floor(mask/2)
53         a1=a(i-floor(mask/2):1:i+floor(mask/2),j-
            floor(mask/2):1:j+floor(mask/2)) ;
54         [count cell]=imhist(a1);
55         r=cell'; // Transpose of matrix
56         nk=round(count)'; // Transpose of matrix
57         M=sum(nk);
58         probablity_r=nk/M; // Probablity
            calculation
59         for x=1:length(r)
60             sum_1=0;
61             for y=1:x
62                 sum_1=sum_1+probeblity_r(y);
63             end
64             s(x)=max(r)*sum_1;
65         end
66         s=round(s); // Rounding Approach
67         //disp(s);

```

```

68         [nr nc]=size(s);
69         temp=s';    // Transpose of matrix
70         b(i,j)=temp(double(a(i,j))+1);
71     end
72     disp(i);
73 end
74 figure,ShowImage(b,'histogram Equalized Image');
75 title('Image Enhancement using Local Histogram
        Equalization ','color ','blue ','fontsize ',4);

```

---

### Scilab code Exa 3.11 Computing Histogram Statistic

```

1 //Ex3_11
2 // Computing Histogram Statistic
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
    Woods
9
10
11
12 clc;
13 close;
14 clear;
15 xdel(winsid())//to close all currently open figure(s
    ).
16 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]);
17 L=max(a);
18 [P Q]=size(a);
19
20 ////////////////////////////////////// Global Histogram

```



```

    Equalization  //////////////////////////////////////
21 [count cell]=imhist(a);
22 //figure ,bar(cell(1:L+1),count(1:L+1),0.2);
23 //title('Histogram');
24 r=cell(1:L+1)'; // Transpose of matrix
25 nk=round(count(1:L+1))'; // Transpose of matrix
26 M=sum(nk);
27 probablity_r=nk/M; // Probablity calculation
28 sum_1=0;
29 for i=1:length(r)
30     sum_1=sum_1+(r(i)*proablity_r(i));
31 end
32 Mean=sum_1;
33 disp('Probablity:');
34 disp(proablity_r);
35 disp('Mean:');
36 disp(Mean);
37
38 Mean1=mean(double(a));
39 disp('Mean Calculated from (5*5)Image:');
40 disp(Mean1);

```

---

check Appendix ?? for dependency:

Ex3\_12.tif

### Scilab code Exa 3.12 Local Enhancement using Histogram Statistic

```

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

```

```

8 //book author: Rafael C. Gonzalez and Richard E.
  Woods
9
10
11
12 clc;
13 close;
14 clear;
15 xdel(winsid())//to close all currently open figure(s
  ).
16 a=imread('Ex3_12.tif');
17 //a=double(a);
18 [M N]=size(a);
19
20 //////////////////////////////////// Global Histogram
  Equalization //////////////////////////////////////
21 [count cell]=imhist(a); // Histogram Calculation
22 figure,ShowImage(a,'Original Image');
23 title('Original Image','color','blue','fontsize',4);
24
25 r=cell'; // Transpose of matrix
26 nk=round(count)'; // Transpose of matrix
27 P=sum(nk);
28 probablity_r=nk/P; // Probablity calculation
29 for i=1:length(r)
30     sum_1=0;
31     for j=1:i
32         sum_1=sum_1+probablity_r(j);
33     end
34     s(i)=max(r)*sum_1;
35 end
36 s=round(s); // Rounding Approach
37 disp(s);
38 [nr nc]=size(s);
39 temp=s'; // Transpose of matrix
40 for i=1:M // Intensity Replacement in Original
  Image
41     for j=1:N

```

```

42         b(i,j)=temp(double(a(i,j))+1);
43     end
44 end
45 figure,ShowImage(b,'histogram Equilized Image');
46 title('Image Enhancement using Global Histogram
Statistic ','color','blue','fontsize',4);
47
48
49 //////////////////////////////////////////////////////////////////// Image Enhancement using
Local Histogram Statistic
//////////////////////////////////////////////////////////////////
50 E=4;K0=0.4;K1=0.02;K2=0.4;
51 mask=3;
52 Mean_G=mean(double(a)); // Global Mean Value
53 Variance_G=variance(double(a)); // Global Variance
Value
54
55 for i=1+floor(mask/2):M-floor(mask/2)
56     for j=1+floor(mask/2):N-floor(mask/2)
57         a1=a(i-floor(mask/2):1:i+floor(mask/2),j-
floor(mask/2):1:j+floor(mask/2)) ;
58         Mean_L=mean(double(a1)); // Local Mean
Value
59         Variance_L=variance(double(a1)); // Local
Variance Value
60         if((Mean_L<=K0*Mean_G) & (K1*Variance_G<=
Variance_L) & (Variance_L<=K2*Variance_G)
)
61             g(i,j)=E*a(i,j);
62         else
63             g(i,j)=a(i,j);
64         end
65     end
66 end
67
68 figure,ShowImage(g,'Local Histogram Statistic');
69 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

```
1 //Ex3_13
2 // Image Smoothing
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
  Woods
9
10
11
12 clc;
13 close;
14 clear;
15 xdel(winsid())//to close all currently open figure(s
  ).
16 a=imread('Ex3_13.tif');
17 [M N]=size(a);
18 figure,ShowImage(a,'Original Image');
19 title('Original Image','color','blue','fontsize',4);
20
21 ////////////////////////////////////////////////// Smoothing with Mask Size
  (3*3) //////////////////////////////////////
22 F=fspecial('average',3);
23 Image=imfilter(a,F);
24 figure,ShowImage(Image,'Original Image');
25 title('Filtered Image with Mask Size(3*3)','color','
  blue','fontsize',4);
```

```

26
27 ////////////////////////////////////////////////// Smoothing with Mask Size
    (5*5) //////////////////////////////////////
28 F=fspecial('average',5);
29 Image=imfilter(a,F);
30 figure,ShowImage(Image,'Original Image');
31 title('Filtered Image with Mask Size(5*5)', 'color', '
    blue', 'fontsize',4);
32
33 ////////////////////////////////////////////////// Smoothing with Mask Size
    (5*5) //////////////////////////////////////
34 F=fspecial('average',9);
35 Image=imfilter(a,F);
36 figure,ShowImage(Image,'Original Image');
37 title('Filtered Image with Mask Size(9*9)', 'color', '
    blue', 'fontsize',4);
38
39 ////////////////////////////////////////////////// Smoothing with Mask Size
    (5*5) //////////////////////////////////////
40 F=fspecial('average',15);
41 Image=imfilter(a,F);
42 figure,ShowImage(Image,'Original Image');
43 title('Filtered Image with Mask Size(15*15)', 'color', '
    , 'blue', 'fontsize',4);
44
45 ////////////////////////////////////////////////// Smoothing with Mask Size
    (5*5) //////////////////////////////////////
46 F=fspecial('average',35);
47 Image=imfilter(a,F);
48 figure,ShowImage(Image,'Original Image');
49 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

```
1 //Ex3_14
2 // Median Filtering for Noise Reduction
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
   Woods
9
10
11
12 clc;
13 close;
14 clear;
15 xdel(winsid())//to close all currently open figure(s
   ).
16 a=imread('Ex3_14.tif');
17 [M N]=size(a);
18 figure,ShowImage(a,'Original Image');
19 title('Original Image','color','blue','fontsize',4);
20
21 //////////////////////////////// Averaging Filter with
   Mask Size (3*3) //////////////////////////////////
22 F=fspecial('average',3);
23 Image=imfilter(a,F);
24 figure,ShowImage(Image,'Original Image');
25 title('Filtered Image with Averaging Filter(3*3)',
   color','blue','fontsize',4);
26
27 //////////////////////////////// Median Filtering with
   Mask Size (5*5) //////////////////////////////////
28 Image=MedianFilter(a,[3 3]);
29 figure,ShowImage(Image,'Original Image');
30 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

```
1 //Ex3_15
2 // Image Sharpning using Laplacian
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
  Woods
9
10
11
12 clc;
13 close;
14 clear;
15 xdel(winsid())//to close all currently open figure(s
  ).
16 a=imread('Ex3_15.tif');
17 [M N]=size(a);
18 figure,ShowImage(a,'Original Image');
19 title('Original Image','color','blue','fontsize',4);
20
21 ////////////////////////////////// Laplacian Filtering
  //////////////////////////////////
22 F=fspecial('laplacian',0);
23 Image1=imfilter(a,F);
24 figure,ShowImage(Image1,'Original Image');
25 title('Filtered Image with Laplacian Mask','color','
  blue','fontsize',4);
```

```

26
27 ////////////////////////////////////////////////////////////////// Laplacian Filtering
   //////////////////////////////////////////////////////////////////
28 F=[1 1 1;1 -8 1;1 1 1];
29 Image2=imfilter(a,F);
30 figure,ShowImage(Image2,'Original Image');
31 title('Filtered Image with Laplacian Mask','color','
      blue','fontsize',4);
32
33 ////////////////////////////////////////////////////////////////// Laplacian Filtering
   //////////////////////////////////////////////////////////////////
34 b=a-(1*Image1);
35 figure,ShowImage(b,'Original Image');
36 title('Filtered Image with Laplacian Mask','color','
      blue','fontsize',4);
37
38
39 ////////////////////////////////////////////////////////////////// Laplacian Filtering
   //////////////////////////////////////////////////////////////////
40 b=a-(1*Image2);
41 figure,ShowImage(b,'Original Image');
42 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

```

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

```



```

6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
    Woods
9
10
11
12 clc;
13 close;
14 clear;
15 xdel(winsid())//to close all currently open figure(s
    ).
16 a=imread('Ex3_16.tif');
17 [M N]=size(a);
18 figure,ShowImage(a,'Original Image');
19 title('Original Image','color','blue','fontsize',4);
20
21 //////////////////////////////// Laplacian Filtering
    ////////////////////////////////
22 F=fspecial('gaussian',5,3);
23 Image1=imfilter(a,F);
24 figure,ShowImage(Image1,'Original Image');
25 title('Filtered Image with gaussian Filter(3*3)',
    color','blue','fontsize',4);
26
27 Unsharp_Mask=a-Image1;
28 figure,ShowImage(Unsharp_Mask,'Original Image');
29 title('Unsharp Mask Image','color','blue','fontsize'
    ,4);
30
31 //////////////////////////////// Un-Sharp Filtering
    ////////////////////////////////
32 k=1;
33 Unsharp=a+(k.*Image1);
34 figure,ShowImage(Unsharp,'Original Image');
35 title('Unsharp Filtered Image','color','blue',
    fontsize',4);
36

```

```

37 ////////////////////////////////////////////////// High-Boost Filtering
   //////////////////////////////////////////////////
38 k=4.5;
39 High_Boost=a+(k.*Image1);
40 figure,ShowImage(High_Boost,'Original Image');
41 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

```

1 //Ex3_17
2 // Use of gradient for Edge Enhancement
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
   Woods
9
10
11
12 clc;
13 close;
14 clear;
15 xdel(winsid())//to close all currently open figure(s
   ).
16 a=rgb2gray(imread('Ex3_17.png'));
17 [M N]=size(a);
18 figure,ShowImage(a,'Original Image');
19 title('Original Image','color','blue','fontsize',4);
20

```

```
21 ////////////////////////////////////////////////// Laplacian Filtering
    //////////////////////////////////////
22 F=fspecial('sobel'); // Sobel Mask
23 Image1=imfilter(a,F);
24 figure,ShowImage(Image1,'Original Image');
25 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

```
1 //Ex4_1
2 // Obtaining the Fourier Transform of a Simple
   Function
3 //Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
   Woods
9
10
11 clc; //to clear command window.
12 clear; //to kill previously defined variables.
13 xdel(winsid());//to close all currently open figure(
   s).
14
15 f=-5:0.01:5;
16 [nr nc]=size(f);
17 signal=ones(nr,nc);
18 A=1;
```

```

19 W=length(f);
20 for i=1:W
21     if(f(i)==0)
22         mag(i)=A;
23     else
24         mag(i)=A*W*(sin(%pi*f(i)*W)/((%pi*f(i)*W)+
                %eps));
25     end
26
27 end
28
29 figure,mtlb_axis([-6 6 0 2]);
30 bar(f,signal,0.1);
31 xlabel('Time Index','color','blue','fontsize',2);
32 ylabel('Amplitude','color','blue','fontsize',2);
33 title('Rectangle Function','color','blue','fontsize',
        ,4);
34
35
36 figure,//mtlb_axis([-15 15 0 2]);
37 plot(f,mag);
38 xlabel('Frequency','color','blue','fontsize',2);
39 ylabel('Amplitude','color','blue','fontsize',2);
40 title('Frequency Spectrum Plot','color','blue','
        fontsize',4);
41
42
43 figure,//mtlb_axis([-15 15 0 2]);
44 plot(f,abs(mag));
45 xlabel('Frequency','color','blue','fontsize',2);
46 ylabel('Amplitude','color','blue','fontsize',2);
47 title('Frequency Spectrum Plot','color','blue','
        fontsize',4);

```

---

Scilab code Exa 4.4 The Mechanics of Computing the DFT

```

1 //Ex4_4
2 // The Mechanics of Computing the DFT
3 //Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
    Woods
9
10 clc;
11 close;
12 clear;
13 xdel(winsid())//to close all currently open figure(s
    ).
14 a=[1 2 4 4];
15 //b=fft2(a);
16 disp('Original Signal:')
17 disp(a);
18 M=length(a);
19 for i=1:M
20     b(i)=0;
21     for j=1:M
22         b(i)=b(i)+(a(j)*exp((-i*2*pi*(i-1)*(j-1)/M)));
23     end
24 end
25 disp('DFT of Signal:')
26 disp(b);
27
28 for i=1:M
29     d(i)=0;
30     for j=1:M
31         d(i)=d(i)+((b(j)*exp((i*2*pi*(i-1)*(j-1)/M)))/
            M);
32     end
33 end
34 disp('IDFT:')
35 disp(abs(d));

```

---

check Appendix ?? for dependency:

Ex4\_7.tif

### Scilab code Exa 4.7 Illustration of Aliasing in Resampled Images

```
1 //Ex4_7
2 // Illustration of Aliasing in Resampled Images
3 //Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
  Woods
9
10
11 clc;
12 close;
13 clear;
14 xdel(winsid())//to close all currently open figure(s
  ).
15 a=imread("Ex4_7.tif");
16 figure,ShowImage(a,'Gray Image');
17 title('Original Image [1025 1025]');
18 // [M,N]=size(a);
19 b=imresize(a,0.5,'nearest');
20 [M,N]=size(b);
21 d=[];
22 f=[]
23 for i=1:N
24     temp=b(:,i);
25     d=[d temp temp];
26 end
27 for i=1:M
```

```

28     temp=d(i,:);
29     f=[f;temp;temp];
30 end
31 figure,ShowImage(f,'Gray Image');
32 title('Resize Image with Pixels Replication','color',
        'blue','fontsize',4);
33
34
35 //////////////////////////////////////////////////////////////////// Averaging Approach to
    Reduce Jaggies Effect ////////////////////////////////////////////////////////////////////
36 filt=fspecial('average',3);
37 a_filter=imfilter(a,filt);
38 b=imresize(a_filter,0.5,'nearest');
39 //figure,ShowImage(b,'Gray Image');
40 //title('Resize Image with nearest Interpolation');
41 [M,N]=size(b);
42 d=[];
43 f=[]
44 for i=1:N
45     temp=b(:,i);
46     d=[d temp temp];
47 end
48 for i=1:M
49     temp=d(i,:);
50     f=[f;temp;temp];
51 end
52 figure,ShowImage(f,'Gray Image');
53 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



```

1 //Ex4_8
2 // Illustration of Jaggies in Image Shrinking
3 //Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
    Woods
9
10
11 clc;
12 close;
13 clear;
14 xdel(winsid())//to close all currently open figure(s
    ).
15 a=imread("Ex4_8.tif");
16 figure,ShowImage(a,'Gray Image');
17 title('Original Image [1024 1024]','color','blue','
    fontsize',4);
18 // [M,N]=size(a);
19 b=imresize(a,[256 256],'bicubic');
20 //figure,ShowImage(b,'Gray Image');
21 //title('Resize Image [256 256] with Bicubic
    Interpolation');
22 [M,N]=size(b);
23 d=[];
24 f=[]
25 for i=1:N
26     temp=b(:,i);
27     d=[d temp temp temp temp];
28 end
29 for i=1:M
30     temp=d(i,:);
31     f=[f;temp;temp;temp;temp];
32 end
33 figure,ShowImage(f,'Gray Image');
34 title('Resize Image [1024 1024] with Pixels

```

```

    Replication ', 'color ', 'blue ', 'fontsize ',4);
35
36
37 //////////////////////////////////////////////////////////////////// Averaging Approach to
    Reduce Jaggies Effect ////////////////////////////////////////////////////////////////////
38 filt=fspecial('average',5);
39 a_filter=imfilter(a,filt);
40 b=imresize(a_filter,[256 256],'bicubic');
41 //figure,ShowImage(b,'Gray Image');
42 //title('Resize Image [256 256] with Bicubic
    Interpolation ');
43 [M,N]=size(b);
44 d=[];
45 f=[]
46 for i=1:N
47     temp=b(:,i);
48     d=[d temp temp temp temp];
49 end
50 for i=1:M
51     temp=d(i,:);
52     f=[f;temp;temp;temp;temp];
53 end
54 figure,ShowImage(f,'Gray Image');
55 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

```

1 //Ex4_9
2 // Illustration of Jaggies in Image Zooming
3 //Version : Scilab 5.4.1

```

```

4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
    Woods
9
10
11 clc;
12 close;
13 clear;
14 xdel(winsid())//to close all currently open figure(s
    ).
15 a=imread("Ex4_9.tif");
16 a=imcrop(a,[323 377 256 256]);
17 //figure,ShowImage(a,'Gray Image');
18 //title('Original Image [1025 1025]');
19 b=imresize(a,[256 256],'bicubic');
20 [M,N]=size(b);
21 d=[];
22 f=[]
23 for i=1:N
24     temp=b(:,i);
25     d=[d temp temp temp temp];
26 end
27 for i=1:M
28     temp=d(i,:);
29     f=[f;temp;temp;temp;temp];
30 end
31 figure,ShowImage(f,'Gray Image');
32 title('Resize Image [1024 1024] with Pixels
    Replication','color','blue','fontsize',4);
33
34
35 ////////////////////////////////////// Bi-linear
    Interpolation //////////////////////////////////////
36
37 f=imresize(a,[1024 1024],'bilinear');

```

```
38 figure, ShowImage(f, 'Gray Image');
39 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

```
1 //Ex4_13
2 //The 2-D Fourier Spectrum of a Simple Function
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
   Woods
9
10 clc;
11 close;
12 clear;
13 xdel(winsid())//to close all currently open figure(s
   ).
14 a=imread("Ex4_13_1.tif");
15 a=imresize(a,0.5);
16 //gray=rgb2gray(a);
17 gray=im2double(a);
```

```

18
19 figure,ShowImage(gray,'Gray Image');
20 title('Original Image','color','blue','fontsize',4);
21 [M,N]=size(gray);
22
23 h1=fft2(gray);//fft2() is used to find 2-Dimensional
    Fast Fourier Transform of an matrix
24 figure,ShowImage(mat2gray(abs(h1)),'Frequency
    spectrum');
25 title('Frequency spectrum','color','blue','fontsize',
    4);
26
27 in=fftshift(h1);//fftshift() is used to rearrange
    the fft output, moving the zero frequency to the
    center of the spectrum.
28 figure,ShowImage(mat2gray(abs(in)),'Frequency
    spectrum');
29 title('Centred Frequency spectrum','color','blue','
    fontsize',4);
30
31 i=log(1+abs(in));
32 inm=mat2gray(i)
33 figure,ShowImage(inm,'Frequency Spectrum');//
    ShowColorImage() is used to show color image,
    figure is command to view images in separate
    window.
34 title('Frequency Spectrum','color','blue','fontsize',
    4);//title() is used for providing a title to
    an image.
35
36 ////////////////////////////////////// Effect of Translation
    //////////////////////////////////////
37 a=imread("Ex4_13_2.png");
38 gray=rgb2gray(a);
39 gray=im2double(gray);
40 figure,ShowImage(gray,'Gray Image');
41 title('Original Image','color','blue','fontsize',4);
42 [M,N]=size(gray);

```

```

43 h2=fft2(gray); //fft2() is used to find 2-Dimensional
    Fast Fourier Transform of an matrix
44 i=log(1+abs(h2));
45 in=fftshift(i); //fftshift() is used to rearrange the
    fft output, moving the zero frequency to the
    center of the spectrum.
46 inm=mat2gray(in)
47 figure,ShowImage(inm,'Frequency Spectrum');//
    ShowColorImage() is used to show color image,
    figure is command to view images in separate
    window.
48 title('Frequency Spectrum','color','blue','fontsize'
    ,4); //title() is used for providing a title to
    an image.
49
50 ////////////////////////////////////// Effect of Rotation
    //////////////////////////////////////
51 a=imread("Ex4_13_3.png");
52 gray=rgb2gray(a);
53 gray=im2double(gray);
54 figure,ShowImage(gray,'Gray Image');
55 title('Original Image','color','blue','fontsize',4);
56 [M,N]=size(gray);
57 h3=fft2(gray); //fft2() is used to find 2-Dimensional
    Fast Fourier Transform of an matrix
58 i=log(1+abs(h3));
59 in=fftshift(i); //fftshift() is used to rearrange the
    fft output, moving the zero frequency to the
    center of the spectrum.
60 inm=mat2gray(in)
61 figure,ShowImage(inm,'Frequency Spectrum');//
    ShowColorImage() is used to show color image,
    figure is command to view images in separate
    window.
62 title('Frequency Spectrum','color','blue','fontsize'
    ,4); //title() is used for providing a title to
    an image.
63

```

```

64
65 //////////////////////////////////////////////////////////////////// Phase Spectrum
   ////////////////////////////////////////////////////////////////////
66 phase=atand(imag(h1),real(h1));
67 phase_1=mat2gray(phase);
68 figure,ShowImage(phase_1,'phase Spectrum');
69 title('phase Spectrum','color','blue','fontsize',4);
70
71 phase=atand(imag(h2),real(h2));
72 phase_1=mat2gray(phase);
73 figure,ShowImage(phase_1,'phase Spectrum');
74 title('phase Spectrum of Translated Object','color',
       'blue','fontsize',4);
75
76 phase=atand(imag(h3),real(h3));
77 phase_1=mat2gray(phase);
78 figure,ShowImage(phase_1,'phase Spectrum');
79 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

```

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

```

```

7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
  Woods
9
10 clc;
11 close;
12 clear;
13 xdel(winsid())//to close all currently open figure(s
  ).
14 a=imread("Ex4_14.tif");
15 mask=imread("Ex4_14_2.tif");
16 mask=im2double(imresize(mask,[512 512]));
17 //gray=rgb2gray(a);
18 gray=im2double(a);
19
20 figure,ShowImage(gray,'Gray Image');
21 title('Original Image','color','blue','fontsize',4);
22 [M,N]=size(gray);
23
24 h=fft2(gray);//fft2() is used to find 2-Dimensional
  Fast Fourier Transform of an matrix
25 in=fftshift(h);//fftshift() is used to rearrange the
  fft output, moving the zero frequency to the
  center of the spectrum.
26 i=log(1+abs(in));
27
28 inm=mat2gray(i);
29 //figure,ShowImage(inm,'Center Frequency Spectrum');
30 //title('Center Frequency Spectrum');
31
32 phase=atand(imag(h),real(h));
33 phase_1=mat2gray(phase);
34 figure,ShowImage(phase_1,'phase Spectrum');
35 title('phase Spectrum','color','blue','fontsize',4);
36
37 phase_mask=atand(imag(fft2(mask)),real(fft2(mask)));
38 phase_2=mat2gray(phase_mask);
39 //figure,ShowImage(phase_2,'phase Spectrum');

```



```

40 //title('phase Spectrum 2');
41
42 Image_recoverd=real(ifft(phase));
43 Image_recoverd=mat2gray(Image_recoverd)
44 //figure,ShowImage(Image_recoverd,'recoverd Image');
45 //title('recoverd Image by only Phase');
46
47
48 Image_recoverd=fftshift(real(ifft(abs(h))));
49 Image_recoverd=mat2gray(Image_recoverd)
50 figure,ShowImage(Image_recoverd,'recoverd Image');
51 title('recoverd Image by only Spectrum','color','
    blue','fontsize',4);
52
53
54 Image_recoverd=real(ifft(fftshift((mask.*in)+phase))
    );
55 Image_recoverd=(mat2gray(Image_recoverd));
56 figure,ShowImage(Image_recoverd,'recoverd Image');
57 title('recoverd Image by Magnitude in mask and Phase
    ','color','blue','fontsize',4);
58
59
60 Image_recoverd=real(ifft(fftshift(in)+abs(fft2(mask)
    )));
61 Image_recoverd=(mat2gray(Image_recoverd));
62 figure,ShowImage(Image_recoverd,'recoverd Image');
63 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

```

1 //Ex4_15
2 // Obtaining a Frequency domain Filter from a Small
   Spatial Mask
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
   Woods
9
10 clc;
11 close;
12 clear;
13 xdel(winsid())//to close all currently open figure(s
   ).
14
15 function [H]=sobelfilter(mask_pad)//lowpassfilter is
   used to filter an image.
16     x=fft2(mask_pad);
17     [nr nc]=size(mask_pad);
18     x_real=real(x);
19     x_imag=imag(x);
20     z=zeros(nr,nc)+%i*x_imag
21     H=fftshift(z);
22     y=log(1+abs(H));
23     y=mat2gray(y)
24     figure,ShowImage(y,'Frequency Spectrum');
25     title('Frequency Spectrum','color','blue','
   fontsize',4);
26 endfunction
27
28
29
30
31 a=imread("Ex4_15.tif");
32 gray=im2double(a);
33 mask=[-1 0 1;-2 0 2;-1 0 1];

```

```

34 figure,ShowImage(gray,'Gray Image');
35 title('Original Image','color','blue','fontsize',4);
36 [M,N]=size(gray);
37 gray_pad=zeros(M+2,N+2); // Zero Padding
38 mask_pad=zeros(M+2,N+2); // Zero Padding
39 gray_pad(1:M,1:N)=gray(1:$,1:$);
40 mask_pad(1:3,1:3)=mask(1:$,1:$);
41
42 h=fft2(gray_pad);//fft2() is used to find 2-
    Dimensional Fast Fourier Transform of an matrix
43
44 in=fftshift(h);//fftshift() is used to rearrange the
    fft output, moving the zero frequency to the
    center of the spectrum.
45 i=log(1+abs(in));
46 inm=mat2gray(i)
47 figure,ShowImage(inm,'Frequency Spectrum');
48 title('Frequency Spectrum','color','blue','fontsize',
    ,4);
49
50 ////////////////////////////////////// Filtering Domain
    Filtering //////////////////////////////////////
51 filt=sobelfilter(mask_pad); // Function which
    generate Filter Mask
52 n=filt.*in;//Multiply the Original Spectrum with the
    Filter Mask.
53 n=fftshift(n);
54 Image_filter=real(iffn(n));
55 Image_filter=mat2gray(Image_filter)
56 figure,ShowImage(Image_filter,'Filtered Image');
57 title('Filtered Image in Frequency Domain','color','
    blue','fontsize',4);
58
59 ////////////////////////////////////// Spatial Domain
    Filtering //////////////////////////////////////
60
61 imf = imfilter(a,mask);
62 //imf=1*(imf.^1.2);

```

```

63 [r c]=find(imf==0 | imf<=110);
64     for i=1:length(r)
65         imf(r(i),c(i)) = 125;
66     end
67 figure,ShowImage(imf,'Filtered Image');
68 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

```

1 //Ex4_16
2 //Image Smoothing Using an ILPF
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
    Woods
9
10 clc;
11 close;
12 clear;
13 xdel(winsid())//to close all currently open figure(s
    ).
14
15 function [H]=lowpassfilter(type1,M,N,D0,n)//
    lowpassfilter is used to filter an image .
16     u=0:(M-1);
17     v=0:(N-1);
18     idx=find(u>M/2);
19     u(idx)=u(idx)-M;

```

```

20     idy=find(v>N/2);
21     v(idy)=v(idy)-N;
22     [U,V]=meshgrid(v,u);
23     D=sqrt(U.^2+V.^2);
24     select type1
25
26     case 'ideal' then
27         H=double(D<=D0);
28     else
29         disp('Unknown filter type. ')
30     end
31 endfunction
32
33
34 /////////////////////////////////////////////////////////////////// Main Programm
35 ///////////////////////////////////////////////////////////////////
35 a=imread("Ex4_16.tif");
36 //gray=rgb2gray(a);
37 gray=im2double(a);
38
39 figure,ShowImage(gray,'Gray Image');
40 title('Original Image');
41 [M,N]=size(gray);
42
43 h=fft2(gray);//fft2() is used to find 2-Dimensional
44     Fast Fourier Transform of an matrix
45 i=log(1+abs(h));
46 in=fftshift(i);//fftshift() is used to rearrange the
47     fft output, moving the zero frequency to the
48     center of the spectrum.
49
50 inm=mat2gray(in)
51 figure,ShowImage(inm,'Frequency Spectrum');
52 title('Frequency Spectrum','color','blue','fontsize',
53     ,4);
54
55 /////////////////////////////////////////////////////////////////// Filtering With Cut-off
56     Frequency 10 ///////////////////////////////////////////////////////////////////
57 filt=lowpassfilter('ideal',M,N,10); // Function

```

```

        which generate Filter Mask Corresponding to Low
        Frequency
52 //filt_shift=fftshift(filt);
53 //figure,ShowImage(filt_shift,'Filter Mask');
54 //title('Filter Mask to Specific Cut-Off Frequency')
    ;
55 n=filt.*h;//Multiply the Original Spectrum with the
    Filter Mask.
56 Image_filter=real(iffn(n));
57 Image_filter=mat2gray(Image_filter)
58 figure,ShowImage(Image_filter,'Filtered Image');
59 title('Filtered Image with Cut-Off Frequency 10','
    color','blue','fontsize',4);
60
61
62 ////////////////////////////////////////////////// Filtering With Cut-off
    Frequency 30 //////////////////////////////////
63 filt=lowpassfilter('ideal',M,N,30); // Function
    which generate Filter Mask Corresponding to Low
    Frequency
64 //filt_shift=fftshift(filt);
65 //figure,ShowImage(filt_shift,'Filter Mask');
66 //title('Filter Mask to Specific Cut-Off Frequency')
    ;
67 n=filt.*h;//Multiply the Original Spectrum with the
    Filter Mask.
68 Image_filter=real(iffn(n));
69 Image_filter=mat2gray(Image_filter)
70 figure,ShowImage(Image_filter,'Filtered Image');
71 title('Filtered Image with Cut-Off Frequency 30','
    color','blue','fontsize',4);
72
73
74 ////////////////////////////////////////////////// Filtering With Cut-off
    Frequency 60 //////////////////////////////////
75 filt=lowpassfilter('ideal',M,N,60); // Function
    which generate Filter Mask Corresponding to Low
    Frequency

```

```

76 //filt_shift=fftshift(filt);
77 //figure,ShowImage(filt_shift,'Filter Mask');
78 //title('Filter Mask to Specific Cut-Off Frequency')
   ;
79 n=filt.*h;//Multiply the Original Spectrum with the
   Filter Mask.
80 Image_filter=real(iffn(n));
81 Image_filter=mat2gray(Image_filter)
82 figure,ShowImage(Image_filter,'Filtered Image');
83 title('Filtered Image with Cut-Off Frequency 60','
   color','blue','fontsize',4);
84
85
86 ////////////////////////////////////////////////// Filtering With Cut-off
   Frequency 160 //////////////////////////////////
87 filt=lowpassfilter('ideal',M,N,160); // Function
   which generate Filter Mask Corresponding to Low
   Frequency
88 //filt_shift=fftshift(filt);
89 //figure,ShowImage(filt_shift,'Filter Mask');
90 //title('Filter Mask to Specific Cut-Off Frequency')
   ;
91 n=filt.*h;//Multiply the Original Spectrum with the
   Filter Mask.
92 Image_filter=real(iffn(n));
93 Image_filter=mat2gray(Image_filter)
94 figure,ShowImage(Image_filter,'Filtered Image');
95 title('Filtered Image with Cut-Off Frequency 160','
   color','blue','fontsize',4);
96
97
98 ////////////////////////////////////////////////// Filtering With Cut-off
   Frequency 460 //////////////////////////////////
99 filt=lowpassfilter('ideal',M,N,460); // Function
   which generate Filter Mask Corresponding to Low
   Frequency
100 //filt_shift=fftshift(filt);
101 //figure,ShowImage(filt_shift,'Filter Mask');

```

## Filtered Image with Cut-Off Frequency 60

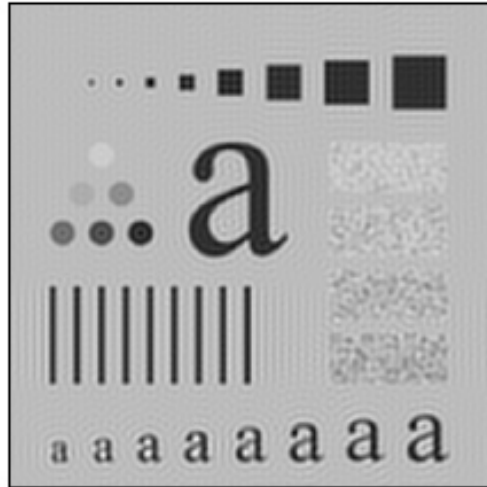


Figure 4.1: Image Smoothing using an ILPF

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

---

check Appendix ?? for dependency:

Ex4\_17.tif



Filtered Image with Cut-Off Frequency 460

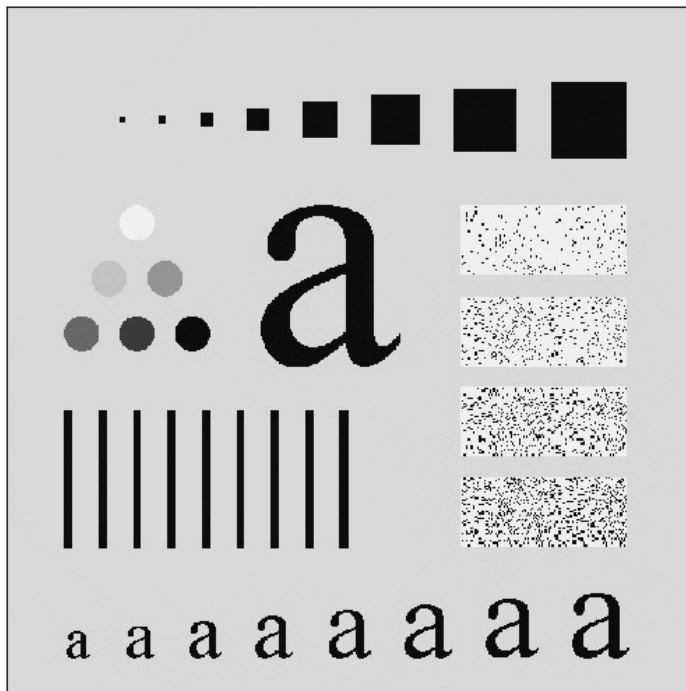


Figure 4.2: Image Smoothing using an ILPF

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

```
1 //Ex4_17
2 // Image Smoothing with a Butterworth LowPass Filter
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
   Woods
9
10 clc;
11 close;
12 clear;
13 xdel(winsid())//to close all currently open figure(s
   ).
14
15 function [H]=lowpassfilter(type1,M,N,D0,n)//
   lowpassfilter is used to filter an image .
16     u=0:(M-1);
17     v=0:(N-1);
18     idx=find(u>M/2);
19     u(idx)=u(idx)-M;
20     idy=find(v>N/2);
21     v(idy)=v(idy)-N;
22     [U,V]=meshgrid(v,u);
23     D=sqrt(U.^2+V.^2);
24     select type1
25
26     case 'butterworth' then
27         if argn(2)==4 then
28             n=1;
29         end
```

```

30         H = ones(M,N)/(1+(D./D0).^(2*n));
31
32     else
33         disp('Unknown filter type. ')
34     end
35 endfunction
36
37
38
39 //////////////////////////////////////////////////// Main Programm
40 ////////////////////////////////////////////////////
41 a=imread('Ex4_17.tif');
42 //gray=rgb2gray(a);
43 gray=im2double(a);
44 figure,ShowImage(gray,'Gray Image');
45 title('Original Image','color','blue','fontsize',4);
46 [M,N]=size(gray);
47
48 h=fft2(gray); //fft2() is used to find 2-Dimensional
49 //Fast Fourier Transform of an matrix
50 i=log(1+abs(h));
51 in=fftshift(i); //fftshift() is used to rearrange the
52 //fft output, moving the zero frequency to the
53 //center of the spectrum.
54 inm=mat2gray(in)
55 figure,ShowImage(inm,'Frequency Spectrum');
56 title('Frequency Spectrum','color','blue','fontsize',
57 //4);
58
59 //////////////////////////////////////////////////// Filtering With Cut-off
60 //Frequency 10 ////////////////////////////////////////////////////
61 filt=lowpassfilter('butterworth',M,N,10); //
62 //Function which generate Filter Mask Corresponding
63 //to Low Frequency
64 //filt_shift=fftshift(filt);
65 //figure,ShowImage(filt_shift,'Filter Mask');
66 //title('Filter Mask to Specific Cut-Off Frequency')

```

```

;
60 n=filt.*h;//Multiply the Original Spectrum with the
    Filter Mask.
61 Image_filter=real(iffn(n));
62 Image_filter=mat2gray(Image_filter)
63 figure,ShowImage(Image_filter,'Filtered Image');
64 title('Filtered Image with Cut-Off Frequency 10','
    color','blue','fontsize',4);
65
66
67 ////////////////////////////////////////////////// Filtering With Cut-off
    Frequency 30 //////////////////////////////////////////////////
68 filt=lowpassfilter('butterworth',M,N,30); //
    Function which generate Filter Mask Corresponding
    to Low Frequency
69 //filt_shift=fftshift(filt);
70 //figure,ShowImage(filt_shift,'Filter Mask');
71 //title('Filter Mask to Specific Cut-Off Frequency')
    ;
72 n=filt.*h;//Multiply the Original Spectrum with the
    Filter Mask.
73 Image_filter=real(iffn(n));
74 Image_filter=mat2gray(Image_filter)
75 figure,ShowImage(Image_filter,'Filtered Image');
76 title('Filtered Image with Cut-Off Frequency 30','
    color','blue','fontsize',4);
77
78
79 ////////////////////////////////////////////////// Filtering With Cut-off
    Frequency 60 //////////////////////////////////////////////////
80 filt=lowpassfilter('butterworth',M,N,60); //
    Function which generate Filter Mask Corresponding
    to Low Frequency
81 //filt_shift=fftshift(filt);
82 //figure,ShowImage(filt_shift,'Filter Mask');
83 //title('Filter Mask to Specific Cut-Off Frequency')
    ;
84 n=filt.*h;//Multiply the Original Spectrum with the

```

```

    Filter Mask.
85 Image_filter=real(iffn(n));
86 Image_filter=mat2gray(Image_filter)
87 figure,ShowImage(Image_filter,'Filtered Image');
88 title('Filtered Image with Cut-Off Frequency 60','
    color','blue','fontsize',4);
89
90
91 ////////////////////////////////////////////////// Filtering With Cut-off
    Frequency 160 //////////////////////////////////////
92 filt=lowpassfilter('butterworth',M,N,160); //
    Function which generate Filter Mask Corresponding
    to Low Frequency
93 //filt_shift=fftshift(filt);
94 //figure,ShowImage(filt_shift,'Filter Mask');
95 //title('Filter Mask to Specific Cut-Off Frequency')
    ;
96 n=filt.*h;//Multiply the Original Spectrum with the
    Filter Mask.
97 Image_filter=real(iffn(n));
98 Image_filter=mat2gray(Image_filter)
99 figure,ShowImage(Image_filter,'Filtered Image');
100 title('Filtered Image with Cut-Off Frequency 160','
    color','blue','fontsize',4);
101
102
103 ////////////////////////////////////////////////// Filtering With Cut-off
    Frequency 460 //////////////////////////////////////
104 filt=lowpassfilter('butterworth',M,N,460); //
    Function which generate Filter Mask Corresponding
    to Low Frequency
105 //filt_shift=fftshift(filt);
106 //figure,ShowImage(filt_shift,'Filter Mask');
107 //title('Filter Mask to Specific Cut-Off Frequency')
    ;
108 n=filt.*h;//Multiply the Original Spectrum with the
    Filter Mask.
109 Image_filter=real(iffn(n));

```

Filtered Image with Cut-Off Frequency 30

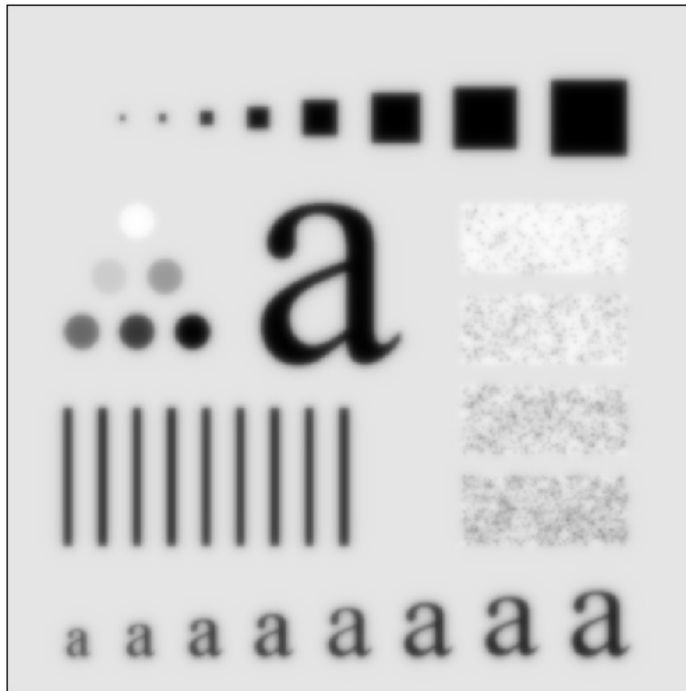


Figure 4.3: Image Smoothing with a Butterworth Lowpass Filter

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

---

check Appendix ?? for dependency:

Ex4\_18.tif

Filtered Image with Cut-Off Frequency 160

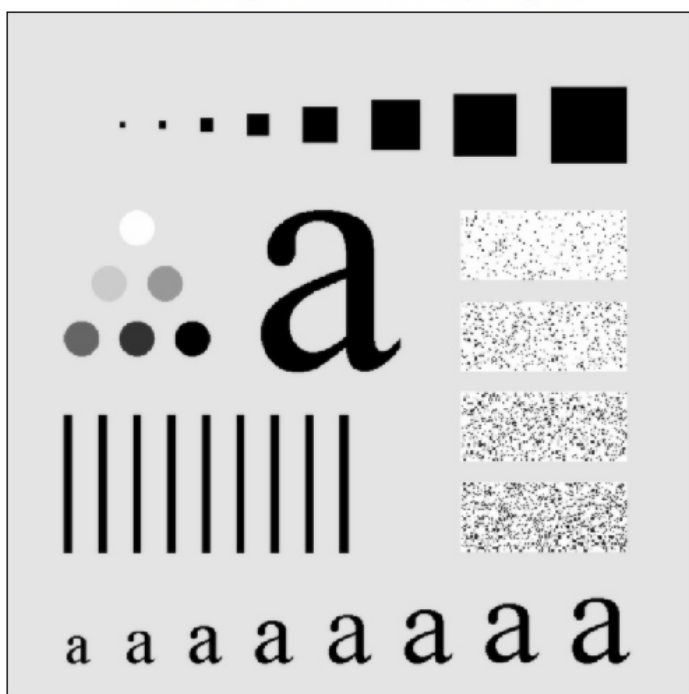


Figure 4.4: Image Smoothing with a Butterworth Lowpass Filter

### Scilab code Exa 4.18 Image Smoothing using Gaussian Lowpass Filter

```
1 //Ex4_18
2 //Image Smoothing Using Gaussian Lowpass Filter.
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
   Woods
9
10 clc;
11 close;
12 clear;
13 xdel(winsid())//to close all currently open figure(s
   ).
14
15 function [H]=lowpassfilter(type1,M,N,D0,n)//
   lowpassfilter is used to filter an image .
16     u=0:(M-1);
17     v=0:(N-1);
18     idx=find(u>M/2);
19     u(idx)=u(idx)-M;
20     idy=find(v>N/2);
21     v(idy)=v(idy)-N;
22     [U,V]=meshgrid(v,u);
23     D=sqrt(U.^2+V.^2);
24     select type1
25
26     case 'gaussian'
27         H=exp(-(D.^2)./(2*(D0^2)));
28     else
29         disp('Unknownfiltertype.')
```



```

30     end
31 endfunction
32
33
34
35 /////////////////////////////////////////////////////////////////// Main Programm
36 ///////////////////////////////////////////////////////////////////
37 a=imread("Ex4_18.tif");
38 //gray=rgb2gray(a);
39 gray=im2double(a);
40 figure,ShowImage(gray,'Gray Image');
41 title('Original Image');
42 [M,N]=size(gray);
43
44 h=fft2(gray);//fft2() is used to find 2-Dimensional
45 //Fast Fourier Transform of an matrix
46 i=log(1+abs(h));
47 in=fftshift(i);//fftshift() is used to rearrange the
48 //fft output, moving the zero frequency to the
49 //center of the spectrum.
50
51 inm=mat2gray(in)
52 figure,ShowImage(inm,'Frequency Spectrum');
53 title('Frequency Spectrum','color','blue','fontsize',
54 //4);
55
56 /////////////////////////////////////////////////////////////////// Filtering With Cut-off
57 //Frequency 10 ///////////////////////////////////////////////////////////////////
58 filt=lowpassfilter('gaussian',M,N,10); // Function
59 //which generate Filter Mask Corresponding to Low
60 //Frequency
61 //filt_shift=fftshift(filt);
62 //figure,ShowImage(filt_shift,'Filter Mask');
63 //title('Filter Mask to Specific Cut-Off Frequency')
64 //;
65 n=filt.*h;//Multiply the Original Spectrum with the
66 //Filter Mask.
67 Image_filter=real(ifft(n));

```

```

58 Image_filter=mat2gray(Image_filter)
59 figure,ShowImage(Image_filter,'Filtered Image');
60 title('Filtered Image with Cut-Off Frequency 10','
        color','blue','fontsize',4);
61
62
63 //////////////////////////////////// Filtering With Cut-off
        Frequency 30 ////////////////////////////////////
64 filt=lowpassfilter('gaussian',M,N,30); // Function
        which generate Filter Mask Corresponding to Low
        Frequency
65 //filt_shift=fftshift(filt);
66 //figure,ShowImage(filt_shift,'Filter Mask');
67 //title('Filter Mask to Specific Cut-Off Frequency')
        ;
68 n=filt.*h;//Multiply the Original Spectrum with the
        Filter Mask.
69 Image_filter=real(iffn(n));
70 Image_filter=mat2gray(Image_filter)
71 figure,ShowImage(Image_filter,'Filtered Image');
72 title('Filtered Image with Cut-Off Frequency 30','
        color','blue','fontsize',4);
73
74
75 //////////////////////////////////// Filtering With Cut-off
        Frequency 60 ////////////////////////////////////
76 filt=lowpassfilter('gaussian',M,N,60); // Function
        which generate Filter Mask Corresponding to Low
        Frequency
77 //filt_shift=fftshift(filt);
78 //figure,ShowImage(filt_shift,'Filter Mask');
79 //title('Filter Mask to Specific Cut-Off Frequency')
        ;
80 n=filt.*h;//Multiply the Original Spectrum with the
        Filter Mask.
81 Image_filter=real(iffn(n));
82 Image_filter=mat2gray(Image_filter)
83 figure,ShowImage(Image_filter,'Filtered Image');

```

```

84 title('Filtered Image with Cut-Off Frequency 60', '
      color', 'blue', 'fontsize', 4);
85
86
87 ////////////////////////////////////////////////// Filtering With Cut-off
      Frequency 160 //////////////////////////////////////
88 filt=lowpassfilter('gaussian',M,N,160); // Function
      which generate Filter Mask Corresponding to Low
      Frequency
89 //filt_shift=fftshift(filt);
90 //figure,ShowImage(filt_shift,'Filter Mask');
91 //title('Filter Mask to Specific Cut-Off Frequency')
      ;
92 n=filt.*h;//Multiply the Original Spectrum with the
      Filter Mask.
93 Image_filter=real(iff(filt));
94 Image_filter=mat2gray(Image_filter)
95 figure,ShowImage(Image_filter,'Filtered Image');
96 title('Filtered Image with Cut-Off Frequency 160', '
      color', 'blue', 'fontsize', 4);
97
98
99 ////////////////////////////////////////////////// Filtering With Cut-off
      Frequency 460 //////////////////////////////////////
100 filt=lowpassfilter('gaussian',M,N,460); // Function
      which generate Filter Mask Corresponding to Low
      Frequency
101 //filt_shift=fftshift(filt);
102 //figure,ShowImage(filt_shift,'Filter Mask');
103 //title('Filter Mask to Specific Cut-Off Frequency')
      ;
104 n=filt.*h;//Multiply the Original Spectrum with the
      Filter Mask.
105 Image_filter=real(iff(filt));
106 Image_filter=mat2gray(Image_filter)
107 figure,ShowImage(Image_filter,'Filtered Image');
108 title('Filtered Image with Cut-Off Frequency 460', '
      color', 'blue', 'fontsize', 4);

```

Filtered Image with Cut-Off Frequency 30



Figure 4.5: Image Smoothing using Gaussian Lowpass Filter

---

check Appendix ?? for dependency:

Ex4\_19.tif

Filtered Image with Cut-Off Frequency 160

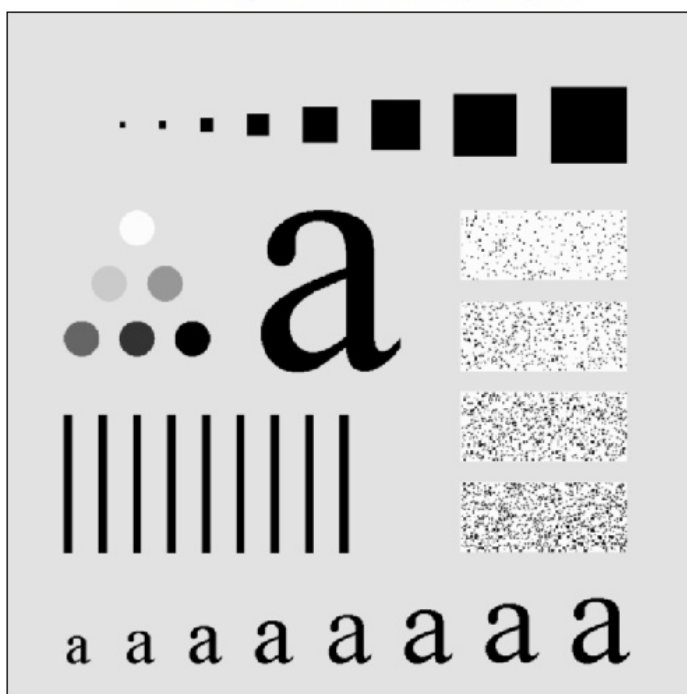


Figure 4.6: Image Smoothing using Gaussian Lowpass Filter

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

```
1 //Ex4_19
2 //Using Highpass Filter and Thresholding for Image
   Enhancement
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
   Woods
9
10
11 clc;
12 close;
13 clear;
14 xdel(winsid())//to close all currently open figure(s
   ).
15
16 function [H]=lowpassfilter(type1,M,N,D0,n)//
   lowpassfilter is used to filter an image .
17     u=0:(M-1);
18     v=0:(N-1);
19     idx=find(u>M/2);
20     u(idx)=u(idx)-M;
21     idy=find(v>N/2);
22     v(idy)=v(idy)-N;
23     [U,V]=meshgrid(v,u);
24     D=sqrt(U.^2+V.^2);
25     select type1
26
27     case 'ideal '
28         H=double(D<=D0);
29
30         case 'Laplacian '
31             H=1+(4*(%pi)^2*D^2);
32
```

```

33     case 'butterworth'
34         if argn(2)==4
35             n=1;
36         end
37         H = ones(M,N) ./ (1+(D./D0) .^(2*n));
38
39         case 'gaussian'
40             H=exp(-(D.^2) ./ (2*(D0^2)));
41         else
42             disp('Unknown filter type. ')
43         end
44     endfunction
45
46
47
48     ////////////////////////////////////// Main Programm
49     //////////////////////////////////////
49     a=imread('Ex4_19.tif');
50     //gray=rgb2gray(a);
51     gray=im2double(imresize(a,[540 540]));
52
53     figure, ShowImage(gray, 'Gray Image');
54     title('Original Image', 'color', 'blue', 'fontsize', 4);
55     [M,N]=size(gray);
56
57     h=fft2(gray); //fft2() is used to find 2-Dimensional
58                 Fast Fourier Transform of an matrix
59     i=log(1+abs(h));
60     in=fftshift(i); //fftshift() is used to rearrange the
61                 fft output, moving the zero frequency to the
62                 center of the spectrum.
63
64     inm=mat2gray(in)
65     //figure, ShowImage(inm, 'Frequency Spectrum');
66     //title('Frequency Spectrum');
67
68     filt=1-lowpassfilter('butterworth',M,N,50,4); //
69                 User Define Function which generate Filter Mask
70     filt_shift=fftshift(filt);

```

```

66 //figure ,ShowImage(filt_shift , 'Filter Mask');
67 //title('Filter Mask to Specific Cut-Off Frequency')
   ;
68
69 n=filt.*h;//Multiply the Original Spectrum with the
   Filter Mask.
70 Image_filter=real(iffn(n));
71 Image_filter=mat2gray(Image_filter)
72 figure,ShowImage(Image_filter,'Filtered Image');
73 title('Filtered Image with Specific Cut-Off
   Frequency','color','blue','fontsize',4);
74
75 thr = maskthresh(Image_filter);
76
77 Image_Enhance=im2bw(Image_filter,thr);
78 figure,ShowImage(Image_Enhance,'Filtered Image');
79 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

```

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

```



Filtered Image with Specific Cut-Off Frequency

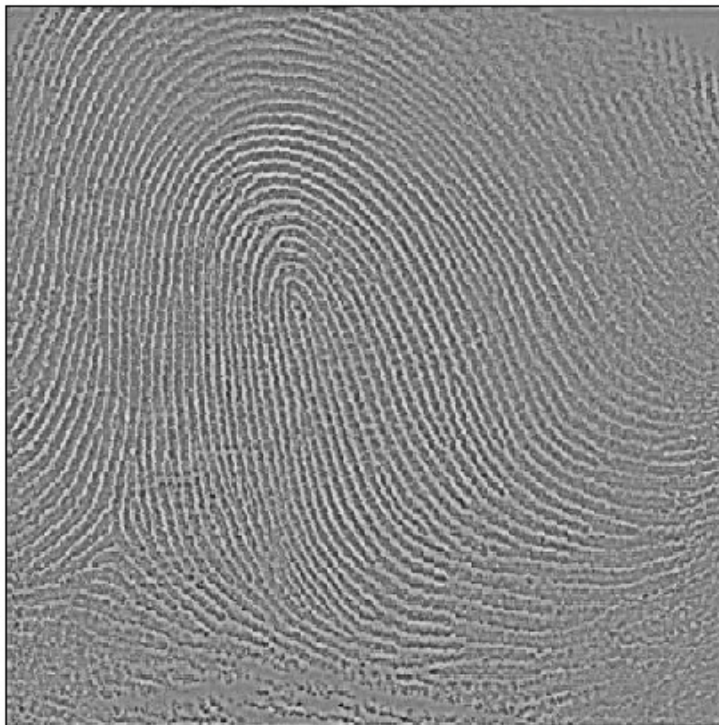


Figure 4.7: Using Highpass Filter and Thresholding for Image enhancement

Enhance Image

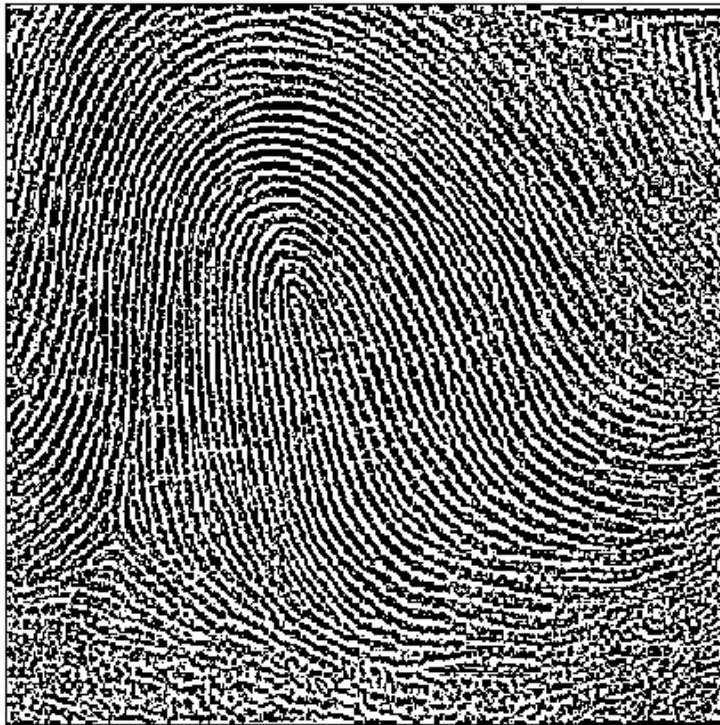


Figure 4.8: Using Highpass Filter and Thresholding for Image enhancement

```

8 //book author: Rafael C. Gonzalez and Richard E.
  Woods
9
10 clc;
11 close;
12 clear;
13 xdel(winsid())//to close all currently open figure(s
   ).
14
15 function [H]=lowpassfilter(type1,M,N,D0,n)//
   lowpassfilter is used to filter an image .
16     u=0:(M-1);
17     v=0:(N-1);
18     idx=find(u>M/2);
19     u(idx)=u(idx)-M;
20     idy=find(v>N/2);
21     v(idy)=v(idy)-N;
22     [U,V]=meshgrid(v,u);
23     D=sqrt(U.^2+V.^2);
24     select type1
25
26     case 'ideal '
27         H=double(D<=D0);
28
29     case 'Laplacian '
30         H_temp=double(D<=D0);
31         H=(4*(%pi)^2*D^2);
32         H=H.*H_temp;
33
34     case 'butterworth '
35         if argn(2)==4
36             n=1;
37         end
38         H = ones(M,N)/(1+(D./D0).^(2*n));
39         H_temp=ones(M,N)+(4*(%pi)^2*D^2);
40         H=H.*H_temp;
41
42     case 'gaussian '

```

```

43         H=exp(-(D.^2)./(2*(D0^2)));
44     else
45         disp('Unknown filter type. ');
46     end
47 endfunction
48
49
50
51 //////////////////////////////////////////////////////////////////// Main Programm
52 ////////////////////////////////////////////////////////////////////
52 a=imread("Ex4_20.tif");
53 //gray=rgb2gray(a);
54 gray=im2double(imresize(a,[540 540]));
55
56 figure,ShowImage(gray,'Gray Image');
57 title('Original Image','color','blue','fontsize',4);
58 [M,N]=size(gray);
59
60 h=fft2(gray);//fft2() is used to find 2-Dimensional
    Fast Fourier Transform of an matrix
61 i=log(1+abs(h));
62 in=fftshift(i);//fftshift() is used to rearrange the
    fft output, moving the zero frequency to the
    center of the spectrum.
63 inm=mat2gray(in);
64 filt=lowpassfilter('Laplacian',M,N,55); // User
    Define Function which generate Filter Mask
    Corresponding to Low Frequency
65 filt_shift=fftshift(filt);
66 n=filt.*h;//Multiply the Original Spectrum with the
    Filter Mask.
67 Image_filter=real(iffn(n));
68 Image_filter=mat2gray(Image_filter);
69
70 z=gray+Image_filter;
71 figure,ShowImage(mat2gray(z),'Filtered Image');
72 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

```
1 //Ex4_21
2 //Image Enhancement using High frequency Emphasis
  Filtering
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
  Woods
9
10
11 clc;
12 close;
13 clear;
14 xdel(winsid())//to close all currently open figure(s
  ).
15
16 function [H]=lowpassfilter(type1,M,N,D0,n)//
  lowpassfilter is used to filter an image .
17   u=0:(M-1);
18   v=0:(N-1);
19   idx=find(u>M/2);
20   u(idx)=u(idx)-M;
21   idy=find(v>N/2);
22   v(idy)=v(idy)-N;
23   [U,V]=meshgrid(v,u);
24   D=sqrt(U.^2+V.^2);
25   select type1
```

```

26
27     case 'ideal '
28         H=double(D<=D0);
29
30     case 'Laplacian '
31         H=1+(4*(%pi)^2*D^2);
32
33
34     case 'butterworth '
35         if argn(2)==4
36             n=1;
37         end
38         H = ones(M,N) ./ (1+(D./D0) .^(2*n));
39
40     case 'gaussian '
41         H=exp(-(D.^2) ./ (2*(D0^2)));
42     else
43         disp('Unknown filter type. ')
44     end
45 endfunction
46
47
48
49 //////////////////////////////////////////////////////////////////// Main Programm
50 ////////////////////////////////////////////////////////////////////
51 a=imread("Ex4_21.tif");
52 //gray=rgb2gray(a);
53 gray=im2double(imresize(a,[540 540]));
54 figure,ShowImage(gray,'Gray Image');
55 title('Original Image','color','blue','fontsize',4);
56 [M,N]=size(gray);
57
58 h=fft2(gray); //fft2() is used to find 2-Dimensional
59               Fast Fourier Transform of an matrix
60 i=log(1+abs(h));
61 in=fftshift(i); //fftshift() is used to rearrange the
62                 fft output, moving the zero frequency to the

```

```

        center of the spectrum.
61 inm=mat2gray(in)
62 figure,ShowImage(inm,'Frequency Spectrum');
63 title('Frequency Spectrum','color','blue','fontsize',
        ,4);
64
65 ////////////////////////////////////// Filtering With
        Cut-off Frequency 10 //////////////////////////////////////
66 filt=1-lowpassfilter('gaussian',M,N,40); // User
        Define Function which generate Filter Mask
67 n=filt.*h;//Multiply the Original Spectrum with the
        Filter Mask.
68 Image_filter=real(iffn(n));
69 Image_filter=mat2gray(Image_filter)
70 figure(1),ShowImage(Image_filter,'Filtered Image');
71 title('Filtered Image (High Pass) with Cut-Off
        Frequency 40','color','blue','fontsize',4);
72
73
74 ////////////////////////////////////// high boost filtering
        //////////////////////////////////////
75 filt=0.5+(0.75.*(1-lowpassfilter('gaussian',M,N
        ,40,4))); // User Define Function which generate
        Filter Mask
76 n=filt.*h;//Multiply the Original Spectrum with the
        Filter Mask.
77 Image_filter=real(iffn(n));
78
79 Image_filter=mat2gray(Image_filter)
80 figure,ShowImage(Image_filter,'Filtered Image');
81 title('Filtered Image with Specific Cut-Off
        Frequency','color','blue','fontsize',4);
82
83
84 Image_Enhance=bricontra(Image_filter,180,170,'m');
        // Brightness Contrast agjustment (Intensity
        Transformation)
85 figure,ShowImage(Image_Enhance,'Filtered Image');

```

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

---

check Appendix ?? for dependency:

Ex4\_22.tif

#### Scilab code Exa 4.22 Image Enhancement using Homomorphic Filtering

```
1 //Ex4_22
2 // Image Enhancement using Homomorphic Filtering
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
  Woods
9
10 clc;
11 close;
12 clear;
13 xdel(winsid())//to close all currently open figure(s
  ).
14
15 function [H]=filter(type1,M,N,D0,low,high,c)//
  lowpassfilter is used to filter an image .
16     u=0:(M-1);
17     v=0:(N-1);
18     idx=find(u>M/2);
19     u(idx)=u(idx)-M;
20     idy=find(v>N/2);
21     v(idy)=v(idy)-N;
```



Filtered Image with Specific Cut-Off Frequency

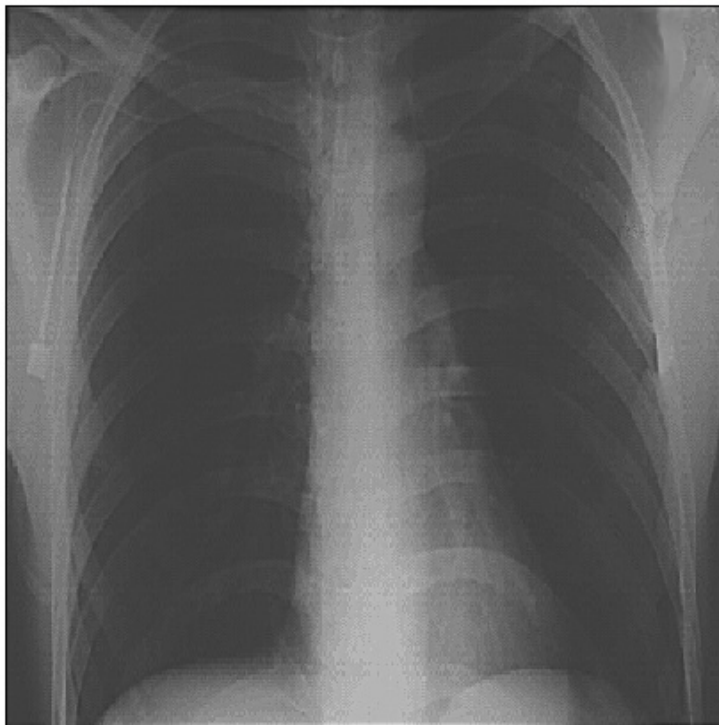


Figure 4.9: Image Enhancement using High Frequency Emphasis Filtering

Enhance Image

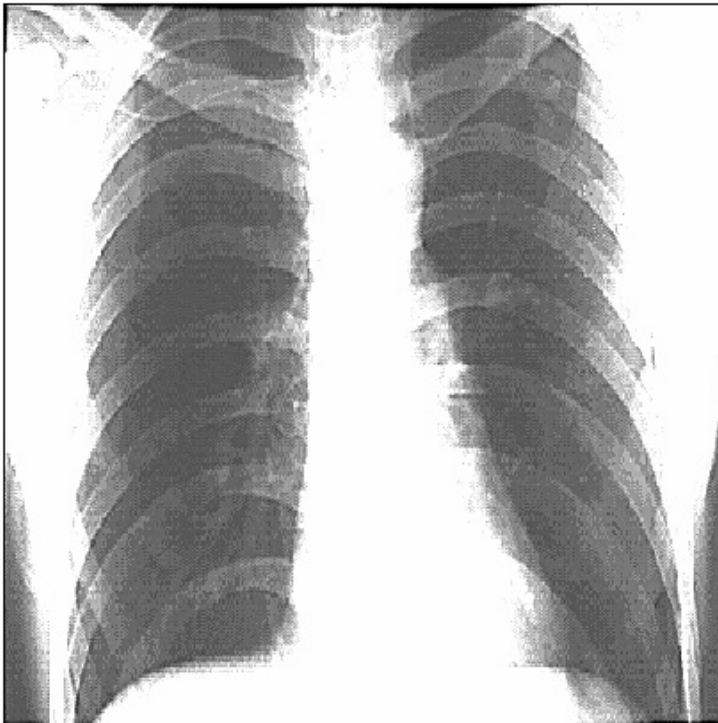


Figure 4.10: Image Enhancement using High Frequency Emphasis Filtering

```

22     [U,V]=meshgrid(v,u); // Generate 2-d matrix from
        1-d matrix
23     D=sqrt(U.^2+V.^2); // distnace calculation
24     select type1
25         case 'Homomorphic '
26             H=((high-low).*(1-(exp(-c*(D.^2)./(D0^2))))))
                +low;
27     else
28         disp('Unknownfiltertype. ')
29     end
30 endfunction
31
32
33 //////////////////////////////////////////////////////////////////// Main Programm
        ////////////////////////////////////////////////////////////////////
34
35 a=imread("Ex4_22.tif");
36 //gray=rgb2gray(a);
37 gray=im2double(imresize(a,[540 540]));
38
39 figure,ShowImage(gray,'Gray Image');
40 title('Original Image','color','blue','fontsize',4);
41 [M,N]=size(gray);
42
43 h=fft2(gray);//fft2() is used to find 2-Dimensional
        Fast Fourier Transform of an matrix
44 i=log(1+abs(h));
45 in=fftshift(i);//fftshift() is used to rearrange the
        fft output, moving the zero frequency to the
        center of the spectrum.
46 inm=mat2gray(in);
47 low=0.25;
48 high=2;
49 c=1;
50 D0=80;
51 filt=filter('Homomorphic',M,N,D0,low,high,c); //
        User Define Function which generate Filter Mask
52

```

```

53 n=filt.*h;//Multiply the Original Spectrum with the
    Filter Mask.
54 Image_filter=real(iffn(n));
55 //Image_Enhance = hiseq(a);
56
57 Image_filter=mat2gray(Image_filter);
58 figure,ShowImage(Image_filter,'Filtered Image');
59 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

```

1 //Ex4_23
2 // Reduction of Moire Pattern Using Notch Filtering
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
    Woods
9
10 clc;
11 close;
12 clear;
13 xdel(winsid())//to close all currently open figure(s
    ).
14
15 function [H]=notchfilter(type1,M,N,D0,n)//notchfilter
    is used to filter an image .
16     u=0:(M-1);
17     v=0:(N-1);

```

```

18     idx=find(u>M/2);
19     u(idx)=u(idx)-M;
20     idy=find(v>N/2);
21     v(idy)=v(idy)-N;
22     [U,V]=meshgrid(v,u);
23     D=sqrt(U.^2+V.^2);
24     x=[41 45 82 86 162 166 203 207];
25     y=[112 55 112 56 114 58 115 58];
26     select type1
27         case 'ideal'
28             //H=double(D<=D0);
29 H=ones(M,N);
30 for a=1:M
31     for b=1:N
32         for i=1:length(x)
33             d=sqrt((a-x(i))*(a-x(i))+(b-y(i))*(b-y(i)));
34             if (d<D0)
35                 //H(a,b)=1-(1/(1+(d/D0)^(2*n)));
36                 H(a,b)=0
37             end
38         end
39     end
40 end
41
42     case 'butterworth'
43         if argn(2)==4
44             n=1;
45         end
46         //H = ones(M,N)./(1+(D./D0).^(2*n));
47         H=ones(M,N);
48     for a=1:M
49     for b=1:N
50         for i=1:length(x)
51             d=sqrt((a-x(i))*(a-x(i))+(b-y(i))*(b-y(i)));
52             if (d<D0)
53                 H(a,b)=1-(1/(1+(d/D0)^(2*n)));
54                 //H(a,b)=0
55             end

```

```

56         end
57     end
58 end
59
60     case 'gaussian'
61         //H=exp(-(D.^2) ./ (2*(D0^2)));
62         H=ones(M,N);
63     for a=1:M
64     for b=1:N
65         for i=1:length(x)
66             d=sqrt((a-x(i))*(a-x(i))+(b-y(i))*(b-y(i)));
67             if (d<D0)
68                 //H(a,b)=1-(1/(1+(d/D0)^(2*n)));
69                 H(a,b)=1-(exp(-(d.^2) ./ (2*(D0^2))));
70                 //H(a,b)=0
71             end
72         end
73     end
74 end
75     else
76         disp('Unknown filter type. ')
77     end
78
79 endfunction
80
81
82 //////////////////////////////////////////////////////////////////// Main Programm
83 ////////////////////////////////////////////////////////////////////
84 a=imread("Ex4_23.tif");
85 //gray=rgb2gray(a);
86 gray=im2double(a);
87
88 figure, ShowImage(gray, 'Gray Image');
89 title('Original Image', 'color', 'blue', 'fontsize', 4);
90 [M,N]=size(gray);
91
92 h=fft2(gray); //fft2() is used to find 2-Dimensional

```

```

Fast Fourier Transform of an matrix
93 i=log(1+abs(h));
94 in=fftshift(i); //fftshift() is used to rearrange the
    fft output, moving the zero frequency to the
    center of the spectrum.
95 inm=mat2gray(in)
96 figure,ShowImage(inm,'Frequency Spectrum');
97 title('Frequency Spectrum','color','blue','fontsize'
    ,4);
98
99 filt=notchfilter('gaussian',M,N,9,2); // User Define
    Function which generate Filter Mask
    Corresponding to Low Frequency
100
101 //filt_shift=fftshift(filt);
102 n=filt.*fftshift(h); //Multiply the Original Spectrum
    with the Filter Mask.
103 figure,ShowImage(abs(n),'Frequency Spectrum');
104 title('Spectrum After Filtering','color','blue','
    fontsize',4);
105 Image_filter=real(ifft(fftshift(n)));
106 Image_filter=mat2gray(Image_filter)
107 figure,ShowImage(Image_filter,'Filtered Image');
108 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

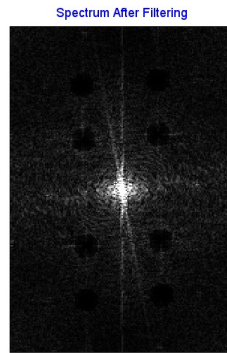


Figure 4.11: Reduction of Moire Patterns Using Notch Filtering



Figure 4.12: Reduction of Moire Patterns Using Notch Filtering



```

1 //Ex4_24
2 // Enhancement of Corrupted Cassini Saturn Image by
   Notch Filtering
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
   Woods
9
10 clc;
11 close;
12 clear;
13 xdel(winsid())//to close all currently open figure(s
   ).
14
15 function [H]=notchfilter(M,N,W)//notchfilter is used
   to filter an image .
16     H=ones(M,N);
17     H(1:ceil(M/2-5),ceil(N/2-W/2):ceil(N/2+W/2))
       =0;
18     H(ceil(M/2+5):M,ceil(N/2-W/2):ceil(N/2+W/2))
       =0;
19
20 endfunction
21
22
23
24 /////////////////////////////////////////////////// Main Programm
   //////////////////////////////////////
25 a=imread("Ex4_24.tif");
26 //gray=rgb2gray(a);
27 gray=im2double(a);
28
29 figure,ShowImage(gray,'Gray Image');
30 title('Original Image','color','blue','fontsize',4);
31 [M,N]=size(gray);

```

```

32
33 h=fft2(gray); //fft2() is used to find 2-Dimensional
    Fast Fourier Transform of an matrix
34 i=log(1+abs(h));
35 in=fftshift(i); //fftshift() is used to rearrange the
    fft output, moving the zero frequency to the
    center of the spectrum.
36 inm=mat2gray(in)
37 figure,ShowImage(inm,'Frequency Spectrum');
38 title('Frequency Spectrum','color','blue','fontsize',
    ,4);
39
40 filt=notchfilter(M,N,7); // User Define Function
    which generate Filter Mask Corresponding to Low
    Frequency
41 filt_pass=1-filt;
42 //filt_shift=fftshift(filt);
43 figure,ShowImage(filt,'Filter Mask');
44 title('Filter Mask (Band stop) to Specific Cut-Off
    Frequency','color','blue','fontsize',4);
45
46 n=filt.*fftshift(h); //Multiply the Original Spectrum
    with the Filter Mask.
47 Image_filter=real(iff(fftshift(n)));
48 Image_filter=mat2gray(Image_filter)
49 figure,ShowImage(Image_filter,'Filtered Image');
50 title('Filtered Image with Specific Cut-Off
    Frequency','color','blue','fontsize',4);
51
52
53 figure,ShowImage(filt_pass,'Filter Mask');
54 title('Filter Mask (Band Pass) to Specific Cut-Off
    Frequency','color','blue','fontsize',4);
55
56 n=filt_pass.*fftshift(h); //Multiply the Original
    Spectrum with the Filter Mask.
57 Image_filter=real(iff(fftshift(n)));
58 Image_filter=mat2gray(Image_filter)

```

```
59 figure, ShowImage(Image_filter, 'Filtered Image');  
60 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**

```
1 //Ex5_1
2 // Noisy Images and their Histogram
3 //To plot the PDF of different Noise Distribution
  and add the same to the gray scale image.
4 //(I)Gaussian (II)Uniform (III)Salt & Pepper (
  IV)Log Normal (V)Rayleigh (VI)Erlang (VII)
  Exponetial
5 // Version : Scilab 5.4.1
6 // Operating System : Window-xp, Window-7
7 //Toolbox: Image Processing Design 8.3.1-1
8 //Toolbox: SIVP 0.5.3.1-2
9 //Reference book name : Digital Image Processing
10 //book author: Rafael C. Gonzalez and Richard E.
  Woods
11
```

```

12 clc;
13 close;
14 clear;
15 xdel(winsid())//to close all currently open figure(s
    ).
16
17 function R=imnoise2(type,M,N,a,b)
18     if argn(2)==3
19         a=0; b=1;
20     end
21
22     select type
23
24     case 'gaussian'
25         rand("normal")
26         R=a+b*rand(M,N);
27
28     case 'uniform'
29         R=a+(b-a)*rand(M,N,"uniform");
30
31     case 'salt & pepper'
32         if argn(2)==3
33             a = 0.15; b = 0.15;
34         end
35         if (a+b) > 1
36             error('The sum Pa + Pb must not exceed
                1. ');
37         end
38         R(1:M,1:N) = 0.5;
39         X = rand(M,N);
40         [r c] = find(X<=a);
41         for i=1:length(r)
42             R(r(i),c(i)) = 0;
43         end
44         u = a + b;
45         [r c] = find(X>a & X<=u);
46         for i=1:length(r)
47             R(r(i),c(i)) = 255;

```

```

48         end
49
50     case 'lognormal'
51         if argn(2)==3
52             a = 1; b = 0.25;
53         end
54         R = a*exp(b*mtlb_randn(M,N));
55
56     case 'rayleigh'
57         if argn(2)==3
58             a = 1; b = 0.25;
59         end
60         R = a + (((-b)*(log(1-rand(M,N,"uniform"))))
61             .^0.5);
62
63     case 'exponential'
64         if argn(2)==3
65             a = 1;
66         end
67         if a<=0
68             error('Parameter a must be positive for
69                 exponential type. ');
70         end
71         k = -1/a;
72         R = k*log(1-rand(M,N,"uniform"));
73
74     case 'erlang'
75         if (b ~= round(b) | b <= 0)
76             error('Param b must be positive for
77                 integer for Erlang. ')
78         end
79         k = -1/a;
80         R = zeros(M,N);
81         for j=1:b
82             R = R + k*log(1-rand(M,N,"uniform"));
83         end
84     else

```

```

83         disp('Unknown filter type. ')
84     end
85
86 endfunction
87
88
89
90 //////////////////////////////////////////////////// Main
91     Programm ////////////////////////////////////////////////////
92 gray=imread("Ex5_1.tif");
93 //gray=rgb2gray(a);
94 //gray=im2double(gray);
95 figure,ShowImage(gray,'Gray Image');
96 title('Original Image');
97 [M,N]=size(gray);
98 [count,cell]=imhist(gray);
99 figure,bar(cell,count,0.2);
100 mtlb_axis([0 255 0 35000]);
101 title('Histogram of Original Image');
102
103 //////////////////////////////////////////////////// Gaussian
104     Noise ////////////////////////////////////////////////////
105 r1=imnoise2('gaussian',M,N,15,5); // Generate
106     Gaussian Noise with Given Mean and Variance
107 gray_noise_gaussian=gray+(r1);
108 figure,ShowImage(gray_noise_gaussian,'Gray Image
109     with Noise');
110 title('Gray Image with Noise gaussian');
111 [count,cell]=imhist(gray_noise_gaussian);
112 figure;bar(cell,count,1.2);
113 mtlb_axis([0 255 0 3000]);
114 title('Gaussian');
115
116 //////////////////////////////////////////////////// Rayleigh
117     Noise ////////////////////////////////////////////////////
118 r2=imnoise2('rayleigh',M,N,0,55); // Generate
119     rayleigh Noise
120 gray_noise_rayleigh=gray+(r2);

```

```

115 figure, ShowImage(gray_noise_rayleigh, 'Gray Image
    with Noise');
116 title('Gray Image with Noise rayleigh');
117 [count, cell]=imhist(gray_noise_rayleigh);
118 figure; bar(cell, count, 1.2);
119 mtlb_axis([0 255 0 4000]);
120 title('Rayleigh');
121
122 //////////////////////////////////////// Erlang (
    Gamma) Noise ////////////////////////////////////////
123 r3=imnoise2('erlang',M,N,2,15); // Generate erlang
    Noise
124 gray_noise_erlang=gray+(r3);
125 figure, ShowImage(gray_noise_erlang, 'Gray Image with
    Noise');
126 title('Gray Image with Noise erlang(Gamma)');
127 [count, cell]=imhist(gray_noise_erlang);
128 figure; bar(cell, count, 1.2);
129 mtlb_axis([0 255 0 9500]);
130 title('Erlang (Gamma)');
131
132 ////////////////////////////////////////
    Exponential Noise ////////////////////////////////////////
133 r4=imnoise2('exponential',M,N,0.15); //Generate
    exponential Noise
134 gray_noise_exponential=gray+(r4);
135 figure, ShowImage(gray_noise_exponential, 'Gray Image
    with Noise');
136 title('Gray Image with Noise exponential');
137 [count, cell]=imhist(gray_noise_exponential);
138 figure; bar(cell, count, 1.2);
139 mtlb_axis([0 255 0 4500]);
140 title('Exponential');
141
142 //////////////////////////////////////// Uniform
    Noise ////////////////////////////////////////
143 r5=imnoise2('uniform',M,N,0,20); // Generate
    uniform Noise

```



```

144 gray_noise_uniform=gray+(r5);
145 figure,ShowImage(gray_noise_uniform,'Gray Image with
      Noise');
146 title('Gray Image with Noise uniform');
147 [count,cell]=imhist(gray_noise_uniform);
148 figure;bar(cell,count,1.2);
149 mtlb_axis([0 255 0 2000]);
150 title('Uniform');
151
152 //////////////////////////////////////// Salt &
      ; pepper Noise ////////////////////////////////////////
153 r6=imnoise2('salt & pepper',M,N,0.15,0.15); //
      Generate salt & pepper Noise
154 gray_noise_salt_pepper=gray+(r6);
155 figure,ShowImage(gray_noise_salt_pepper,'Gray Image
      with Noise');
156 title('Gray Image with Noise salt&pepper');
157 [count,cell]=imhist(gray_noise_salt_pepper);
158 figure;bar(cell,count,1.2);
159 mtlb_axis([0 255 0 35000]);
160 title('Salt & pepper');
161
162 ////////////////////////////////////////
      lognormal Noise ////////////////////////////////////////
163 //r7=imnoise2('lognormal',M,N,5,0.65); // Generate
      lognormal Noise
164 //gray_noise_lognormal=gray+(r7);
165 //figure,ShowImage(gray_noise_lognormal,'Gray Image
      with Noise');
166 //title('Gray Image with Noise lognormal');
167 //[count,cell]=imhist(gray_noise_lognormal);
168 //figure;bar(cell,count,1.2);
169 //mtlb_axis([0 255 0 5500]);
170 //title('lognormal');

```

---

check Appendix ?? for dependency:

Ex5\_2.tif

## Scilab code Exa 5.2 Illustration of Mean Filters

```
1 //Ex5_2
2 // Illustration of Mean Filters
3 //To impliment the Following Mean Restoration filter
4 //          (I)Arithmetic  (II)Geometric  (
5 //          III)Harmonic (IV)Contra Harmonic
6 // Version : Scilab 5.4.1
7 // Operating System : Window-xp, Window-7
8 //Toolbox: Image Processing Design 8.3.1-1
9 //Toolbox: SIVP 0.5.3.1-2
10 //Reference book name : Digital Image Processing
11 //book author: Rafael C. Gonzalez and Richard E.
12 //          Woods
13 clc;
14 close;
15 clear;
16 xdel(winsid())//to close all currently open figure(s
17 //          ).
18
19 function [f]=arithmetic_mean(v,m,n)
20 //          w=fspecial('average',m);
21 //          f=imfilter(v,w);
22 endfunction
23
24 function [f]=geometric_mean1(g,m,n);//gmean1() is
25 //          used to filter an image using Geometric mean
26 //          filter
27 //          size1=m;
28 //          q=m*n;
29 //          g=double(g);
```

```

28     [nr,nc]=size(g);
29     temp=zeros(nr+2*floor(size1/2),nc+2*floor(size1
        /2));
30     temp(ceil(size1/2):nr+ceil(size1/2)-1,ceil(size1
        /2):nc+ceil(size1/2)-1)=g(1:$,1:$)
31     temp=temp+1;
32     for i=ceil(size1/2):nr+ceil(size1/2)-1
33         for j=ceil(size1/2):nc+ceil(size1/2)-1
34             t=temp(i-floor(size1/2):1:i+floor(size1
                /2),j-floor(size1/2):1:j+floor(size1
                /2)) ;
35             temp2(i,j)=prod(t);
36         end
37     end
38     temp3=temp2.^(1/q);
39     nn=temp3(ceil(size1/2):nr+ceil(size1/2)-1,ceil(
        size1/2):nc+ceil(size1/2)-1)
40     f1=nn-1;
41     f=mat2gray(f1)
42 endfunction
43
44 function [f]=geometric_mean2(g,m,n); //gmean2() is
    used to filter an image using Geometric mean
    filter
45     size1=m;
46     q=m*n;
47     [nr,nc]=size(g);
48     temp=zeros(nr+2*floor(size1/2),nc+2*floor(size1
        /2));
49     temp(ceil(size1/2):nr+ceil(size1/2)-1,ceil(size1
        /2):nc+ceil(size1/2)-1)=g(1:$,1:$)
50     for i=ceil(size1/2):nr+ceil(size1/2)-1
51         for j=ceil(size1/2):nc+ceil(size1/2)-1
52             t=temp(i-floor(size1/2):1:i+floor(size1
                /2),j-floor(size1/2):1:j+floor(size1
                /2)) ;
53             temp2(i,j)=geomean(t);
54         end

```

```

55     end
56     nn=temp2(ceil(size1/2):nr+ceil(size1/2)-1,ceil(
        size1/2):nc+ceil(size1/2)-1)
57     f=mat2gray(nn)
58 endfunction
59
60 function [f]=Harmonic_mean(g,m,n) //harmean1() is
    used to filter an image using Harmonic mean
    filter.
61     size1=m;
62     d=m*n;
63     g=double(g);
64     [nr,nc]=size(g);
65     temp=zeros(nr+2*floor(size1/2),nc+2*floor(size1
        /2));
66     temp(ceil(size1/2):nr+ceil(size1/2)-1,ceil(size1
        /2):nc+ceil(size1/2)-1)=g(1:$,1:$);
67
68     for i=ceil(size1/2):nr+ceil(size1/2)-1
69         for j=ceil(size1/2):nc+ceil(size1/2)-1
70             t=temp(i-floor(size1/2):1:i+floor(size1
                /2),j-floor(size1/2):1:j+floor(size1
                /2)) ;
71             t1=ones(m,n)./(t+%eps);
72             t2=sum(t1);
73             temp2(i,j)=d/t2;
74         end
75     end
76     nn=temp2(ceil(size1/2):nr+ceil(size1/2)-1,ceil(
        size1/2):nc+ceil(size1/2)-1);
77     f=mat2gray(nn);
78 endfunction
79
80 function [f]=Contra_Harmonic_mean(g,m,n,Q) //
    charmean1() is use to filter an image using
    Contra Harmonic mean filter
81     size1=m;
82     d=m*n;

```

```

83     g=double(g);
84     [nr,nc]=size(g);
85     temp=zeros(nr+2*floor(size1/2),nc+2*floor(size1
           /2));
86     temp(ceil(size1/2):nr+ceil(size1/2)-1,ceil(size1
           /2):nc+ceil(size1/2)-1)=g(1:$,1:$)
87     disp(Q)
88     for i=ceil(size1/2):nr+ceil(size1/2)-1
89         for j=ceil(size1/2):nc+ceil(size1/2)-1
90             t=temp(i-floor(size1/2):1:i+floor(size1
                   /2),j-floor(size1/2):1:j+floor(size1
                   /2)) ;
91             d1=(t+%eps).^Q;
92             n1=(t+%eps).^(Q+1);
93             d2=sum(d1);
94             n2=sum(n1);
95             temp2(i,j)=n2/(d2);
96         end
97     end
98     nn=temp2(ceil(size1/2):nr+ceil(size1/2)-1,ceil(
           size1/2):nc+ceil(size1/2)-1)
99     f=nn;
100 endfunction
101
102 ////////////////////////////////////////////////////////////////////          Main
103     Programm          ////////////////////////////////////////////////////////////////////
104 gray=imread("Ex5_2.tif");
105 //gray=rgb2gray(a);
106 //gray=im2double(gray);
107 figure,ShowImage(gray,'Gray Image');
108 title('Original Image');
109 [M,N]=size(gray);
110
111 ////////////////////////////////////////////////////////////////////
112     Arithmetical Mean Filter          ////////////////////////////////////////////////////////////////////
113 v=imnoise(gray,'gaussian',0,0.02);
114 figure,ShowImage(v,'Noisy Image');

```

```

114 title('Image with Gaussian Noise');
115 m=3;n=3;
116 [f]=arithmetic_mean(v,m,n);
117 figure,ShowImage(f,'Recovered Image');
118 title('Recovered Image with Arithmetical Mean Filter
      ');
119
120 //////////////////////////////////////// Geometric
      Mean Filter ////////////////////////////////////////
121 v=imnoise(gray,'gaussian',0,0.02);
122 figure,ShowImage(v,'Noisy Image');
123 title('Image with Gaussian Noise');
124 m=3;n=3;
125 [f]=geometric_mean1(v,m,n);
126 figure,ShowImage(f,'Recovered Image');
127 title('Recovered Image with Geometric Mean Filter');
128
129
130 //////////////////////////////////////// Geometric
      Mean Filter ////////////////////////////////////////
131 //v=imnoise(gray,'gaussian',0,0.02);
132 //figure,ShowImage(v,'Noisy Image');
133 //title('Image with Gaussian Noise');
134 //m=3;n=3;
135 //[f]=geometric_mean2(v,m,n);
136 //figure,ShowImage(f,'Recovered Image');
137 //title('Recovered Image with Geometric Mean Filter
      ');
138
139
140 //////////////////////////////////////// Harmonic
      Mean Filter ////////////////////////////////////////
141 //temp(1:M,1:N)=0.5;
142 //r3=imnoise(temp,'salt & pepper',0.1); //
      Generate salt & pepper Noise
143 //gray_noise_salt=gray; // Add salt
      Noise Only
144 //[r c]=find(r3==1);

```

```

145 //          for i=1:length(r)
146 //              gray_noise_salt(r(i),c(i)) = 255;
147 //          end
148 //figure,ShowImage(gray_noise_salt,'Noisy Image');
149 //title('Image with Salt Noise');
150 //m=3;n=3;
151 //[f]=Harmonic_mean(gray_noise_salt,m,n);
152 //figure,ShowImage(f,'Recovered Image');
153 //title('Recovered Image with Harmonic Mean Filter')
    ;
154 //
155 /////////////////////////////////////////////////// Contra_Harmonic
    Mean Filter (Pepper) ///////////////////////////////////
156 temp(1:M,1:N)=0.5;
157 r3=imnoise(temp,'salt & pepper',0.05); //Generate
    salt & pepper Noise
158 gray_noise_pepper=gray; //Add
    pepper Noise Only
159 [r c]=find(r3==0); //Find
    pepper Noise Only
160     for i=1:length(r)
161         gray_noise_pepper(r(i),c(i)) = 0;
162     end
163 figure,ShowImage(gray_noise_pepper,'Noisy Image');
164 title('Image with pepper Noise');
165 m=3;n=3;Q=1.5;
166 [f]=Contra_Harmonic_mean(gray_noise_pepper,m,n,Q);
167 figure,ShowImage(f,'Recovered Image');
168 title('Recovered Image with Contra Harmonic Mean
    Filter [ Q=1.5 ]');
169
170 ///////////////////////////////////////////////////
    Contra_Harmonic Mean Filter (Salt)
    ///////////////////////////////////
171 temp(1:M,1:N)=0.5;
172 r3=imnoise(temp,'salt & pepper',0.1); //Generate
    salt & pepper Noise
173 gray_noise_salt=gray; //Add salt

```

```

Noise Only
174 [r c]=find(r3==1);
175     for i=1:length(r)
176         gray_noise_salt(r(i),c(i)) = 255;
177     end
178 figure,ShowImage(gray_noise_salt,'Noisy Image');
179 title('Image with Salt Noise');
180 m=3;n=3;Q=-1.5;
181 [f]=Contra_Harmonic_mean(gray_noise_salt,m,n,Q);
182 figure,ShowImage(f,'Recovered Image');
183 title('Recovered Image with Contra Harmonic Mean
    Filter [ Q=-1.5 ]');
184
185
186 //////////////////////////////////////// Contra_Harmonic Mean
    Filter (Pepper) ////////////////////////////////////////
187 temp(1:M,1:N)=0.5;
188 r3=imnoise(temp,'salt & pepper',0.05); //Generate
    salt & pepper Noise
189 gray_noise_pepper=gray; // Add
    pepper Noise Only
190 [r c]=find(r3==0); //Find
    pepper Noise Only
191     for i=1:length(r)
192         gray_noise_pepper(r(i),c(i)) = 0;
193     end
194 figure,ShowImage(gray_noise_pepper,'Noisy Image');
195 title('Image with pepper Noise');
196 m=3;n=3;Q=-1.5;
197 [f]=Contra_Harmonic_mean(gray_noise_pepper,m,n,Q);
198 figure,ShowImage(f,'Recovered Image');
199 title('Recovered Image with Contra Harmonic Mean
    Filter [ Q=-1.5 ]');
200
201 ////////////////////////////////////////
    Contra_Harmonic Mean Filter (Salt)
    ////////////////////////////////////////
202 temp(1:M,1:N)=0.5;

```



```

203 r3=imnoise(temp,'salt & pepper',0.1);    //Generate
      salt & pepper Noise
204 gray_noise_salt=gray;                    //Add salt
      Noise Only
205 [r c]=find(r3==1);
206     for i=1:length(r)
207         gray_noise_salt(r(i),c(i)) = 255;
208     end
209 figure,ShowImage(gray_noise_salt,'Noisy Image');
210 title('Image with Salt Noise');
211 m=3;n=3;Q=1.5;
212 [f]=Contra_Harmonic_mean(gray_noise_salt,m,n,Q);
213 figure,ShowImage(f,'Recovered Image');
214 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

```

1 //Ex5_3
2 // Illustration of Order Statistic filter
3 //To impliment the Following Order Statistic
      Restoration filter
4 //          (I)Median (II)MAX (III)MIN (IV
      )Mid Point (V)Alpha trimmed.
5
6 // Version : Scilab 5.4.1
7 // Operating System : Window-xp, Window-7
8 //Toolbox: Image Processing Design 8.3.1-1
9 //Toolbox: SIVP 0.5.3.1-2
10 //Reference book name : Digital Image Processing
11 //book author: Rafael C. Gonzalez and Richard E.
      Woods

```

```

12
13 clc;
14 close;
15 clear;
16 xdel(winsid())//to close all currently open figure(s
    ).
17
18 function [f]=arithmetic_mean(v,m,n)
19         w=fspecial('average',m);
20         f=imfilter(v,w);
21 endfunction
22
23 function [f]=geometric_mean1(g,m,n);//gmean1() is
    used to filter an image using Geometric mean
    filter
24     size1=m;
25     q=m*n;
26     g=double(g);
27     [nr,nc]=size(g);
28     temp=zeros(nr+2*floor(size1/2),nc+2*floor(size1
        /2));
29     temp(ceil(size1/2):nr+ceil(size1/2)-1,ceil(size1
        /2):nc+ceil(size1/2)-1)=g(1:$,1:$)
30     temp=temp+1;
31     for i=ceil(size1/2):nr+ceil(size1/2)-1
32         for j=ceil(size1/2):nc+ceil(size1/2)-1
33             t=temp(i-floor(size1/2):1:i+floor(size1
                /2),j-floor(size1/2):1:j+floor(size1
                /2));
34             temp2(i,j)=prod(t);
35         end
36     end
37     temp3=temp2.^(1/q);
38     nn=temp3(ceil(size1/2):nr+ceil(size1/2)-1,ceil(
        size1/2):nc+ceil(size1/2)-1)
39     f1=nn-1;
40     f=mat2gray(f1)
41 endfunction

```

```

42
43 function [f]=restoration_filter(v,type,m,n,Q,d)
44     if argn(2) ==2
45         m=7;n=7;Q=1.5;d=10;
46     elseif argn(2) ==5
47         Q=parameter;d=parameter;
48     elseif argn(2) ==4
49         Q=1.5;d=2;
50     else
51         disp('wrong number of inputs');
52     end
53
54     select type
55
56     case 'median'
57         f=MedianFilter(v,[m n]);
58
59     case 'MIN'
60         size1=m;
61         [nr,nc]=size(v);
62         temp=zeros(nr+2*floor(size1/2),nc+2*floor(
63             size1/2));
64         temp(ceil(size1/2):nr+ceil(size1/2)-1,ceil(
65             size1/2):nc+ceil(size1/2)-1)=v(1:$,1:$);
66         for i=ceil(size1/2):nr+ceil(size1/2)-1
67             for j=ceil(size1/2):nc+ceil(size1/2)-1
68                 t=temp(i-floor(size1/2):1:i+floor(
69                     size1/2),j-floor(size1/2):1:j+
70                     floor(size1/2)) ;
71                 y=gsort(t);
72                 temp2(i-floor(size1/2),j-floor(size1
73                     /2))=min(y);
74             end
75         end
76         f=mat2gray(temp2);
77
78     case 'MAX'
79         size1=m;

```

```

75     [nr,nc]=size(v);
76     temp=zeros(nr+2*floor(size1/2),nc+2*floor(
        size1/2));
77     temp(ceil(size1/2):nr+ceil(size1/2)-1,ceil(
        size1/2):nc+ceil(size1/2)-1)=v(1:$,1:$);
78     for i=ceil(size1/2):nr+ceil(size1/2)-1
79         for j=ceil(size1/2):nc+ceil(size1/2)-1
80             t=temp(i-floor(size1/2):1:i+floor(
                size1/2),j-floor(size1/2):1:j+
                floor(size1/2)) ;
81             y=gsort(t);
82             temp2(i-floor(size1/2),j-floor(size1
                /2))=max(y);
83         end
84     end
85     f=mat2gray(temp2);
86
87     case 'Mid_Point'
88     size1=m;
89     [nr,nc]=size(v);
90     temp=zeros(nr+2*floor(size1/2),nc+2*floor(
        size1/2));
91     temp(ceil(size1/2):nr+ceil(size1/2)-1,ceil(
        size1/2):nc+ceil(size1/2)-1)=v(1:$,1:$);
92     for i=ceil(size1/2):nr+ceil(size1/2)-1
93         for j=ceil(size1/2):nc+ceil(size1/2)-1
94             t=temp(i-floor(size1/2):1:i+floor(
                size1/2),j-floor(size1/2):1:j+
                floor(size1/2)) ;
95             y=gsort(t);
96             temp2(i-floor(size1/2),j-floor(size1
                /2))=0.5*(min(y)+max(y));
97         end
98     end
99     f=mat2gray(temp2);
100
101     else
102     disp('Unknown filter type. ')

```

```

103     end
104 endfunction
105
106 function [f]=alphatrim(g,m,n,d)//alphatrim() is used
    to filter an image using alpha-trimmed mean
    filter
107     size1=m;
108     [nr,nc]=size(g);
109     temp=zeros(nr+2*floor(size1/2),nc+2*floor(size1
        /2));
110     temp(ceil(size1/2):nr+ceil(size1/2)-1,ceil(size1
        /2):nc+ceil(size1/2)-1)=g(1:$,1:$)
111
112     for i=ceil(size1/2):nr+ceil(size1/2)-1
113         for j=ceil(size1/2):nc+ceil(size1/2)-1
114             t=temp(i-floor(size1/2):1:i+floor(size1
                /2),j-floor(size1/2):1:j+floor(size1
                /2))
115             y=gsort(t);
116             a=y(:)
117             b=a';
118             t1=b(1+d/2:$-d/2);
119             temp2(i-floor(size1/2),j-floor(size1/2))
                =mean(t1);
120         end
121     end
122     f=mat2gray(temp2)
123 endfunction
124
125
126 ////////////////////////////////////////////////////          Main
    Programm          ////////////////////////////////////////////////////
127
128 gray=imread("Ex5_3.tif");
129 //gray=rgb2gray(a);
130 //gray=im2double(gray);
131 figure,ShowImage(gray,'Gray Image');
132 title('Original Image');

```

```

133 [M,N]=size(gray);
134
135 ////////////////////////////////////////          Median
      Filter          ////////////////////////////////////////
136 v=imnoise(gray,'salt & pepper',0.1);
137 figure,ShowImage(v,'Noisy Image');
138 title('Original Image with Salt & Pepper Noise')
      ;
139 //Filtering the corrupted image with median filter
140 h=restoration_filter(v,'median',3,3);
141 figure,ShowImage(h,'Recovered Image');
142 title('Recovered Image with Median Filter');
143 //Filtering the corrupted image with median filter
144 h1=restoration_filter(h,'median',3,3);
145 figure,ShowImage(h1,'Recovered Image');
146 title('Recovered Image with Median Filter');
147 //Filtering the corrupted image with median filter
148 h2=restoration_filter(h1,'median',3,3);
149 figure,ShowImage(h2,'Recovered Image');
150 title('Recovered Image with Median Filter');
151
152
153 ////////////////////////////////////////          MAX Filter
      ////////////////////////////////////////
154 temp(1:M,1:N)=0.5;
155 r3=imnoise(temp,'salt & pepper',0.1);    //
      Generate salt & pepper Noise
156 gray_noise_pepper=gray;                  // Add
      Pepper Noise Only
157 [r c]=find(r3==0);
158     for i=1:length(r)
159         gray_noise_pepper(r(i),c(i)) = 0;
160     end
161 figure,ShowImage(gray_noise_pepper,'Noisy Image');
162 title('Noisy Image with Pepper Noise');
163
164 //Filtering the Salt Noise corrupted image with MAX
      filter

```

```

165 h=restoration_filter(gray_noise_pepper,'MAX',3,3);
166 figure,ShowImage(h,'Recovered Image');
167 title('Recovered Image with MAX Filter');
168
169
170 ////////////////////////////////////////          MIN
    Filter          ////////////////////////////////////////
171 temp(1:M,1:N)=0.5;
172 r3=imnoise(temp,'salt & pepper',0.1);    //
    Generate salt & pepper Noise
173 gray_noise_salt=gray;                    // Add salt
    Noise Only
174 [r c]=find(r3==1);
175     for i=1:length(r)
176         gray_noise_salt(r(i),c(i)) = 255;
177     end
178 figure,ShowImage(gray_noise_salt,'Noisy Image');
179 title('Noisy Image');
180
181 //Filtering the Salt Noise corrupted image with MIN
    filter
182 h=restoration_filter(gray_noise_salt,'MIN',3,3);
183 figure,ShowImage(h,'Recovered Image');
184 title('Recovered Image with MIN Filter');
185
186
187 ////////////////////////////////////////          Mid-
    Point Filter          ////////////////////////////////////////
188 //v=imnoise(gray,'gaussian',0,0.02);
189 //figure,ShowImage(v,'Noisy Image');
190 //title('Image with Gaussian Noise');
191 ////Filtering the Salt Noise corrupted image with
    Mid-Point filter
192 //h=restoration_filter(v,'Mid-Point',3,3);
193 //figure,ShowImage(h,'Recovered Image');
194 //title('Recovered Image with Mid-Point Filter');
195
196

```

```

197 ////////////////////////////////////////////////// Alpha
    Trimmed Filter //////////////////////////////////////////////////
198 v=imnoise(gray,'gaussian',0,0.02);
199 v=imnoise(v,'salt & pepper',0.05);
200 figure,ShowImage(v,'Noisy Image');
201 title('Image with Gaussian and Salt&Pepper Noise
    ');
202 m=5;n=5;d=5;
203 [f]=arithmetic_mean(v,m,n); // Filtering with
    Arithmetical mean
204 figure,ShowImage(f,'Recovered Image');
205 title('Recovered Image with Arithmetical Mean Filter
    ');
206 [f]=geometric_mean1(v,m,n); // Filtering with
    Geometric mean
207 figure,ShowImage(f,'Recovered Image');
208 title('Recovered Image with Geometric Mean Filter');
209 //Filtering the corrupted image with median filter
210 h=restoration_filter(v,'median',5,5); // Filtering
    with median Filtering
211 figure,ShowImage(h,'Recovered Image');
212 title('Recovered Image with Median Filter');
213 f=alphatrim(v,m,n,d); // Filtering with alphatrim
    Filtering
214 figure,ShowImage(f,'Recovered Image');
215 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

```

1 //Ex5_4
2 //Illustration of Adaptive Local Noise Reduction
    Filtering

```



```

3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
    Woods
9
10 clc;
11 clear;
12 close;
13 xdel(winsid());
14
15 ////////////////////////////////// Function File
    //////////////////////////////////
16 function [f]=arithmetic_mean(v,m,n)
17         w=fspecial('average',m);
18         f=imfilter(v,w);
19 endfunction
20
21 function [f]=geometric_mean1(g,m,n); //gmean1() is
    used to filter an image using Geometric mean
    filter
22     size1=m;
23     q=m*n;
24     g=double(g);
25     [nr,nc]=size(g);
26     temp=zeros(nr+2*floor(size1/2),nc+2*floor(size1
        /2));
27     temp(ceil(size1/2):nr+ceil(size1/2)-1,ceil(size1
        /2):nc+ceil(size1/2)-1)=g(1:$,1:$)
28     temp=temp+1;
29     for i=ceil(size1/2):nr+ceil(size1/2)-1
30         for j=ceil(size1/2):nc+ceil(size1/2)-1
31             t=temp(i-floor(size1/2):1:i+floor(size1
                /2),j-floor(size1/2):1:j+floor(size1
                /2)) ;
32             temp2(i,j)=prod(t);

```

```

33         end
34     end
35     temp3=temp2.^(1/q);
36     nn=temp3(ceil(size1/2):nr+ceil(size1/2)-1,ceil(
        size1/2):nc+ceil(size1/2)-1)
37     f1=nn-1;
38     f=mat2gray(f1)
39 endfunction
40
41
42 ////////////////////////////////////////////////// Main Programm
43 //////////////////////////////////////////////////
44 A=imread("Ex5_4.tif");
45 B = imnoise(A,'gaussian',0,0.01);
46 [rw1 ,cl1]=size(B);
47 figure;
48 ShowImage(B,'Gaussian noise added');
49 title('Image with gaussian noise','color','blue','
        fontsize',4);
50
51 //////////////////////////////////////////////////
52 //////////////////////////////////////////////////
53 //////////////////////////////////////////////////
54 //////////////////////////////////////////////////
55 //////////////////////////////////////////////////
56 //////////////////////////////////////////////////
57 //////////////////////////////////////////////////
58 //////////////////////////////////////////////////
59 //////////////////////////////////////////////////
60 //////////////////////////////////////////////////
61 //////////////////////////////////////////////////
62 //////////////////////////////////////////////////
63 //////////////////////////////////////////////////

```

```

64 ///////////////////////////////////////////////////Adaptive Local Noise Reduction
   ///////////////////////////////////////////////////
65 B= double(B);
66 M=7;
67 N=7;
68 lvar=zeros([rw1-M+1,c11-N+1]);
69 lmean=zeros([rw1-M+1,c11-N+1]);
70 temp=zeros([rw1-M+1,c11-N+1]);
71 F=zeros([rw1-M+1,c11-N+1]);
72 sz=(rw1-M+1)*(c11-N+1);
73 for i=1:rw1-M+1
74     for j=1:c11-N+1
75         temp=B(i:i+(M-1),j:j+(N-1));
76         lmean(i,j)=mean(temp);
77         lvar(i,j)=mean(temp.*temp)-mean(temp).^2;
78     end
79 end
80 nvar=sum(lvar)/sz;
81 lvar=max(lvar,nvar);
82 C=B(M/2:rw1-M/2,N/2:c11-N/2);
83 F=nvar./lvar;
84 F=F.*(C-lmean);
85 F=C-F;
86 F=uint8(F);
87 figure;
88 ShowImage(F,'Restored');
89 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

1 //Ex5\_5

```

2 // Illustration of Adaptive Median Filter
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
   Woods
9
10 clc;
11 clear;
12 close;
13 xdel(winsid());
14 A=imread("Ex5_5.tif");
15 A=imresize(A,[256 256]);
16 A=imnoise(A,'salt & pepper',0.25); // Add Sali &
   Pepper Noise
17 figure,ShowImage(A,'Salt & pepper Image');
18 title('Image with Salt & pepper noise (Density =
   0.25)', 'color', 'blue', 'fontsize',4);
19 figure,ShowImage(MedianFilter(A,[7 7]),'Median
   filter with mask 7x7');
20 title('Restored Image using Median filter with 7*7
   Mask', 'color', 'blue', 'fontsize',4);
21
22 //////////////// Adaptive Median Filter
   ////////////////
23 [r c]=size(A);
24 n=7 // Maximum Window size
25 a=(n-1)/2;
26 C=zeros(r-2*a,c-2*a);
27 for i=a+1:(r-a)
28     for j=a+1:(c-a)
29         for b=3:2:7
30             d=(b-1)/2
31             x=A(i,j);
32             p=imcrop(A,[i-d j-d b b]) // Crop the Sub
   Image form Original Iamge

```

```

33         med=median(p); // To Find Median Value
34         maxx=max(p); // To Find Max Value
35         minn=min(p); // To Find Min Value
36         if (med>minn & med<maxx) then
37             if(x>minn & x<minn) then
38                 C(j-a+1,i-a+1)=x;
39                 clear p;
40                 break;
41             else
42                 C(j-a+1,i-a+1)=med;
43                 clear p;
44                 break;
45             end
46         elseif b<7 then
47             continue;
48         else
49             C(j-a+1,i-a+1)=med;
50             clear p;
51             break;
52         end
53     end
54 end
55 end
56 figure;ShowImage(C,'Adaptive Median filter Image
    using code');
57 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

```

1 //Ex5_8
2 //Removal of Periodic Noise by Notch Filtering

```

```

3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
    Woods
9
10 clc;
11 close;
12 clear;
13 xdel(winsid())//to close all currently open figure(s
    ).
14
15 function [H]=notchfilter(M,N,W)//notchfilter is used
    to filter an image .
16     H=ones(M,N);
17     H(1:ceil(M/2-10),ceil(N/2-W/2):ceil(N/2+W/2)
        )=0;
18     H(ceil(M/2+10):M,ceil(N/2-W/2):ceil(N/2+W/2)
        )=0;
19
20 endfunction
21
22
23
24 /////////////////////////////////////////////////////////////////// Main Programm
    ///////////////////////////////////////////////////////////////////
25 a=imread("Ex5_8.tif");
26 //gray=rgb2gray(a);
27 gray=im2double(imresize(a,0.5));
28 figure,ShowImage(gray,'Gray Image');
29 title('Original Image');
30 [M,N]=size(gray);
31
32 h=fft2(gray);//fft2() is used to find 2-Dimensional
    Fast Fourier Transform of an matrix
33 i=log(1+abs(h));

```

```

34 in=fftshift(i); //fftshift() is used to rearrange the
    fft output, moving the zero frequency to the
    center of the spectrum.
35 inm=mat2gray(in)
36 figure,ShowImage(inm,'Frequency Spectrum');
37 title('Frequency Spectrum');
38
39 filt=notchfilter(M,N,3); // User Define Function
    which generate Filter Mask Corresponding to Low
    Frequency
40 filt_pass=1-filt;
41 //filt_shift=fftshift(filt);
42 figure,ShowImage(filt,'Filter Mask');
43 title('Filter Mask (Band stop) to Specific Cut-Off
    Frequency');
44
45 n=filt.*fftshift(h); //Multiply the Original Spectrum
    with the Filter Mask.
46 Image_filter=real(iff(fftshift(n)));
47 Image_filter=mat2gray(Image_filter)
48 figure,ShowImage(Image_filter,'Filtered Image');
49 title('Filtered Image with Specific Cut-Off
    Frequency');
50
51
52 figure,ShowImage(filt_pass,'Filter Mask');
53 title('Filter Mask (Band Pass) to Specific Cut-Off
    Frequency');
54
55 n=filt_pass.*fftshift(h); //Multiply the Original
    Spectrum with the Filter Mask.
56 Image_filter=real(iff(fftshift(n)));
57 Image_filter=mat2gray(Image_filter)
58 figure,ShowImage(Image_filter,'Filtered Image');
59 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

```
1 //Ex5_10
2 //Image Blurring Due to Motion
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
  Woods
9
10 clc;
11 close;
12 clear;
13 xdel(winsid())//to close all currently open figure(s
  ).
14
15 gray=imread("Ex5_10.png");
16 gray=im2double(rgb2gray(gray));
17 //gray=im2double(imresize(a,0.5));
18 figure,ShowImage(gray,'Gray Image');
19 title('Original Image','color','blue','fontsize',4);
20 [M,N]=size(gray);
21
22 h=fft2(gray);//fft2() is used to find 2-Dimensional
  Fast Fourier Transform of an matrix
23 i=log(1+abs(h));
24 in=fftshift(i);//fftshift() is used to rearrange the
  fft output, moving the zero frequency to the
  center of the spectrum.
25 inm=mat2gray(in)
26
```



```

27 a=0.1;b=0.1;T=1; // Motion and Exposure Value
28 for u=1:M
29     for v=1:N
30         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
31     end
32 end
33
34 n=h.*H;//Multiply the Original Spectrum with the
    Degradation Function.
35 Image_filter=abs(iffn(n));
36 Image_filter=mat2gray(Image_filter)
37 figure,ShowImage(Image_filter,'Filtered Image');
38 title('Motion Blure Image','color','blue','fontsize'
    ,4);

```

---

check Appendix ?? for dependency:

Ex5\_11.png

### Scilab code Exa 5.11 Inverse Filtering

```

1 //Ex5_11
2 //Inverse Filtering
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
    Woods
9
10 clc;
11 close;
12 clear;

```

```

13 xdel(winsid())//to close all currently open figure(s
    ).
14
15 function [H,H1]=lowpassfilter(type1,M,N,D0,n,k)//
    lowpassfilter is used to filter an image .
16     u=0:(M-1);
17     v=0:(N-1);
18     idx=find(u>M/2);
19     u(idx)=u(idx)-M;
20     idy=find(v>N/2);
21     v(idy)=v(idy)-N;
22     [U,V]=meshgrid(v,u);
23     D=sqrt(U.^2+V.^2); //Distance Calculation
24     D=fftshift(D);
25     for i=1:M
26         for j=1:N
27             H(i,j)=exp(-k.*((i-(M/2))^2+(j-(N/2))^2)
                .^(5/6)); //Atmospheric Degradation
                Function
28         end
29     end
30
31     select type1
32
33     case 'inverse'
34         if argn(2)==4
35             n=1;k=0.0025;
36         end
37         H=H;
38         H1=H;
39
40     case 'butterworth'
41         if argn(2)==4
42             n=1;
43         end
44 //     H1 = (ones(M,N)./(1+(D./D0).^(2*n)));
45     H1=double(D<=D0);
46     H=H.*H1;

```

```

47
48         else
49             disp('Unknown filter type. ')
50         end
51     endfunction
52
53     ////////////////////////////////////// Main Programm
54     //////////////////////////////////////
55     gray=imread('Ex5_11.png');
56     gray=im2double(rgb2gray(gray));
57     figure, ShowImage(gray, 'Gray Image');
58     title('Original Image', 'color', 'blue', 'fontsize', 4);
59     [M,N]=size(gray);
60     h=fft2(gray); //fft2() is used to find 2-Dimensional
61                   Fast Fourier Transform of an matrix
62     in=fftshift(h); //fftshift() is used to rearrange the
63                   fft output, moving the zero frequency to the
64                   center of the spectrum.
65     i=log(1+abs(in));
66     inm=mat2gray(i)
67
68     ////////////////////////////////////// Filtering With Cut-off
69     Frequency 480 //////////////////////////////////////
70     [filt,H1]=lowpassfilter('inverse',M,N,480,1,0.0025);
71                   // Function which generate Filter Mask
72                   Corresponding to Low Frequency
73     //filt_shift=fftshift(filt);
74     //figure, ShowImage(abs(filt), 'Filter Mask');
75     //title('Filter Mask to Specific Cut-Off Frequency')
76     ;
77     n=in./(filt+%eps); //Multiply the Original Spectrum
78                   with the Filter Mask.
79     Image_filter=abs(ifft(fftshift(n)));
80     Image_filter=mat2gray(Image_filter)
81     figure, ShowImage(Image_filter, 'Filtered Image');
82     title('Filtered Image with Full Inverse Filter',

```

```

        color ','blue','fontsize',4);
76
77 ////////////////////////////////////////////////// Filtering With Cut-off
    Frequency 40 //////////////////////////////////////////////////
78 [filt,H1]=lowpassfilter('butterworth',M,N
    ,40,10,0.0025); // Function which generate Filter
    Mask Corresponding to Low Frequency
79 //filt_shift=fftshift(filt);
80 //figure,ShowImage(abs(filt),'Filter Mask');
81 //title('Filter Mask to Specific Cut-Off Frequency')
    ;
82 n=(in.*H1)./(filt+%eps);//Multiply the Original
    Spectrum with the Filter Mask.
83 Image_filter=abs(ifft(fftshift(n)));
84 Image_filter=mat2gray(Image_filter)
85 figure,ShowImage(Image_filter,'Filtered Image');
86 title('Filtered Image with Cut-Off Frequency 40','
    color ','blue','fontsize',4);
87
88 ////////////////////////////////////////////////// Filtering With Cut-off
    Frequency 70 //////////////////////////////////////////////////
89 [filt,H1]=lowpassfilter('butterworth',M,N
    ,70,10,0.0025); // Function which generate Filter
    Mask Corresponding to Low Frequency
90 //filt_shift=fftshift(filt);
91 //figure,ShowImage(abs(filt),'Filter Mask');
92 //title('Filter Mask to Specific Cut-Off Frequency')
    ;
93 n=(in.*H1)./(filt+%eps);//Multiply the Original
    Spectrum with the Filter Mask.
94 Image_filter=abs(ifft(fftshift(n)));
95 Image_filter=mat2gray(Image_filter)
96 figure,ShowImage(Image_filter,'Filtered Image');
97 title('Filtered Image with Cut-Off Frequency 70','
    color ','blue','fontsize',4);
98
99 ////////////////////////////////////////////////// Filtering With Cut-off
    Frequency 100 //////////////////////////////////////////////////

```

```

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

```

---

check Appendix ?? for dependency:

Ex5\_12.png

### Scilab code Exa 5.12 Comparison of Inverse Filtering and Wiener Filtering

```

1 //Ex5_12
2 //Comparison of Inverse Filtering and Wiener
    Filtering
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
    Woods
9
10 clc;
11 close;
12 clear;

```

```

13 xdel(winsid())//to close all currently open figure(s
    ).
14
15 function [H,H1]=lowpassfilter(type1,M,N,D0,n,k)//
    lowpassfilter is used to filter an image .
16     u=0:(M-1);
17     v=0:(N-1);
18     idx=find(u>M/2);
19     u(idx)=u(idx)-M;
20     idy=find(v>N/2);
21     v(idy)=v(idy)-N;
22     [U,V]=meshgrid(v,u);
23     D=sqrt(U.^2+V.^2); //Distance Calculation
24     D=fftshift(D);
25     for i=1:M
26         for j=1:N
27             H(i,j)=exp(-k.*((i-(M/2))^2+(j-(N/2))^2)
                .^(5/6)); //Atmospheric Degradation
                Function
28         end
29     end
30
31     select type1
32
33     case 'inverse'
34         if argn(2)==4
35             n=1;k=0.0025;
36         end
37         H=H;
38         H1=H;
39
40     case 'butterworth'
41         if argn(2)==4
42             n=1;
43         end
44 //     H1 = (ones(M,N)./(1+(D./D0).^(2*n)));
45     H1=double(D<=D0);
46     H=H.*H1;

```

```

47
48         else
49             disp('Unknown filter type. ')
50         end
51     endfunction
52
53     ////////////////////////////////////// Main Programm
54     //////////////////////////////////////
55     gray=imread('Ex5_12.png');
56     gray=im2double(rgb2gray(gray));
57     figure, ShowImage(gray, 'Gray Image');
58     title('Original Image', 'color', 'blue', 'fontsize', 4);
59     [M,N]=size(gray);
60     h=fft2(gray); //fft2() is used to find 2-Dimensional
61     Fast Fourier Transform of an matrix
62     in=fftshift(h); //fftshift() is used to rearrange the
63     fft output, moving the zero frequency to the
64     center of the spectrum.
65     i=log(1+abs(in));
66     inm=mat2gray(i)
67
68     ////////////////////////////////////// Filtering With Cut-off
69     Frequency 480 //////////////////////////////////////
70     [filt,H1]=lowpassfilter('inverse',M,N,480,1,0.0025);
71     // Function which generate Filter Mask
72     Corresponding to Low Frequency
73     // filt_shift=fftshift(filt);
74     //figure, ShowImage(abs(filt), 'Filter Mask');
75     //title('Filter Mask to Specific Cut-Off Frequency')
76     ;
77     n=in./(filt+%eps); //Multiply the Original Spectrum
78     with the Filter Mask.
79     Image_filter=abs(ifft(fftshift(n)));
80     Image_filter=mat2gray(Image_filter)
81     figure, ShowImage(Image_filter, 'Filtered Image');
82     title('Filtered Image with Full Inverse Filter',

```

```

        color ','blue','fontsize',4);
76
77 ////////////////////////////////////////////////// Filtering With Cut-off
    Frequency 40 //////////////////////////////////////////////////
78 [filt,H1]=lowpassfilter('butterworth',M,N
    ,40,10,0.0025); // Function which generate Filter
    Mask Corresponding to Low Frequency
79 //filt_shift=fftshift(filt);
80 //figure,ShowImage(abs(filt),'Filter Mask');
81 //title('Filter Mask to Specific Cut-Off Frequency')
    ;
82 n=(in.*H1)./(filt+%eps); //Multiply the Original
    Spectrum with the Filter Mask.
83 Image_filter=abs(ifft(fftshift(n)));
84 Image_filter=mat2gray(Image_filter)
85 figure,ShowImage(Image_filter,'Filtered Image');
86 title('Filtered Image with Cut-Off Frequency 40','
    color ','blue','fontsize',4);
87
88
89 ////////////////////////////////////////////////// Filtering With Cut-off
    Frequency 40 //////////////////////////////////////////////////
90 [filt,H1]=lowpassfilter('butterworth',M,N
    ,40,10,0.0025); // Function which generate Filter
    Mask Corresponding to Low Frequency
91 //filt_shift=fftshift(filt);
92 //figure,ShowImage(abs(filt),'Filter Mask');
93 //title('Filter Mask to Specific Cut-Off Frequency')
    ;
94 n=(in.*H1)./(((1/filt+%eps).*(filt^2/(filt^2+6)))+
    %eps); // Wiener Filtering.
95 Image_filter=abs(ifft(fftshift(n)));
96 Image_filter=mat2gray(Image_filter)
97 figure,ShowImage(Image_filter,'Filtered Image');
98 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**

```
1 //Ex6_3 :
2 //Intensity Slicing
3
4 // Version : Scilab 5.4.1
5 // Operating System : Window-xp, Window-7
6 //Toolbox: Image Processing Design 8.3.1-1
7 //Toolbox: SIVP 0.5.3.1-2
8 //Reference book name : Digital Image Processing
9 //book author: Rafael C. Gonzalez and Richard E.
   Woods
10
11 clc;
12 close;
13 clear;
14 xdel(winsid())//to close all currently open figure(s
   ).
15 gray=imread("Ex6_3.tif");
```

```

16 [nr nc]=size(gray);
17
18 figure,ShowImage(gray,'Gray Image');
19 title('Original Image');
20 min_image=min(gray); // Find Minimum Intensity value
21 max_image=max(gray); // Find Maximum Intensity value
22
23 color_RED=[0 255 0 0 0 255 255 255]; // RED
    Component Value of the Pseudo Color
24 color_GREEN=[0 0 0 255 255 255 0 255]; // GREEN
    Component Value of the Pseudo Color
25 color_BLUE=[0 255 255 255 0 0 0 255]; // BLUE
    Component Value of the Pseudo Color
26 k=8;
27 Slice_Image=[];
28 for y=1:k // Decide Total No. of Level
29 for i=1:nr
30     for j=1:nc
31         if(gray(i,j)>=((max_image/k)*(y-1)) & gray(i
            ,j)<((max_image/k)*y))
32             Slice_Image(i,j,1)=color_RED(y);
33             Slice_Image(i,j,2)=color_GREEN(y);
34             Slice_Image(i,j,3)=color_BLUE(y);
35         end
36     end
37 end
38 end
39 imshow(Slice_Image);//,'Intensity Slicing');
40 //title('Image After Intensity Slicing');

```

---

check Appendix ?? for dependency:

Ex6\_4.tif

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

```

1 //Ex6_4 :
2 //Use of Color to Highlight Rainfall Levels
3
4 // Version : Scilab 5.4.1
5 // Operating System : Window-xp, Window-7
6 //Toolbox: Image Processing Design 8.3.1-1
7 //Toolbox: SIVP 0.5.3.1-2
8 //Reference book name : Digital Image Processing
9 //book author: Rafael C. Gonzalez and Richard E.
    Woods
10
11 clc;
12 close;
13 clear;
14 xdel(winsid())//to close all currently open figure(s
    ).
15 gray=imread("Ex6_4.tif");
16 gray=imresize(gray,0.25);
17 [nr nc]=size(gray);
18
19 figure,ShowImage(gray,'Gray Image');
20 title('Original Image');
21 min_image=min(gray); // Find Minimum Intensity value
22 max_image=max(gray); // Find Maximum Intensity value
23
24 color_RED=[0 255 0 0 0 255 255 255]; // RED
    Component Value of the Pseudo Color
25 color_GREEN=[0 0 0 255 255 255 0 255]; // GREEN
    Component Value of the Pseudo Color
26 color_BLUE=[0 255 255 255 0 0 0 255]; // BLUE
    Component Value of the Pseudo Color
27 k=8;
28 Slice_Image=[];
29 for y=1:k // Decide Total No. of Level
30 for i=1:nr
31     for j=1:nc
32         if(gray(i,j)>=((max_image/k)*(y-1)) & gray(i
            ,j)<((max_image/k)*y))

```

```

33         Slice_Image(i,j,1)=color_RED(y);
34         Slice_Image(i,j,2)=color_GREEN(y);
35         Slice_Image(i,j,3)=color_BLUE(y);
36     end
37 end
38 end
39 end
40 imshow(Slice_Image);//,'Intensity Slicing ');
41 //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

```

1 //Ex6_5 :
2 //Use of Psedocolor for highlighting Exposives
   Contained in Luggage.
3
4 // Version : Scilab 5.4.1
5 // Operating System : Window-xp, Window-7
6 //Toolbox: Image Processing Design 8.3.1-1
7 //Toolbox: SIVP 0.5.3.1-2
8 //Reference book name : Digital Image Processing
9 //book author: Rafael C. Gonzalez and Richard E.
   Woods
10
11 clc;
12 close;
13 clear;
14 xdel(winsid())//to close all currently open figure(s
   ).
15
16 theta=0:450
17 RED=abs(255*sind(theta));

```

```

18 GREEN=abs(255*sind(theta-40));
19 BLUE=abs(255*sind(theta-80));
20 figure;
21 subplot(311),plot(theta,RED);
22 title('RED Intensity Transformation');
23 subplot(312),plot(theta,GREEN);
24 title('GREEN Intensity Transformation');
25 subplot(313),plot(theta,BLUE);
26 title('BLUE Intensity Transformation');
27
28 gray=rgb2gray(imread("Ex6_5.png"));
29 //gray=imresize(gray,0.25);
30 [nr nc]=size(gray);
31
32 figure,ShowImage(gray,'Gray Image');
33 title('Original Image');
34 //min_image=min(gray); // Find Minimum Intensity
    value
35 //max_image=max(gray); // Find Maximum Intensity
    value
36 //
37 //color_RED=[0 255 0 0 0 255 255 255]; // RED
    Component Value of the Pseudo Color
38 //color_GREEN=[0 0 0 255 255 255 0 255]; // GREEN
    Component Value of the Pseudo Color
39 //color_BLUE=[0 255 255 255 0 0 0 255]; // BLUE
    Component Value of the Pseudo Color
40 //k=8;
41 Slice_Image=[];
42 //for y=1:k // Decide Total No. of Level
43 for i=1:nr
44     for j=1:nc
45         Slice_Image(i,j,1)=RED(gray(i,j));
46         Slice_Image(i,j,2)=GREEN(gray(i,j));
47         Slice_Image(i,j,3)=BLUE(gray(i,j));
48     end
49 end
50 //end

```

```
51 imshow(Slice_Image);//,'Intensity Slicing ');
52 //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

```
1 //Ex6_6 :
2 //Color Coding of Multi Spectral Images.
3
4 // Version : Scilab 5.4.1
5 // Operating System : Window-xp, Window-7
6 //Toolbox: Image Processing Design 8.3.1-1
7 //Toolbox: SIVP 0.5.3.1-2
8 //Reference book name : Digital Image Processing
9 //book author: Rafael C. Gonzalez and Richard E.
   Woods
10
11 clc;
12 close;
13 clear;
14 xdel(winsid())//to close all currently open figure(s
   ).
15 gray1=imresize(imread("Ex6_6_1.tif"),0.5);
```

```

16 gray2=imresize(imread("Ex6_6_2.tif"),0.5);
17 gray3=imresize(imread("Ex6_6_3.tif"),0.5);
18 gray4=imresize(imread("Ex6_6_4.tif"),0.5);
19
20 figure,ShowImage(gray1,'Gray Image');
21 title('Visible RED Band Component');
22 figure,ShowImage(gray2,'Gray Image');
23 title('Visible GREEN Band Component');
24 figure,ShowImage(gray3,'Gray Image');
25 title('Visible BLUE Band Component');
26 figure,ShowImage(gray4,'Gray Image');
27 title('Near Infrared Band Image');
28
29 temp(:,:,1)=gray1; //Visible RED Band Component
30 temp(:,:,2)=gray2; //Visible GREEN Band Component
31 temp(:,:,3)=gray3; //Visible BLUE Band Component
32 figure,ShowColorImage(temp,'Color Image');
33 title('Color Composite Image');
34
35 temp1(:,:,1)=gray4; //Near Infrared Band Component
36 temp1(:,:,2)=gray2; //Visible GREEN Band Component
37 temp1(:,:,3)=gray3; //Visible BLUE Band Component
38 figure,ShowColorImage(temp1,'Color Image');
39 title('Color Composite Image');

```

---

check Appendix ?? for dependency:

Ex6\_7.tif

### Scilab code Exa 6.7 Computing Color Image Components

```

1 //Ex6_7 :
2 //Computing Color Image Components.
3
4 // Version : Scilab 5.4.1
5 // Operating System : Window-xp, Window-7

```

```

6 //Toolbox: Image Processing Design 8.3.1-1
7 //Toolbox: SIVP 0.5.3.1-2
8 //Reference book name : Digital Image Processing
9 //book author: Rafael C. Gonzalez and Richard E.
    Woods
10
11 clc;
12 close;
13 clear;
14 xdel(winsid())//to close all currently open figure(s
    ).
15 Color=imread("Ex6_7.tif");
16 Color=imresize(Color,0.5);
17 [nr nc]=size(Color);
18
19 figure,ShowColorImage(Color,'Gray Image');
20 title('Original Image');
21
22
23 Slice_Image=[];
24
25 for i=1:nr
26     for j=1:nc
27         Slice_Image(i,j,1)=255-Color(i,j,1);
28         Slice_Image(i,j,2)=255-Color(i,j,2);
29         Slice_Image(i,j,3)=255-Color(i,j,3);
30     end
31 end
32
33 ShowColorImage(Slice_Image,'RGB Image');;
34 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

```
1 //Ex6_9 :
2 //Tonal Transformations.
3
4 // Version : Scilab 5.4.1
5 // Operating System : Window-xp, Window-7
6 //Toolbox: Image Processing Design 8.3.1-1
7 //Toolbox: SIVP 0.5.3.1-2
8 //Reference book name : Digital Image Processing
9 //book author: Rafael C. Gonzalez and Richard E.
   Woods
10
11 clc;
12 close;
13 clear;
14 xdel(winsid())//to close all currently open figure(s
   ).
15
16 //////////////// Tonal Correction for the Flat
   Image ////////////////
17 Color=imread("Ex6_9_1.tif");
18 Color=imresize(Color,0.5);
19 [nr nc]=size(Color);
20 figure,ShowColorImage(Color, 'Gray Image');
21 title('Original Image');
22 D=0:256;
23 D0=155; // Cut-off Number
24 n=2; // Order of Butter Wirth Approximation
25 H1 = 1-ones(1,1)./(1+(D./D0).^(2*n)); // Transfer
   Function (Design from the Butterworth
   Approximation)
```

```

26 figure,plot(H1);
27 title('RGB Intensity Transformation Function');
28 Slice_Image=[];
29 for i=1:nr
30     for j=1:nc
31         Slice_Image(i,j,1)=H1(uint16(Color(i,j,1)
32             )+1);
32         Slice_Image(i,j,2)=H1(uint16(Color(i,j,2)
33             )+1);
33         Slice_Image(i,j,3)=H1(uint16(Color(i,j,3)
34             )+1);
34     end
35 end
36 ShowColorImage(Slice_Image,'RGB Image');
37 title('Tonal Corrected image');
38
39 //////////////// Tonal Correction for the Light
40 Image ////////////////
41 Color=imread('Ex6_9_2.tif');
42 Color=imresize(Color,0.5);
43 [nr nc]=size(Color);
44 figure,ShowColorImage(Color,'Gray Image');
45 title('Original Image');
46 D=0:1/256:1;
47 H2=1*D^3.0; // Transfer Function (Design from the
48 Gamma Function).
49 figure,plot(H2);
50 title('RGB Intensity Transformation Function');
51 Slice_Image=[];
52 for i=1:nr
53     for j=1:nc
54         Slice_Image(i,j,1)=H2(uint16(Color(i,j,1)
55             )+1);
56         Slice_Image(i,j,2)=H2(uint16(Color(i,j,2)
57             )+1);
58         Slice_Image(i,j,3)=H2(uint16(Color(i,j,3)
59             )+1);
60     end
61 end

```

```

56 end
57 ShowColorImage(Slice_Image, 'RGB Image');
58 title('Tonal Corrected image');
59
60 /////////////////////////////////////////////////// Tonal Correction for the Dark
   Image ///////////////////////////////////////////////////
61 Color=imread(" Ex6_9_3.tif");
62 Color=imresize(Color,0.5);
63 [nr nc]=size(Color);
64 figure, ShowColorImage(Color, 'Gray Image');
65 title('Original Image');
66 D=0:1/256:1;
67 H3=1*D^0.35;
68 figure, plot(H3);
69 title('RGB Intensity Transformation Function');
70 Slice_Image=[];
71 for i=1:nr
72     for j=1:nc
73         Slice_Image(i,j,1)=H3(uint16(Color(i,j,1)
74             )+1);
75         Slice_Image(i,j,2)=H3(uint16(Color(i,j,2)
76             )+1);
77         Slice_Image(i,j,3)=H3(uint16(Color(i,j,3)
78             )+1);
79     end
80 end
81 ShowColorImage(Slice_Image, 'RGB Image');
82 title('Tonal Corrected image');

```

---

check Appendix ?? for dependency:

Ex6\_10.tif

### Scilab code Exa 6.10 Color Balancing

1 //Ex6\_10 :

```

2 //Color Balancing.
3
4 // Version : Scilab 5.4.1
5 // Operating System : Window-xp, Window-7
6 //Toolbox: Image Processing Design 8.3.1-1
7 //Toolbox: SIVP 0.5.3.1-2
8 //Reference book name : Digital Image Processing
9 //book author: Rafael C. Gonzalez and Richard E.
    Woods
10
11 clc;
12 close;
13 clear;
14 xdel(winsid())//to close all currently open figure(s
    ).
15
16 //////////////// Tonal Correction for the Flat
    Image ////////////////
17 Color=imread("Ex6_10.tif");
18 Color=imresize(Color,0.25);
19 [nr nc]=size(Color);
20 figure,ShowColorImage(Color,'Gray Image');
21 title('Original Image','color','blue','fontsize',4);
22
23 C=255-Color(:,:,1);
24 M=255-Color(:,:,2);
25 Y=255-Color(:,:,3);
26 //////////////// Color Balance Correction in
    Cyan Component ////////////////
27 D=0:1/256:1;
28 H1=1*D^2.5; // Transfer Function (Design from the
    Gamma Function).
29 H2=1*D^0.5; // Transfer Function (Design from the
    Gamma Function).
30 figure,subplot(211),plot(H1);
31 xlabel('Intensity');
32 ylabel('Magnitude');
33 title('HSI Intensity Transformation Function(Heavy

```

```

        in Cyan)');
34 subplot(212),plot(H2);
35 xlabel('Intensity');
36 ylabel('Magnitude');
37 title('HSI Intensity Transformation Function (Weak
        in Cyan)', 'color', 'blue', 'fontsize',4);
38
39 C_Modify=[];
40 for i=1:nr
41     for j=1:nc
42         C_Modify1(i,j,1)=H1(uint16(C(i,j,1))+1);
43         C_Modify2(i,j,1)=H2(uint16(C(i,j,1))+1);
44     end
45 end
46 Balance_Image1(:,:,1)=C_Modify1;
47 Balance_Image1(:,:,2)=M;
48 Balance_Image1(:,:,3)=Y;
49 figure,ShowColorImage(Balance_Image1,'RGB Image');
50 title('Color Balanced image', 'color', 'blue', '
        fontsize',4);
51
52 Balance_Image2(:,:,1)=C_Modify2;
53 Balance_Image2(:,:,2)=M;
54 Balance_Image2(:,:,3)=Y;
55 figure,ShowColorImage(Balance_Image2,'RGB Image');
56 title('Color Balanced image', 'color', 'blue', '
        fontsize',4);
57
58 /////////////////////////////////// Color Balance Correction in
        Magenta Component ///////////////////////////////////
59 D=0:1/256:1;
60 H1=1*D^2.5; // Transfer Function (Design from the
        Gamma Function).
61 H2=1*D^0.5; // Transfer Function (Design from the
        Gamma Function).
62 figure,subplot(211),plot(H1);
63 xlabel('Intensity');
64 ylabel('Magnitude');

```

```

65 title('HSI Intensity Transformation Function (Heavy
        in Megenta)', 'color', 'blue', 'fontsize', 4);
66 subplot(212), plot(H2);
67 xlabel('Intensity');
68 ylabel('Magnitude');
69 title('HSI Intensity Transformation Function (Weak
        in Megenta)', 'color', 'blue', 'fontsize', 4);
70 for i=1:nr
71     for j=1:nc
72         Y_Modify1(i,j,1)=H1(uint16(Y(i,j,1))+1);
73         Y_Modify2(i,j,1)=H2(uint16(Y(i,j,1))+1);
74     end
75 end
76 Balance_Image1(:,:,1)=255-C;
77 Balance_Image1(:,:,2)=255-M;
78 Balance_Image1(:,:,3)=255-Y_Modify1;
79 figure, ShowColorImage(Balance_Image1, 'RGB Image');
80 title('Color Balanced image', 'color', 'blue', '
        fontsize', 4);
81
82 Balance_Image2(:,:,1)=255-C;
83 Balance_Image2(:,:,2)=255-M;
84 Balance_Image2(:,:,3)=255-Y_Modify2;
85 figure, ShowColorImage(Balance_Image2, 'RGB Image');
86 title('Color Balanced image', 'color', 'blue', '
        fontsize', 4);
87
88 ////////////////////////////////// Color Balance Correction in
        Yellow Component //////////////////////////////////
89 D=0:1/256:1;
90 H1=1*D^2.5; // Transfer Function (Design from the
        Gamma Function).
91 H2=1*D^0.5; // Transfer Function (Design from the
        Gamma Function).
92 figure, subplot(211), plot(H1);
93 xlabel('Intensity');
94 ylabel('Magnitude');
95 title('HSI Intensity Transformation Function (Heavy

```

```

        in Yellow) ', 'color', 'blue', 'fontsize', 4);
96 subplot(212), plot(H2);
97 xlabel('Intensity');
98 ylabel('Magnitude');
99 title('HSI Intensity Transformation Function (Weak
        in Yellow) ', 'color', 'blue', 'fontsize', 4);
100 for i=1:nr
101     for j=1:nc
102         M_Modify1(i,j,1)=H1(uint16(M(i,j,1))+1);
103         M_Modify2(i,j,1)=H2(uint16(M(i,j,1))+1);
104     end
105 end
106 Balance_Image1(:,:,1)=255-C;
107 Balance_Image1(:,:,2)=255-M_Modify1;
108 Balance_Image1(:,:,3)=255-Y;
109 figure, ShowColorImage(Balance_Image1, 'RGB Image');
110 title('Color Balanced image', 'color', 'blue', '
        fontsize', 4);
111
112 Balance_Image2(:,:,1)=255-C;
113 Balance_Image2(:,:,2)=255-M_Modify2;
114 Balance_Image2(:,:,3)=255-Y;
115 figure, ShowColorImage(Balance_Image2, 'RGB Image');
116 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

```

1 //Ex6_11 :
2 //Histogram Equalization in the HSI Color Space
3
4 // Version : Scilab 5.4.1

```

```

5 // Operating System : Window-xp, Window-7
6 //Toolbox: Image Processing Design 8.3.1-1
7 //Toolbox: SIVP 0.5.3.1-2
8 //Reference book name : Digital Image Processing
9 //book author: Rafael C. Gonzalez and Richard E.
    Woods
10
11 clc;
12 close;
13 clear;
14 xdel(winsid())//to close all currently open figure(s
    ).
15
16 //////////////// Tonal Correction for the Flat
    Image ////////////////
17 Color=imread("Ex6_11.tif");
18 Color=imresize(Color,0.5);
19 [nr nc]=size(Color);
20 figure,ShowColorImage(Color,'Gray Image');
21 title('Original Image','color','blue','fontsize',4);
22
23 HSI=rgb2hsv(Color);
24 figure,ShowImage(HSI(:,:,3),'Gray Image');
25 title('Original Image');
26 [count cell]=imhist(HSI(:,:,3));
27 figure,bar(cell,count,0.2);
28
29 [P Q]=size(Color);
30 r=cell'; // Transpose of matrix
31 nk=round(count)'; // Transpose of matrix
32 M=sum(nk);
33 probablity_r=nk/M; // Probablity calculation
34 for i=1:length(r)
35     sum_1=0;
36     for j=1:i
37         sum_1=sum_1+probablity_r(j);
38     end
39     s(i)=max(r)*sum_1;

```



```

40 end
41 s=round(s); // Rounding Approach
42 disp(s);
43 [nr nc]=size(s);
44 temp=s'; // Transpose of matrix
45 for i=1:P // Intensity Replacement in Original
    Image
46     for j=1:Q
47         b(i,j)=temp(double(HSI(i,j,3))+1);
48     end
49 end
50 HSI(:,:,3)=b(:,:,);
51 Color1=hsv2rgb(HSI);
52 figure,ShowColorImage(Color1,'histogram Equalized
    Image');
53 title('histogram Equalized Image','color','blue','
    fontsize',4);

```

---

check Appendix ?? for dependency:

Ex6\_12.tif

### Scilab code Exa 6.12 Color Image Smoothing by Neighbourhood Averaging

```

1 //Ex6_12
2 //Color Image Smoothing by Neighbourhood Averaging.
3
4 // Version : Scilab 5.4.1
5 // Operating System : Window-xp, Window-7
6 //Toolbox: Image Processing Design 8.3.1-1
7 //Toolbox: SIVP 0.5.3.1-2
8 //Reference book name : Digital Image Processing
9 //book author: Rafael C. Gonzalez and Richard E.
    Woods
10
11 clc;

```

```

12 close;
13 clear;
14 xdel(winsid())//to close all currently open figure(s
    ).
15 rgb=imread("Ex6_12.tif");
16 [nr nc]=size(rgb2gray(rgb)); // find the size of
    image
17
18 figure,ShowColorImage(rgb,'Gray Image');
19 title('Original Image','color','blue','fontsize',4);
20
21 R=rgb(:,:,1);//Separation of red component from
    image
22 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.
23 title('Red component separation from original image',
    'color','blue','fontsize',4);//title() is used
    for providing a title to an image.
24 G=rgb(:,:,2);//Separation of green component from
    image
25 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.
26 title('Green component separation from original
    image','color','blue','fontsize',4);//title() is
    used for providing a title to an image.
27 B=rgb(:,:,3);//Separation of blue component from
    image
28 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.
29 title('Blue component separation from original image
    ','color','blue','fontsize',4);//title() is used
    for providing a title to an image.

```

```

30
31 HSI=rgb2hsv(rgb);
32 H=HSI(:,:,1); //Separation of Hue component from
    image
33 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.
34 title('Red component separation from original image',
    'color','blue','fontsize',4);//title() is used
    for providing a title to an image.
35 S=HSI(:,:,2); //Separation of Saturation component
    from image
36 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.
37 title('Green component separation from original
    image','color','blue','fontsize',4);//title() is
    used for providing a title to an image.
38 I=HSI(:,:,3); //Separation of Intensity component
    from image
39 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.
40 title('Blue component separation from original image',
    'color','blue','fontsize',4);//title() is used
    for providing a title to an image.
41
42 mask=fspecial('average',5);
43 Filtered_Image1=imfilter(rgb,mask);
44 figure,ShowColorImage(Filtered_Image1,'Average Color
    image');//ShowColorImage() is used to show color
    image, figure is command to view images in
    separate window.
45 title('RGB image after Smoothing [5*5]','color','
    blue','fontsize',4);//title() is used for

```

```

    providing a title to an image.
46
47
48 HSI(:,:,3)=imfilter(I,mask);
49 Filtered_Image2=hsv2rgb(HSI);
50 figure,ShowColorImage(Filtered_Image2,'Average Color
    image');//ShowColorImage() is used to show color
    image, figure is command to view images in
    separate window.
51 title('RGB image after Smoothing Intensity Component
    [5*5]','color','blue','fontsize',4);//title() is
    used for providing a title to an image.
52 gray1=im2double(rgb2gray(Filtered_Image1));
53 gray2=rgb2gray(Filtered_Image2);
54 difference=gray1-gray2;
55 //difference=imsubtract(rgb2gray(Filtered_Image1),
    rgb2gray(Filtered_Image2));
56 //difference=im2double(Filtered_Image1)-
    Filtered_Image2;
57 figure,ShowImage(difference,'Difference Color image'
    );//ShowColorImage() is used to show color image,
    figure is command to view images in separate
    window.
58 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 Sharpening with the Laplacian

```

1 //Ex6_13
2 //Sharpening with the Laplacian
3 // Version : Scilab 5.4.1

```

```

4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
    Woods
9
10 clc;
11 close;
12 clear;
13 xdel(winsid())//to close all currently open figure(s
    ).
14 rgb=imread("Ex6_13.tif");
15 [nr nc]=size(rgb2gray(rgb)); // find the size of
    image
16 //figure ,ShowColorImage(rgb,'Gray Image');
17 //title('Original Image');
18
19 R=rgb(:,:,1);//Separation of red component from
    image
20 G=rgb(:,:,2);//Separation of green component from
    image
21 B=rgb(:,:,3);//Separation of blue component from
    image
22 mask=fspecial('laplacian'); // Generate laplacian
    mask
23 Filtered_Image1(:,:,1)=imfilter(R,mask);
24 Filtered_Image1(:,:,2)=imfilter(G,mask);
25 Filtered_Image1(:,:,3)=imfilter(B,mask);
26 figure,ShowColorImage(Filtered_Image1,'Average Color
    image');//ShowColorImage() is used to show color
    image, figure is command to view images in
    separate window.
27 title('RGB image after Sharpning','color','blue','
    fontsize',4);//title() is used for providing a
    title to an image.
28
29 HSI=rgb2hsv(rgb);

```

```

30 H=HSI(:,:,1); //Separation of Hue component from
    image
31 S=HSI(:,:,2); //Separation of Saturation component
    from image
32 I=HSI(:,:,3); //Separation of Intensity component
    from image
33 HSI(:,:,3)=imfilter(I,mask);
34 Filtered_Image2=hsv2rgb(HSI); // Convert HSI to RGB
    Image
35 figure,ShowColorImage(Filtered_Image2,'Average Color
    image'); //ShowColorImage() is used to show color
    image, figure is command to view images in
    separate window.
36 title('RGB image after Sharpning Intensity Component
    ','color','blue','fontsize',4); //title() is used
    for providing a title to an image.
37 gray1=im2double(rgb2gray(Filtered_Image1));
38 gray2=rgb2gray(Filtered_Image2);
39 difference=gray1-gray2; // Difference Image
40 figure,ShowImage(difference,'Difference Color image'
    ); //ShowColorImage() is used to show color image,
    figure is command to view images in separate
    window.
41 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

```

1 //Ex6_14
2 //Segmentation in HSI Space
3 // Version : Scilab 5.4.1

```

```

4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
   Woods
9
10 clc;
11 close;
12 clear;
13 xdel(winsid())//to close all currently open figure(s
   ).
14 rgb=imread("Ex6_14.tif");
15 [nr nc]=size(rgb2gray(rgb)); // find the size of
   image
16 //figure ,ShowColorImage(rgb, 'Gray Image');
17 //title('Original Image','color','blue','fontsize
   ',4);
18 //
19 HSI=rgb2hsv(rgb);
20 H=HSI(:, :, 1);//Separation of Hue component from
   image
21 figure,ShowImage(H, 'Gray Image');
22 title('Hue Component','color','blue','fontsize',4);
23 S=HSI(:, :, 2);//Separation of Saturation component
   from image
24 figure,ShowImage(S, 'Saturation Component');
25 title('Saturation Component','color','blue','
   fontsize',4);
26 I=HSI(:, :, 3);//Separation of Intensity component
   from image
27 figure,ShowImage(I, 'Intensity Component');
28 title('Intensity Component','color','blue','fontsize
   ',4);
29
30 S_Max=max(S); // Calculate Maximum Value
31 thresh=0.35;
32 S_threshold=im2bw(S,thresh); // used for

```

```

    Binarization
33 //S_threshold = imcomplement(S_threshold)
34 figure,ShowImage(S_threshold,'Binary Image');
35 title('Binary Saturation Mask','color','blue','
    fontsize',4);
36
37 temp=H.*S_threshold;
38 figure,ShowImage(temp,'Binary Image');
39 title('Binary Saturation Mask with Multiplication','
    color','blue','fontsize',4);
40
41 [count cell]=imhist(temp);
42 figure,bar(cell,count,0.2);
43 title('Histogram','color','blue','fontsize',4);
44 thresh=0.9;
45 temp_threshold=im2bw(temp,thresh);
46 figure,ShowImage(temp_threshold,'Binary Image');
47 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

```

1 //Ex6_16
2 //Edge Detection Vector Space
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
    Woods
9

```



```

10 clc;
11 close;
12 clear;
13 xdel(winsid()) //to close all currently open figure(s
    ).
14 rgb=imread("Ex6_16.tif");
15 [nr nc]=size(rgb2gray(rgb)); // find the size of
    image
16 figure,ShowColorImage(rgb, 'Color Image');
17 title('Original Image', 'color', 'blue', 'fontsize', 4);
18
19 R=rgb(:, :, 1); //Separation of red component from
    image
20 G=rgb(:, :, 2); //Separation of green component from
    image
21 B=rgb(:, :, 3); //Separation of blue component from
    image
22
23 Image_Edge=edge(R, 'canny', 0.18); // Gradient
    Computation by Canny
24 figure,ShowImage(Image_Edge, 'Edge Image');
25 title('Gradient Image', 'color', 'blue', 'fontsize', 4);
26
27 Image_Edge=edge(G, 'canny', 0.17); // Gradient
    Computation by Canny
28 figure,ShowImage(Image_Edge, 'Edge Image');
29 title('Gradient Image', 'color', 'blue', 'fontsize', 4);
30
31 Image_Edge=edge(B, 'canny', 0.19); // Gradient
    Computation by Canny
32 figure,ShowImage(Image_Edge, 'Edge Image');
33 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

```
1 //Ex6_17
2 //Illustration of the effects of converting noisy
  RGB Images to HSI
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
  Woods
9
10 clc;
11 close;
12 clear;
13 xdel(winsid())//to close all currently open figure(s
  ).
14 R=rgb2gray(imread("Ex6_17_R.tif"));
15 G=rgb2gray(imread("Ex6_17_G.tif"));
16 B=rgb2gray(imread("Ex6_17_B.tif"));
17
18 figure,ShowImage(R,'Red Component');
19 title('Red Component','color','blue','fontsize',4);
20 figure,ShowImage(G,'Green Component');
21 title('Green Component','color','blue','fontsize',4)
  ;
22 figure,ShowImage(B,'Blue Component');
23 title('Blue Component','color','blue','fontsize',4);
24
25 rgb(:,:,1)=R; //Merging of Red component from image
```

```

26 rgb(:,:,2)=G; //Merging of Green component from
    image
27 rgb(:,:,3)=B; //Merging of Blue component from image
28
29 figure,ShowColorImage(rgb,'Color Image');
30 title('Color Image','color','blue','fontsize',4);
31
32 HSI=rgb2hsv(rgb);
33 figure,ShowImage(HSI(:,:,1),'Hue Image');
34 title('Hue Component','color','blue','fontsize',4);
35 figure,ShowImage(HSI(:,:,2),'Saturation Image');
36 title('Saturation Component','color','blue','
    fontsize',4);
37 figure,ShowImage(HSI(:,:,3),'Intensity Image');
38 title('Intensity Component','color','blue','fontsize
    ',4);
39
40
41 G=imnoise(G,'salt & pepper',0.05);
42 rgb(:,:,2)=G; //Merging of Green component from
    image
43 figure,ShowColorImage(rgb,'Color Image');
44 title('Color Image with Salt & Pepper Niose in Green
    Component','color','blue','fontsize',4);
45 HSI=rgb2hsv(rgb);
46 figure,ShowImage(HSI(:,:,1),'Hue Image');
47 title('Hue Component','color','blue','fontsize',4);
48 figure,ShowImage(HSI(:,:,2),'Saturation Image');
49 title('Saturation Component','color','blue','
    fontsize',4);
50 figure,ShowImage(HSI(:,:,3),'Intensity Image');
51 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**

```
1 //Ex8_2
2 //Image Entropy Estimation
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
   Woods
9
10 clc;
11 close;
12 clear;
13 xdel(winsid())//to close all currently open figure(s
   ).
14 A=imread("Ex8_2.tif");
15
```

```

16 figure, ShowImage(A, 'Original Image');
17 title('Original Image', 'color', 'blue', 'fontsize', 4);
18 [nr nc]=size(A);
19 [Count Cell]=imhist(A);
20 //figure, bar(Cell, Count);
21 [r c]=find(Count>0);
22 Probablity=Count(r)/(nr*nc); //Probablity
    Calculation
23 //disp(Probablity);
24 Intensity=Cell(r);
25 //disp(Intensity);
26
27 Sum=0;
28 for i=1:length(r)
29     p=Probablity(i);
30     Sum=Sum+(-p*log2(p));
31 end
32
33 disp('Entropy')
34 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

```
1 //Ex9_1
2 //Using Erosion to remove image component
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
  Woods
9
10 clc;
11 close;
12 clear;
13 xdel(winsid())//to close all currently open figure(s
  ).
```

```

14
15 //////////////// Tonal Correction for the Flat
    Image ////////////////
16 Image=imread("Ex9_1.tif");
17 //Color=imresize(Color,0.25);
18 [nr nc]=size(Image);
19 figure,ShowImage(Image,'Gray Image');
20 title('Binary Image of Wire Bond [486*486]','color',
    'blue','fontsize',4);
21 Mask=CreateStructureElement('square',11);
22 Image_Eroide=ErodeImage(Image,Mask);
23 figure,ShowImage(Image_Eroide,'Eroide Image');
24 title('Eroide Image with 11*11 Square Mask','color',
    'blue','fontsize',4);
25
26 Mask=CreateStructureElement('square',15);
27 Image_Eroide=ErodeImage(Image,Mask);
28 figure,ShowImage(Image_Eroide,'Eroide Image');
29 title('Eroide Image with 15*15 Square Mask','color',
    'blue','fontsize',4);
30
31 Mask=CreateStructureElement('square',45);
32 Image_Eroide=ErodeImage(Image,Mask);
33 figure,ShowImage(Image_Eroide,'Eroide Image');
34 title('Eroide 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

```

1 //Ex9_2
2 //An Illustration of Dilation
3 // Version : Scilab 5.4.1

```

```

4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
  Woods
9
10 clc;
11 close;
12 clear;
13 xdel(winsid())//to close all currently open figure(s
  ).
14
15 Image=imread("Ex9_2.tif");
16 //Color=imresize(Color,0.25);
17 [nr nc]=size(Image);
18 figure,ShowImage(Image,'Gray Image');
19 title('Original Image','color','blue','fontsize',4);
20
21 Mask.Width=3;
22 Mask.Height=3;
23 Mask.Data=[%F %T %F;%T %T %T;%F %T %F];
24
25 //Mask=[0 1 0;1 1 1;0 1 0];
26 Image_Eroide=ErodeImage(Image,Mask);
27 figure,ShowImage(Image_Eroide,'Eroide Image');
28 title('Eroide 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

```

1 //Ex9_4

```



```

2 //Use of opening and closing for Morphological
  Filtering
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
  Woods
9
10 clc;
11 close;
12 clear;
13 xdel(winsid())//to close all currently open figure(s
  ).
14
15 Color=imread("Ex9_4.png");
16 Image=imresize(rgb2gray(Color),2,'bicubic');
17 Image=im2bw(Image,0.75);
18 [nr nc]=size(Image);
19 figure,ShowImage(Image,'Gray Image');
20 title('Noisy Image','color','blue','fontsize',4);
21
22 Mask=CreateStructureElement('square',3); // Create
  Structuring Element
23 Image_Eroide=ErodeImage(Image,Mask); // Erosion
  Operation
24 figure,ShowImage(Image_Eroide,'Eriode Image');
25 title('Eriode Image with 3*3 Square Mask','color','
  blue','fontsize',4);
26
27 Image_Open=OpenImage(Image,Mask); // Opening
  Operation
28 figure,ShowImage(Image_Open,'Open Image');
29 title('Opening Image with 3*3 Square Mask','color','
  blue','fontsize',4);
30
31 Image_Dilate=DilateImage(Image_Open,Mask); //

```

```

    Dilusion of Open Image
32 figure,ShowImage(Image_Dilate,'Dilate Image');
33 title('Dilate Image with 3*3 Square Mask','color','
    blue','fontsize',4);
34
35 Image_Close=CloseImage(Image_Dilate,Mask);    //
    Opening Operation
36 figure,ShowImage(Image_Close,'Closing Image');
37 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

```

1 //Ex9_5
2 //Boundary Extraction by Morphological Processing
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
    Woods
9
10 clc;
11 close;
12 clear;
13 xdel(winsid())//to close all currently open figure(s
    ).
14
15 Color=imread("Ex9_5.png");
16 Image=rgb2gray(Color);
17 Image=im2bw(Image,0.75);

```

```

18 [nr nc]=size(Image);
19 figure,ShowImage(Image,'Binary Image');
20 title('Binary Image','color','blue','fontsize',4);
21
22 Mask=CreateStructureElement('square',5); // Create
    Structuring Element
23 Image_Eroide=ErodeImage(Image,Mask); // Erosion
    Operation
24 Image_Boundray=Image-Image_Eroide;
25 //Image_Open=OpenImage(Image,Mask); // Opening
    Operation
26 figure,ShowImage(Image_Boundray,'Boundray Image');
27 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

```

1 //Ex9_7
2 //Using Connected Components to Detect Foreign
    Object in Packaged Food
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
    Woods
9
10 clc;
11 close;
12 clear;

```

```

13 xdel(winsid())//to close all currently open figure(s
    ).
14
15 Color=imread("Ex9_7.png");
16 Image=rgb2gray(Color);
17 //Image=im2bw(Image,0.65);
18 [nr nc]=size(Image);
19 figure,ShowImage(Image,'Binary Image');
20 title('Binary Image','color','blue','fontsize',4);
21
22 Image_Binary=im2bw(Image,0.825); // Binarization
    Process with Specific Threshold
23 figure,ShowImage(Image_Binary,'Binary Image');
24 title('Binary Image','color','blue','fontsize',4);
25
26 Mask=CreateStructureElement('square',3); // Create
    Structuring Element
27 Image_Eroide=ErodeImage(Image_Binary,Mask); //
    Erosion Operation
28 figure,ShowImage(Image_Eroide,'Eroide Image');
29 title('Eroide Image ','color','blue','fontsize',4);
30
31 BlobImage=SearchBlobs(Image_Eroide); // Connected
    Component labelling
32 IsCalculated = CreateFeatureStruct(%f); // Feature
    struct is generated.
33 IsCalculated.PixelList = %t; // The bounding box
    shall be calculated for each blob.
34 BlobStatistics = AnalyzeBlobs(BlobImage,
    IsCalculated);
35
36 Blob_Total=max(BlobImage);
37 Blob_Area=[];
38 for i=1:Blob_Total
39     temp=size(BlobStatistics(i).PixelList);
40     Blob_Area=[Blob_Area temp(1)];
41 end
42

```

```
43 disp('Blob Area', 'Blob Area')
```

---

check Appendix ?? for dependency:

Ex9\_9.png

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

```
1 //Ex9_9
2 // Illustration of Gray Scale Erosion and Dilation
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
   Woods
9
10 clc;
11 close;
12 clear;
13 xdel(winsid())//to close all currently open figure(s
   ).
14
15 function [f]=restoration_filter(v,type,m,n,Q,d)
16     if argn(2) ==2
17         m=7;n=7;Q=1.5;d=10;
18     elseif argn(2) ==5
19         Q=parameter;d=parameter;
20     elseif argn(2) ==4
21         Q=1.5;d=2;
22     else
23         disp('wrong number of inputs');
24     end
25
26     select type
```

```

27
28     case 'median'
29         f=MedianFilter(v,[m n]);
30
31     case 'MIN'
32         size1=m;
33         [nr,nc]=size(v);
34         temp=zeros(nr+2*floor(size1/2),nc+2*floor(
35             size1/2));
36         temp(ceil(size1/2):nr+ceil(size1/2)-1,ceil(
37             size1/2):nc+ceil(size1/2)-1)=v(1:$,1:$);
38         for i=ceil(size1/2):nr+ceil(size1/2)-1
39             for j=ceil(size1/2):nc+ceil(size1/2)-1
40                 t=temp(i-floor(size1/2):1:i+floor(
41                     size1/2),j-floor(size1/2):1:j+
42                     floor(size1/2)) ;
43                 y=gsort(t);
44                 temp2(i-floor(size1/2),j-floor(size1
45                     /2))=min(y);
46             end
47         end
48         f=mat2gray(temp2);
49
50     case 'MAX'
51         size1=m;
52         [nr,nc]=size(v);
53         temp=zeros(nr+2*floor(size1/2),nc+2*floor(
54             size1/2));
55         temp(ceil(size1/2):nr+ceil(size1/2)-1,ceil(
56             size1/2):nc+ceil(size1/2)-1)=v(1:$,1:$);
57         for i=ceil(size1/2):nr+ceil(size1/2)-1
58             for j=ceil(size1/2):nc+ceil(size1/2)-1
59                 t=temp(i-floor(size1/2):1:i+floor(
60                     size1/2),j-floor(size1/2):1:j+
61                     floor(size1/2)) ;
62                 y=gsort(t);
63                 temp2(i-floor(size1/2),j-floor(size1
64                     /2))=max(y);

```

```

55         end
56     end
57     f=mat2gray(temp2);
58
59     case 'Mid_Point '
60     size1=m;
61     [nr,nc]=size(v);
62     temp=zeros(nr+2*floor(size1/2),nc+2*floor(
        size1/2));
63     temp(ceil(size1/2):nr+ceil(size1/2)-1,ceil(
        size1/2):nc+ceil(size1/2)-1)=v(1:$,1:$);
64     for i=ceil(size1/2):nr+ceil(size1/2)-1
65         for j=ceil(size1/2):nc+ceil(size1/2)-1
66             t=temp(i-floor(size1/2):1:i+floor(
                size1/2),j-floor(size1/2):1:j+
                floor(size1/2)) ;
67             y=gsort(t);
68             temp2(i-floor(size1/2),j-floor(size1
                /2))=0.5*(min(y)+max(y));
69         end
70     end
71     f=mat2gray(temp2);
72
73     else
74         disp('Unknown filter type. ')
75     end
76 endfunction
77
78 ////////////////////////////////////////////////////          Main
79     Programm          ////////////////////////////////////
80 a=imread("Ex9_9.png");
81 gray=rgb2gray(a);
82 //gray=im2double(gray);
83 figure,ShowImage(gray,'Gray Image');
84 title('Original X-Ray Image','color','blue','
        fontsize',4);
85 [M,N]=size(gray);

```

```

86
87 ////////////////////////////////////////////////////          MIN
      Filter          ////////////////////////////////////////////////////
88 h=restoration_filter(gray,'MIN',3,3);
89 figure,ShowImage(h,'Recovered Image');
90 title('Erosion using Flat Structuring Element',
      color','blue','fontsize',4);
91
92
93 ////////////////////////////////////////////////////          MAX Filter
      ////////////////////////////////////////////////////
94 h=restoration_filter(gray,'MAX',3,3);
95 figure,ShowImage(h,'Recovered Image');
96 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

```

1 //Ex9_10
2 // Illustration of Gray Scale Opening and Closing
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
      Woods
9
10 clc;
11 close;
12 clear;

```



```

13 xdel(winsid())//to close all currently open figure(s
    ).
14
15 function [f]=restoration_filter(v,type,m,n,Q,d)
16     if argn(2) ==2
17         m=7;n=7;Q=1.5;d=10;
18     elseif argn(2)==5
19         Q=parameter;d=parameter;
20     elseif argn(2)==4
21         Q=1.5;d=2;
22     else
23         disp('wrong number of inputs');
24     end
25
26     select type
27
28     case 'median'
29         f=MedianFilter(v,[m n]);
30
31     case 'MIN'
32         size1=m;
33         [nr,nc]=size(v);
34         temp=zeros(nr+2*floor(size1/2),nc+2*floor(
            size1/2));
35         temp(ceil(size1/2):nr+ceil(size1/2)-1,ceil(
            size1/2):nc+ceil(size1/2)-1)=v(1:$,1:$);
36         for i=ceil(size1/2):nr+ceil(size1/2)-1
37             for j=ceil(size1/2):nc+ceil(size1/2)-1
38                 t=temp(i-floor(size1/2):1:i+floor(
                    size1/2),j-floor(size1/2):1:j+
                    floor(size1/2)) ;
39                 y=gsort(t);
40                 temp2(i-floor(size1/2),j-floor(size1
                    /2))=min(y);
41             end
42         end
43         f=mat2gray(temp2);
44

```

```

45     case 'MAX'
46         size1=m;
47         [nr,nc]=size(v);
48         temp=zeros(nr+2*floor(size1/2),nc+2*floor(
49             size1/2));
50         temp(ceil(size1/2):nr+ceil(size1/2)-1,ceil(
51             size1/2):nc+ceil(size1/2)-1)=v(1:$,1:$);
52         for i=ceil(size1/2):nr+ceil(size1/2)-1
53             for j=ceil(size1/2):nc+ceil(size1/2)-1
54                 t=temp(i-floor(size1/2):1:i+floor(
55                     size1/2),j-floor(size1/2):1:j+
56                     floor(size1/2)) ;
57                 y=gsort(t);
58                 temp2(i-floor(size1/2),j-floor(size1
59                     /2))=max(y);
60             end
61         end
62         f=mat2gray(temp2);
63
64         case 'Mid_Point'
65             size1=m;
66             [nr,nc]=size(v);
67             temp=zeros(nr+2*floor(size1/2),nc+2*floor(
68                 size1/2));
69             temp(ceil(size1/2):nr+ceil(size1/2)-1,ceil(
70                 size1/2):nc+ceil(size1/2)-1)=v(1:$,1:$);
71             for i=ceil(size1/2):nr+ceil(size1/2)-1
72                 for j=ceil(size1/2):nc+ceil(size1/2)-1
73                     t=temp(i-floor(size1/2):1:i+floor(
74                         size1/2),j-floor(size1/2):1:j+
75                         floor(size1/2)) ;
76                     y=gsort(t);
77                     temp2(i-floor(size1/2),j-floor(size1
78                         /2))=0.5*(min(y)+max(y));
79                 end
80             end
81             f=mat2gray(temp2);
82

```

```

73         else
74         disp('Unknownfiltertype. ')
75     end
76 endfunction
77
78 /////////////////////////////////////////////////// Main
79     Programm ///////////////////////////////////////////////////
80 a=imread("Ex9_10.png");
81 gray=rgb2gray(a);
82 //gray=im2double(gray);
83 figure,ShowImage(gray,'Gray Image');
84 title('Original X-Ray Image','color','blue','
85     fontsize',4);
86 [M,N]=size(gray);
87 /////////////////////////////////////////////////// Gray Scale Opening
88 ///////////////////////////////////////////////////
89 h=restoration_filter(restoration_filter(gray,'MIN'
90     ,3,3),'MAX',3,3);
91 figure,ShowImage(h,'Recovered Image');
92 title('Opening using Flat Structuring Element','
93     color','blue','fontsize',4);
94
95 /////////////////////////////////////////////////// Gray Scale Closing
96 ///////////////////////////////////////////////////
97 h=restoration_filter(restoration_filter(gray,'MAX'
98     ,3,3),'MIN',3,3);
99 figure,ShowImage(h,'Recovered Image');
100 title('Closing using Flat Structuring Element','
101     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

```
1 //Ex10_1
2 // Detection of Isolated Point in an Image
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
   Woods
9
10 clc;
11 close;
12 clear;
13 xdel(winsid())//to close all currently open figure(s
   ).
14
15 a=imread("Ex10_1.tif");
```

```

16 //gray=rgb2gray(a);
17 //a=im2double(a);
18 figure,ShowImage(a,'Gray Image');
19 title('Original X-Ray Image','color','blue','
    fontsize',4);
20 [M,N]=size(a);
21
22 Mask=[1 1 1;1 -8 1;1 1 1];
23 Filtered_Image=imfilter(a,Mask);
24 figure,ShowImage(Filtered_Image,'Filter Image');
25 title('Original X-Ray Image','color','blue','
    fontsize',4);
26
27 thresh=uint8(229.5);
28 disp(thresh);
29 image=im2bw(Filtered_Image,0.996);
30 figure,ShowImage(image,'Filter Image');
31 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

```

1 //Ex10_2
2 // Using the Laplacian for the Detection
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
    Woods
9

```

```

10 clc;
11 close;
12 clear;
13 xdel(winsid())//to close all currently open figure(s
    ).
14
15 a=imread("Ex10_2.tif");
16 figure,ShowImage(a,'Gray Image');
17 title('Wire Bond Template Image','color','blue','
    fontSize',4);
18 [M,N]=size(a);
19
20 Mask=[1 1 1;1 -8 1;1 1 1]; // Mask for the
    Lapalacian
21 Filtered_Image=imfilter(a,Mask); // Filtering the
    Original Image with the Mask
22 figure,ShowImage(Filtered_Image,'Filter Image');
23 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

```

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

```

Thresholded gradient Image

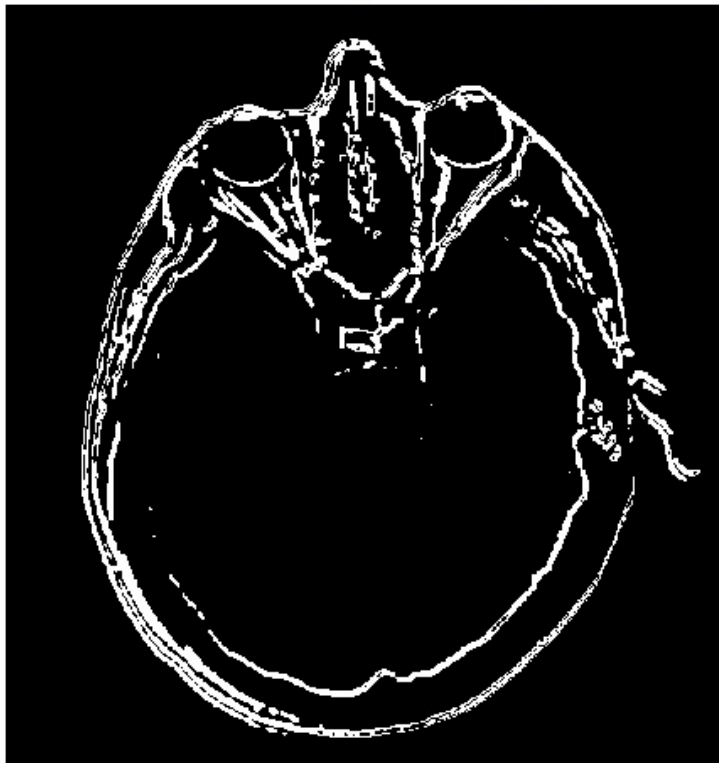


Figure 10.1: Using the Laplacian for the Detection

Canny Edge Detected Image

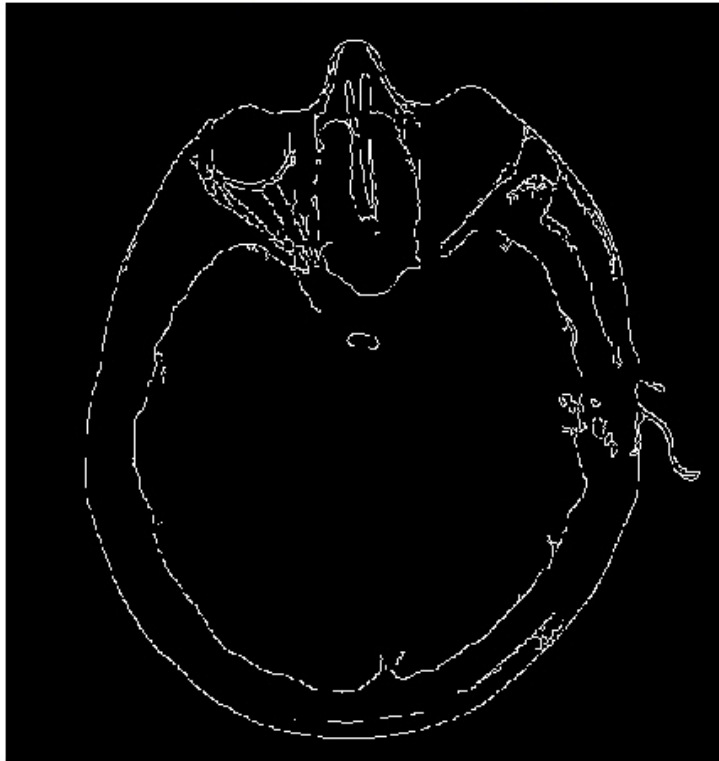


Figure 10.2: Using the Laplacian for the Detection



```

7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
  Woods
9
10 clc;
11 close;
12 clear;
13 xdel(winsid())//to close all currently open figure(s
  ).
14
15 a=imread("Ex10_2.tif");
16 figure,ShowImage(a,'Gray Image');
17 title('Wire Bond Template Image','color','blue','
  fontSize',4);
18 [M,N]=size(a);
19
20 Mask_Diagonal=[2 -1 -1;-1 2 -1;-1 -1 2]; // Mask
  for the +45 Line Detetion
21 Filtered_Image=imfilter(a,Mask_Diagonal); //
  Filtering the Original Image with the Mask
22 figure,ShowImage(Filtered_Image,'Filter Image');
23 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

```

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

```

```

6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
    Woods
9
10 clc;
11 close;
12 clear;
13 xdel(winsid())//to close all currently open figure(s
    ).
14
15 a=imread("Ex10_4.tif");
16 a1=im2double(a);
17 figure,ShowImage(a,'Gray Image');
18 title('Original Image','color','blue','fontsize',4);
19 [M,N]=size(a);
20 first_order=zeros(M,N);
21 second_order=zeros(M,N);
22
23 for i=2:M-1
24     for j=2:N-1
25         first_order(i,j)=a(i,j+1)-a(i,j);
26         second_order(i,j)=double(a1(i,j+1)+a1(i,j-1)
            -(2*a1(i,j)));
27     end
28 end
29
30 figure,ShowImage(double(first_order),'First Order
    Difference Image');
31 title('First Order Difference Image','color','blue',
    'fontsize',4);
32
33 forward_count=0;
34 reverse_count=0;
35 for j= 2:N-1 // Finding First Zero Crossing Point
36     if(second_order(5,j)==0 & second_order(5,j
        +1)>0)
37         forward_count=forward_count+1;

```

```

38         if(forward_count==1)
39             ther1=second_order(5,j+1);
40             break;
41         end
42     end
43 end
44
45 for j= N-1:-1:2 // Finding Last Zero Crossing Point
46     if(second_order(5,j)==0 & second_order(5,j
47         -1)<0)
48         reverse_count=reverse_count+1;
49         if(reverse_count==1)
50             ther2=second_order(5,j-1);
51             break;
52         end
53     end
54
55 for i=1:M // Removing unwanted Intensity range
56     for j=1:N
57         if(second_order(i,j)==ther1)
58             second_order1(i,j)=255;
59         else if(second_order(i,j)==ther2)
60             second_order1(i,j)=-255;
61         else
62             second_order1(i,j)=128;
63         end
64     end
65 end
66 end
67
68 figure,ShowImage(second_order1,'Second Order
69     Difference Image');
70 title('Second Order Difference Image','color','blue',
71     'fontsize',4);
72
73 t=a(5,1:N);
74 t1=first_order(5,1:N);

```

```

73 t2=second_order1(5,1:N);
74 figure;
75 subplot(311);
76 plot(1:length(t),t);//
77 title('Intensity Profile','color','blue','fontsize'
      ,4);
78 subplot(312);
79 mtlb_axis([1,N,0,1.5]);
80 plot(1:length(t1),t1);//
81 title('Intensity Profile of First order Derivative',
      'color','blue','fontsize',4);
82 subplot(313);
83 plot(1:length(t2),t2);//
84 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

```

1 //Ex10_6
2 // Illustration of the 2 D Gradient Magnitude and
  Angle
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
  Woods
9
10 clc;
11 close;
12 clear;

```

```

13 xdel(winsid())//to close all currently open figure(s
    ).
14
15 a=imread("Ex10_6.tif");
16 //a=im2double(a);
17 figure,ShowImage(a,'Gray Image');
18 title('Original Image','color','blue','fontsize',4);
19 [M,N]=size(a);
20
21 Mask=[-1 -2 -1;0 0 0;1 2 1]; // Mask for the Sobel
22 GradientX_Image=imfilter(a,Mask); // Filtering the
    Original Image with the Mask
23 figure,ShowImage(GradientX_Image,'Filter Image');
24 title('Sobel X-direction Gradient Image','color','
    blue','fontsize',4);
25
26 Mask=[-1 0 1;-2 0 2;-1 0 1]; // Mask for the Sobel
27 GradientY_Image=imfilter(a,Mask); // Filtering the
    Original Image with the Mask
28 figure,ShowImage(GradientY_Image,'Filter Image');
29 title('Sobel Y-direction Gradient Image','color','
    blue','fontsize',4);
30
31 Gradient=GradientX_Image+GradientY_Image;
32 figure,ShowImage(Gradient,'Filter Image');
33 title('Sobel X+Y Gradient Image','color','blue','
    fontsize',4);
34
35 //Alpha=atan(double(GradientY_Image),double(
    GradientX_Image));
36 //figure,ShowImage(Alpha,'Angle Image');
37 //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

```
1 //Ex10_7
2 // Illustration of the Marr-Hildreth Edge Detection
  Methods
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
  Woods
9
10 clc;
11 close;
12 clear;
13 xdel(winsid())//to close all currently open figure(s
  ).
14
15 a=imread("Ex10_7.tif");
16 a=im2double(a);
17 figure,ShowImage(a,'Gray Image');
18 title('Original Image','color','blue','fontsize',4);
19 [M,N]=size(a);
20 sigma=4;
21 for i=1:25
22     for j=1:25
23         Mask(i,j)=[(i^2+j^2-(2*sigma^2))/sigma^4]*
                exp(-(i^2+j^2)/(2*sigma^2)); // Mask
                Generation
24     end
25 end
26
27 Filter_Image=imfilter(a,Mask); // Filtering the
  Original Image with the Mask
28 figure,ShowImage(Filter_Image,'Filter Image');
29 title('Laplacian of gaussian Image','color','blue','
  fontsize',4);
```

```

30 b=zeros(M,N);
31 temp=Filter_Image;
32 for i=2:M-1 // Zero Crossing Detection
33     for j=2:N-1
34         //temp=[Filter_Image(i-1:i+1,j-1:j+1)];
35         if((temp(i-1,j-1)>0 & temp(i+1,j+1)<0) | (
36             temp(i-1,j-1)<0 & temp(i+1,j+1)>0)) then
37             b(i,j)=255;
38         else if ((temp(i-1,j+1)>0 & temp(i+1,j-1)<0)
39             | (temp(i-1,j+1)<0 & temp(i+1,j-1)>0))
40             then
41                 b(i,j)=255;
42         else if ((temp(i,j+1)>0 & temp(i,j-1)<0) | (
43             temp(i,j+1)<0 & temp(i,j-1)>0)) then
44                 b(i,j)=255;
45         else if ((temp(i-1,j)>0 & temp(i+1,j)<0) | (
46             temp(i-1,j)<0 & temp(i+1,j)>0)) then
47                 b(i,j)=255;
48         end
49     end
50 end
51 figure,ShowImage(b,'Zero Crossing Image');
52 title('Zero Crossing Detected Image','color','blue',
53     'fontsize',4);

```

---

check Appendix ?? for dependency:

Ex10\_8.tif

Scilab code Exa 10.8 Illustration of the Canny Edge Detection Methods

1 //Ex10-8

```

2 // Illustration of the Canny Edge Detection Methods
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
   Woods
9
10 clc;
11 close;
12 clear;
13 xdel(winsid())//to close all currently open figure(s
   ).
14
15 a=imread("Ex10_8.tif");
16 //a=im2double(a);
17 figure,ShowImage(a,'Gray Image');
18 title('Original Image','color','blue','fontsize',4);
19 [M,N]=size(a);
20 ////////// Threshlded Gradient of Smoothed
   Image //////////
21 a1=imfilter(a,fspecial('average',5));
22 Mask=[-1 -2 -1;0 0 0;1 2 1]; // Mask for the Sobel
23 GradientX_Image=imfilter(a1,Mask); // Filtering the
   Original Image with the Mask
24 //figure,ShowImage(GradientX_Image,'Filter Image');
25 //title('Sobel X-direction Gradient Image','color','
   blue','fontsize',4);
26
27 Mask=[-1 0 1;-2 0 2;-1 0 1]; // Mask for the Sobel
28 GradientY_Image=imfilter(a1,Mask); // Filtering the
   Original Image with the Mask
29 //figure,ShowImage(GradientY_Image,'Filter Image');
30 //title('Sobel Y-direction Gradient Image','color','
   blue','fontsize',4);
31
32 Gradient=GradientX_Image+GradientY_Image;

```



```

33 //figure ,ShowImage(Gradient , 'Filter Image');
34 //title('Sobel X+Y Gradient Image','color','blue','
    fontsize',4);
35
36 th=84; // 33% of the Maximum Value in Gradient
    Image
37 [row col]=find(Gradient>84);
38 Gradient_Thresh=zeros(M,N);
39 for i=1:length(row)
40     Gradient_Thresh(row(i),col(i))=255;
41 end
42 figure,ShowImage(Gradient_Thresh,'Filter Image');
43 title('Thresholded gradient Image','color','blue','
    fontsize',4);
44
45 /////////////////////////////////////////////////// Marr-Hildreth Edge Detection
    //////////////////////////////////////
46 a=im2double(a);
47 sigma=4;
48 for i=1:25
49     for j=1:25
50         Mask(i,j)=[(i^2+j^2-(2*sigma^2))/sigma^4]*
            exp(-(i^2+j^2)/(2*sigma^2)); // Mask
            Generation
51     end
52 end
53
54 Filter_Image=imfilter(a,Mask); // Filtering the
    Original Image with the Mask
55 //figure,ShowImage(Filter_Image,'Filter Image');
56 //title('Laplacian of gaussian Image','color','blue
    ','fontsize',4);
57 b=zeros(M,N);
58 temp=Filter_Image;
59 for i=2:M-1 // Zero Crossing Detection
60     for j=2:N-1
61         //temp=[Filter_Image(i-1:i+1,j-1:j+1)];
62         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
63         b(i,j)=255;
64     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
65         b(i,j)=255;
66     else if ((temp(i,j+1)>0 & temp(i,j-1)<0) | (
        temp(i,j+1)<0 & temp(i,j-1)>0)) then
67         b(i,j)=255;
68     else if ((temp(i-1,j)>0 & temp(i+1,j)<0) | (
        temp(i,j+1)<0 & temp(i,j-1)>0)) then
69         b(i,j)=255;
70     end
71     end
72     end
73     end
74 end
75 end
76 figure,ShowImage(b,'Zero Crossing Image');
77 title('Marr-Hildreth Edge Detected Image','color','
    blue','fontsize',4);
78
79 ////////////////////////////////// Canny Edge Deteced Image
    //////////////////////////////////
80 a=imread("Ex10_8.tif");
81 E=edge(a,'canny',[0.15 0.60]);
82 figure,ShowImage(E,'Canny Image');
83 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

```

1 //Ex10_9
2 // Another illustration of the three principal Edge
   Detection Methods
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
   Woods
9
10 clc;
11 close;
12 clear;
13 xdel(winsid())//to close all currently open figure(s
   ).
14
15 a=imread("Ex10_9.tif");
16 //a=im2double(a);
17 figure,ShowImage(a,'Gray Image');
18 title('Original Image','color','blue','fontsize',4);
19 [M,N]=size(a);
20 ////////////////////////////////////////////////// Threshlded Gradient of Smoothed
   Image //////////////////////////////////////
21 a1=imfilter(a,fspecial('average',5));
22 Mask=[-1 -2 -1;0 0 0;1 2 1]; // Mask for the Sobel
23 GradientX_Image=imfilter(a1,Mask); // Filtering the
   Original Image with the Mask
24 //figure,ShowImage(GradientX_Image,'Filter Image');
25 //title('Sobel X-direction Gradient Image','color','
   blue','fontsize',4);
26
27 Mask=[-1 0 1;-2 0 2;-1 0 1]; // Mask for the Sobel
28 GradientY_Image=imfilter(a1,Mask); // Filtering the
   Original Image with the Mask
29 //figure,ShowImage(GradientY_Image,'Filter Image');
30 //title('Sobel Y-direction Gradient Image','color','
   blue','fontsize',4);

```

```

31
32 Gradient=GradientX_Image+GradientY_Image;
33 //figure ,ShowImage(Gradient , 'Filter Image');
34 //title('Sobel X+Y Gradient Image','color','blue','
    fontsize',4);
35
36 th=84; // 33% of the Maximum Value in Gradient
    Image
37 [row col]=find(Gradient>84);
38 Gradient_Thresh=zeros(M,N);
39 for i=1:length(row)
40     Gradient_Thresh(row(i),col(i))=255;
41 end
42 figure,ShowImage(Gradient_Thresh,'Filter Image');
43 title('Thresholded gradient Image','color','blue','
    fontsize',4);
44
45 //////////////////////////////////// Marr-Hildreth Edge Detection
    ////////////////////////////////////
46 a=im2double(a);
47 sigma=3;
48 for i=1:19
49     for j=1:19
50         Mask(i,j)=[(i^2+j^2-(2*sigma^2))/sigma^4]*
            exp(-(i^2+j^2)/(2*sigma^2)); // Mask
            Generation
51     end
52 end
53
54 Filter_Image=imfilter(a,Mask); // Filtering the
    Original Image with the Mask
55 //figure,ShowImage(Filter_Image,'Filter Image');
56 //title('Laplacian of gaussian Image','color','blue
    ','fontsize',4);
57 b=zeros(M,N);
58 temp=Filter_Image;
59 th=0.0021;
60 for i=2:M-1 // Zero Crossing Detection

```

```

61     for j=2:N-1
62         //temp=[Filter_Image(i-1:i+1,j-1:j+1)];
63         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
64             b(i,j)=255;
65         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
66             b(i,j)=255;
67         else if ((temp(i,j+1)>th & temp(i,j-1)<th) |
            (temp(i,j+1)<th & temp(i,j-1)>th)) then
68             b(i,j)=255;
69         else if ((temp(i-1,j)>th & temp(i+1,j)<th) |
            (temp(i,j+1)<th & temp(i,j-1)>th)) then
70             b(i,j)=255;
71         end
72     end
73 end
74 end
75 end
76 end
77 figure,ShowImage(b,'Zero Crossing Image');
78 title('Marr-Hildreth Edge Detected Image','color','
    blue','fontsize',4);
79
80 ////////////////////////////////////////////////// Canny Edge Deteced Image
    //////////////////////////////////////
81 a=imread("Ex10_9.tif");
82 E=edge(a,'canny',[0.05 0.95]);
83 figure,ShowImage(E,'Canny Image');
84 title('Canny Edge Detected Image','color','blue','
    fontsize',4);

```

---

check Appendix ?? for dependency:

Ex10\_15.tif

### Scilab code Exa 10.15 Global Thresholding

```
1 //Ex10_15
2 // Global Thresholding
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
   Woods
9
10 clc;
11 close;
12 clear;
13 xdel(winsid())//to close all currently open figure(s
   ).
14
15 a=imread("Ex10_15.tif");
16 a1=im2double(a);
17 figure,ShowImage(a,'Gray Image');
18 title('Noisy Finger Print','color','blue','fontsize'
   ,4);
19 [M,N]=size(a);
20
21 [count cell]=imhist(a);
22 figure,plot2d3(cell,count);
23 title('Histogram','color','blue','fontsize',4);
24
25 b=im2bw(a1,0.495);
26 figure,ShowImage(b,'Binary Image');
27 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

```
1 //Ex10_16
2 // Optimum Global Thresholding using Otsu's Method
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
   Woods
9
10 clc;
11 close;
12 clear;
13 xdel(winsid())//to close all currently open figure(s
   ).
14
15 a=imread("Ex10_16.tif");
16 a1=im2double(a);
17 figure,ShowImage(a,'Gray Image');
18 title('Original Image','color','blue','fontsize',4);
19 [M,N]=size(a);
20
21 [count cell]=imhist(a);
22 figure,plot2d3(cell,count);
23 title('Histogram','color','blue','fontsize',4);
24
25 ////////////////////////////////////////////////////////////////// Global Threshold
   Approach //////////////////////////////////////////////////////////////////
26 th_Global=iterthresh(a1);
27 b1=im2bw(a1,th_Global);
```

```

28 figure,ShowImage(b1,'Binary Image');
29 title('Segmented Result Using Global Thresholding
        Algorithm','color','blue','fontsize',4);
30
31 //////////////////////////////////////// Otsu Method
32 ////////////////////////////////////////
33 normal_hist=count/(M*N);
34 Sum=0;
35 cumu_mean=0;
36 for k=1:max(cell)+1
37     Sum=Sum+normal_hist(k);
38     P1(k)=Sum;
39     cumu_mean=cumu_mean+(k*normal_hist(k));
40     m(k)=cumu_mean;
41     Mg=cumu_mean;
42     sigma_B(k)=(((Mg*P1(k))-m(k))^2)/(%eps+(P1(k)
        *(1-P1(k))));
43 end
44
45 th_Otsu=find(sigma_B==max(sigma_B))+10;
46 b2=im2bw(a1,(th_Otsu/255));
47 figure,ShowImage(b2,'Binary Image');
48 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



Segmented Result Using Global Thresholding Algorithm

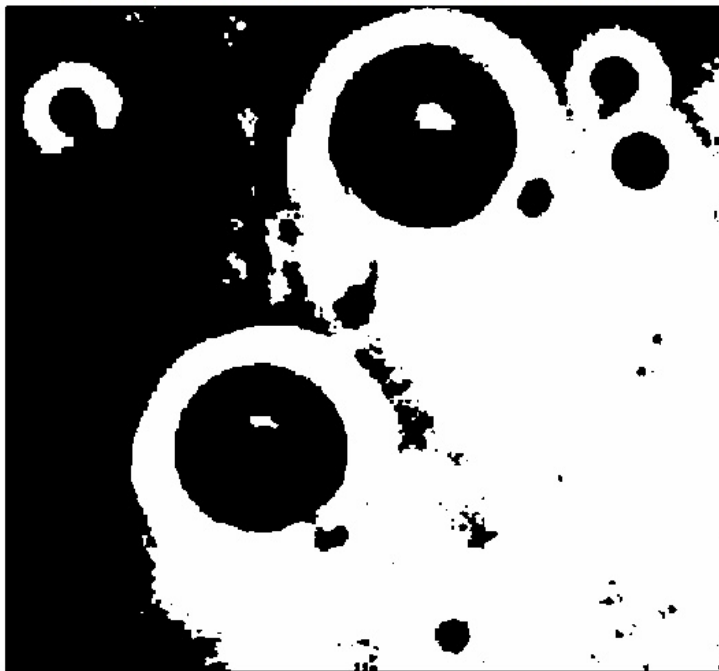


Figure 10.3: Optimum Global Thresholding using Otsu Method

Segmented Result Using Otsu Thresholding Algorithm



Figure 10.4: Optimum Global Thresholding using Otsu Method

```

1 //Ex10_18
2 // Using Edge Information Based on the Laplacian to
   Improve Global Thresholding
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
   Woods
9
10 clc;
11 close;
12 clear;
13 xdel(winsid())//to close all currently open figure(s
   ).
14
15 a=imread("Ex10_18.tif");
16 a1=im2double(a);
17 figure,ShowImage(a,'Gray Image');
18 title('Original Image','color','blue','fontsize',4);
19 [M,N]=size(a);
20
21 [count cell]=imhist(a);
22 figure,plot2d3(cell,count);
23 title('Histogram','color','blue','fontsize',4);
24
25 ////////////////////////////////////// Otsu Method
   //////////////////////////////////////
26
27 normal_hist=count/(M*N);
28 Sum=0;
29 cumu_mean=0;
30 for k=1:max(cell)+1
31     Sum=Sum+normal_hist(k);
32     P1(k)=Sum;
33     cumu_mean=cumu_mean+(k*normal_hist(k));
34     m(k)=cumu_mean;

```

```

35     Mg=cumu_mean;
36     sigma_B(k)=(((Mg*P1(k))-m(k))^2)/(eps+(P1(k)
        *(1-P1(k))));
37 end
38
39 th_Otsu=42;           //find(sigma_B==max(sigma_B));
40 b2=im2bw(a1,(th_Otsu/255));
41 figure,ShowImage(b2,'Binary Image');
42 title('Segmented Result Using Otsu Thresholding
        Algorithm','color','blue','fontsize',4);
43
44
45 mask=fspecial('laplacian');
46 c=abs(imfilter(a,mask));
47 figure,ShowImage(mat2gray(c),'Binary Image');
48 title('Laplacian Image','color','blue','fontsize',4)
        ;
49
50 //d=c.*a1;
51 //[count cell]=imhist(d);
52 //figure,plot2d3(cell,count);
53 //title('Histogram','color','blue','fontsize',4);
54
55 th_Otsu=115;         //find(sigma_B==max(sigma_B));
56 b3=im2bw(a1,(th_Otsu/255));
57 figure,ShowImage(b3,'Binary Image');
58 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

1 //Ex10\_19

```

2 // Multipal Global Thresholding
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
   Woods
9
10 clc;
11 close;
12 clear;
13 xdel(winsid())//to close all currently open figure(s
   ).
14
15 a=imread("Ex10_19.tif");
16 a1=im2double(a);
17 figure,ShowImage(a,'Gray Image');
18 title('Original Image','color','blue','fontsize',4);
19 [M,N]=size(a);
20
21 [count cell]=imhist(a);
22 figure,plot2d3(cell,count);
23 title('Histogram','color','blue','fontsize',4);
24
25 //////////////////////////////////// Otsu Method
   ////////////////////////////////////
26
27 normal_hist=count/(M*N);
28 Sum=0;
29 cumu_mean=0;
30 for k=1:max(cell)+1
31     Sum=Sum+normal_hist(k);
32     P1(k)=Sum;
33     cumu_mean=cumu_mean+(k*normal_hist(k));
34     m(k)=cumu_mean;
35     Mg=cumu_mean;
36     sigma_B(k)=(((Mg*P1(k))-m(k))^2)/(%eps+(P1(k)

```

```

        *(1-P1(k)))));
37 end
38
39 th_otsu=[80 177];          // find(sigma_B==max(sigma_B
    ));
40 b2=im2bw(a1,(th_otsu(1)/255));
41 b3=im2bw(a1,(th_otsu(2)/255));
42 b4=b2+b3;
43 figure,ShowImage(mat2gray(b4),'Binary Image');
44 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

```

1 //Ex10_20
2 // Variable Thresholding Via Image Partitioning
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
    Woods
9
10 clc;
11 close;
12 clear;
13 xdel(winsid())//to close all currently open figure(s
    ).
14
15
16 function th1=otsu(count,cell)

```

```

17 normal_hist=count/(M/2*N/3);
18 Sum=0;
19 cumu_mean=0;
20 for k=1:max(cell)+1
21     Sum=Sum+normal_hist(k);
22     P1(k)=Sum;
23     cumu_mean=cumu_mean+(k*normal_hist(k));
24     m(k)=cumu_mean;
25     Mg=cumu_mean;
26     sigma_B(k)=(((Mg*P1(k))-m(k))^2)/(%eps+(P1(k)
        *(1-P1(k))));
27 end
28
29 th1=find(sigma_B==max(sigma_B));
30 endfunction
31
32
33 a=imread("Ex10_20.tif");
34 a=imresize(a,[650 813],'bicubic');
35 a1=im2double(a);
36 figure,ShowImage(a,'Gray Image');
37 title('Original Image','color','blue','fontsize',4);
38 [M,N]=size(a);
39
40 [count cell]=imhist(a);
41 figure,plot2d3(cell,count);
42 title('Histogram','color','blue','fontsize',4);
43
44 ////////////////////////////////// Iterative
    Thresholding //////////////////////////////////
45 thr = iterthresh(a1);
46 b1=im2bw(a1,thr);
47 figure,ShowImage(b1,'Gray Image');
48 title('Segmentation Using Iterative Global
    Thresholding','color','blue','fontsize',4);
49
50 ////////////////////////////////// Otsu Method
    //////////////////////////////////

```

```

51
52 normal_hist=count/(M*N);
53 Sum=0;
54 cumu_mean=0;
55 for k=1:max(cell)+1
56     Sum=Sum+normal_hist(k);
57     P1(k)=Sum;
58     cumu_mean=cumu_mean+(k*normal_hist(k));
59     m(k)=cumu_mean;
60     Mg=cumu_mean;
61     sigma_B(k)=(((Mg*P1(k))-m(k))^2)/(%eps+(P1(k)
        *(1-P1(k))));
62 end
63
64 th=find(sigma_B==max(sigma_B));
65 b2=im2bw(a1,(th/255));
66 figure,ShowImage(mat2gray(b2),'Binary Image');
67 title('Image Segmented using Otsu Threshold','color'
        , 'blue', 'fontsize',4);
68
69 //////////////////////////////////// Otsu with Image
        Partitioning ////////////////////////////////////
70 count=[];
71 cell=[];
72 z=1;
73 th2=[40 50 70 40 50 70];
74 for i=1:M/2:M
75     for j=1:N/3:N
76         [count cell]=imhist(a(i:(i-1)+(M/2),j:(j-1)
            +(N/3)));
77         th1=otsu(count,cell);
78         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));
79         z=z+1;
80     end
81 end
82
83 figure,ShowImage(mat2gray(b3),'Binary Image');

```



```
84 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

```
1 //Ex10_22  
2 // Document Thresholding Using Moving Averages  
3 // Version : Scilab 5.4.1  
4 // Operating System : Window-xp, Window-7  
5 //Toolbox: Image Processing Design 8.3.1-1  
6 //Toolbox: SIVP 0.5.3.1-2  
7 //Reference book name : Digital Image Processing  
8 //book author: Rafael C. Gonzalez and Richard E.  
    Woods  
9  
10 clc;  
11 close;  
12 clear;  
13 xdel(winsid())//to close all currently open figure(s  
    ).  
14  
15 a=imread("Ex10_22.tif");  
16 a1=im2double(a);  
17 figure,ShowImage(a1,'Gray Image');  
18 title('Original Image','color','blue','fontsize',4);  
19 [M,N]=size(a);  
20  
21 Threshold = CalculateOtsuThreshold(a1);  
22 Thresh_Image=im2bw(a1,Threshold);  
23 figure,ShowImage(Thresh_Image,'Binary Image');  
24 title('Thresholded Image with Otsu Method','color','  
    blue','fontsize',4);
```

```

25
26
27 mask=zeros(1,20);
28 array=[];
29 for i=1:M
30     if(pmodulo(i,2)==0)
31         array=[array mtlb_fliplr(a1(i,:))];
32     else
33         array=[array a1(i,:)];
34     end
35 end
36 disp('first');
37 for i=1:length(array)
38     for j=1:length(mask)
39         if(j<length(mask)) then
40             mask(j)=mask(j+1);
41         else
42             mask(j)=array(i);
43         end
44     end
45     avg(1,i)=sum(mask)/length(mask);
46 end
47 disp('Second');
48 len=1;
49 for i=1:M
50     if(pmodulo(i,2)==0)
51         b(i,:)=avg(len:len+N-1);
52         len=len+N;
53     else
54         b(i,:)=avg(len:len+N-1);
55         len=len+N;
56     end
57 end
58 disp('Last');
59
60 b=0.5*b;
61 for i=1:M
62     for j=1:N

```

Thresholded Image with Otsu Method

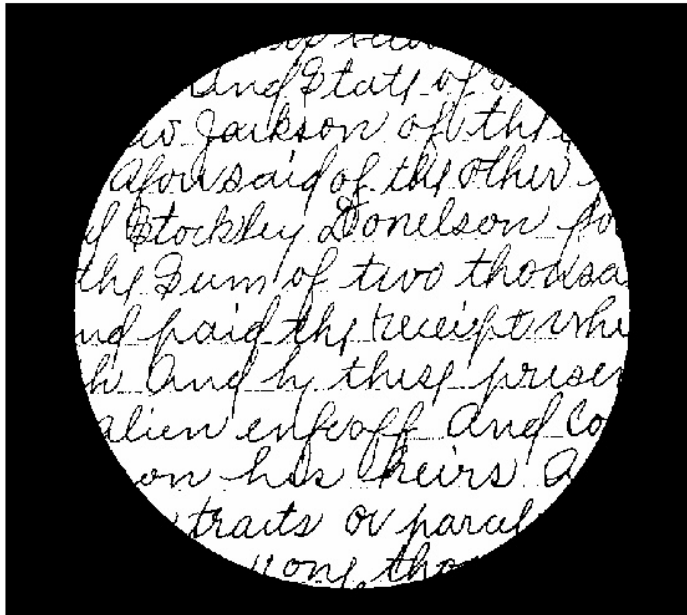


Figure 10.5: Document Thresholding Using Moving Averages

```
63         if(b(i,j)>a1(i,j)) then
64             c(i,j)=0;
65         else
66             c(i,j)=1;
67         end
68     end
69 end
70 figure,ShowImage(c,'Binary Image');
71 title('Local Thresholding Using Moving Average',
       'color','blue','fontsize',4);
```

---

### Scilab code Exa 10.23 Segmentation by Region Growing

```
1 //Ex10_23
2 // Segmentation by Region Growing
3 // Version : Scilab 5.4.1
4 // Operating System : Window-xp, Window-7
5 //Toolbox: Image Processing Design 8.3.1-1
6 //Toolbox: SIVP 0.5.3.1-2
7 //Reference book name : Digital Image Processing
8 //book author: Rafael C. Gonzalez and Richard E.
   Woods
9
10 clc;
11 close;
12 clear;
13 xdel(winsid())//to close all currently open figure(s
   ).
14 a=imread("Ex10_23.tif");
15 a1=im2double(a);
16 figure,ShowImage(a1,'Gray Image');
17 title('Original Image','color','blue','fontsize',4);
18 [M,N]=size(a);
19
20 [count cell]=imhist(a);
21 figure,plot2d3(cell,count);
22 title('Histogram','color','blue','fontsize',4);
23
24 th=254/255;
25 Thresh_Image=im2bw(a1,th);
26 figure,ShowImage(Thresh_Image,'Gray Image');
27 title('Thresholded Image','color','blue','fontsize',
   ,4);
28 for i=1:M
29     for j=1:N
```

```

30         if(Thresh_Image(i,j)) then
31             Thresh_Image1(i,j)=1;
32         else
33             Thresh_Image1(i,j)=0;
34         end
35     end
36 end
37
38 BlobImage = SearchBlobs(Thresh_Image); // Connected
    Component Labelling
39 IsCalculated = CreateFeatureStruct(%f); // Feature
    struct is generated.
40 IsCalculated.Centroid = %t; // The bounding box
    shall be calculated for each blob.
41 BlobStatistics = AnalyzeBlobs(BlobImage,
    IsCalculated);
42 Seed_Image=zeros(M,N);
43 for i=1:max(BlobImage) // Centroid Calculation
44     Seed_Image(BlobStatistics(i).Centroid(1,2),
        BlobStatistics(i).Centroid(1,1))=1;
45 end
46 figure,ShowImage(Seed_Image,'Gray Image');
47 title('Seed Point Image','color','blue','fontsize',
    ,4);
48
49 Diff=uint8(255*imsubtract(a1,Thresh_Image1));
50 figure,ShowImage(Diff,'Gray Image');
51 title('Seed Point Image','color','blue','fontsize',
    ,4);
52 [count cell]=imhist(Diff);
53 figure,plot2d3(cell,count);
54 title('Histogram','color','blue','fontsize',4);
55
56 Thresh_Image2=uint8(zeros(M,N))
57 for i=1:M
58     for j=1:N
59         if(Diff(i,j)<=68) then
60             Thresh_Image2(i,j)=255;

```

```
61         else if(Diff(i,j)>68 & Diff(i,j)<=165) then
62             Thresh_Image2(i,j)=125;
63         else
64             Thresh_Image2(i,j)=0;
65         end
66     end
67 end
68 end
69 figure,ShowImage(uint8(Thresh_Image2),'Gray Image');
70 title('Seed Point Image','color','blue','fontsize'
       ,4);
```

---

Seed Point Image

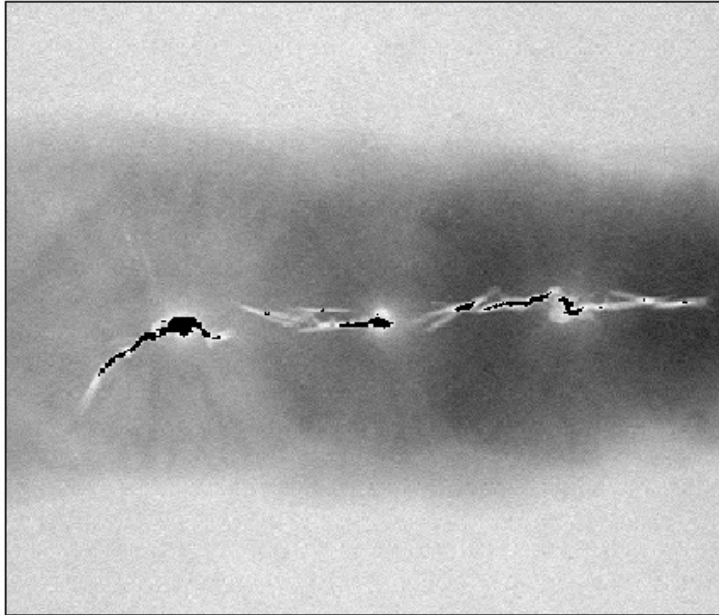


Figure 10.6: Segmentation by Region Growing

Seed Point Image

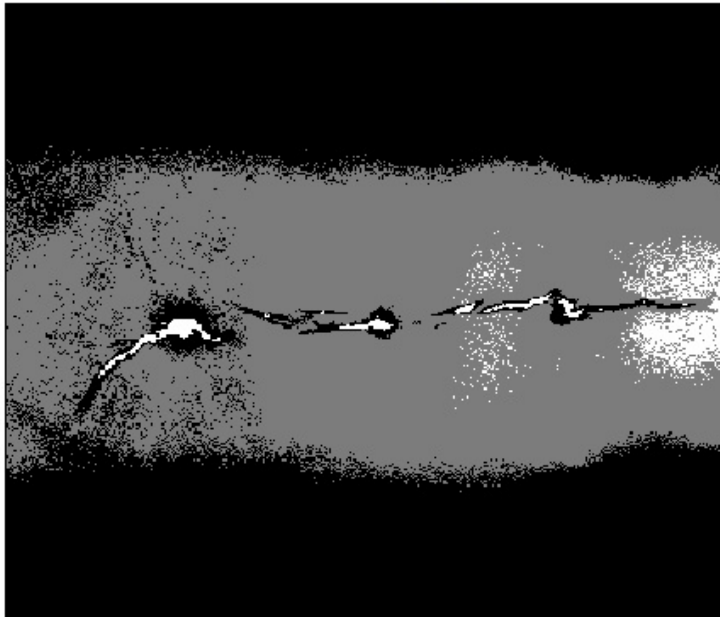
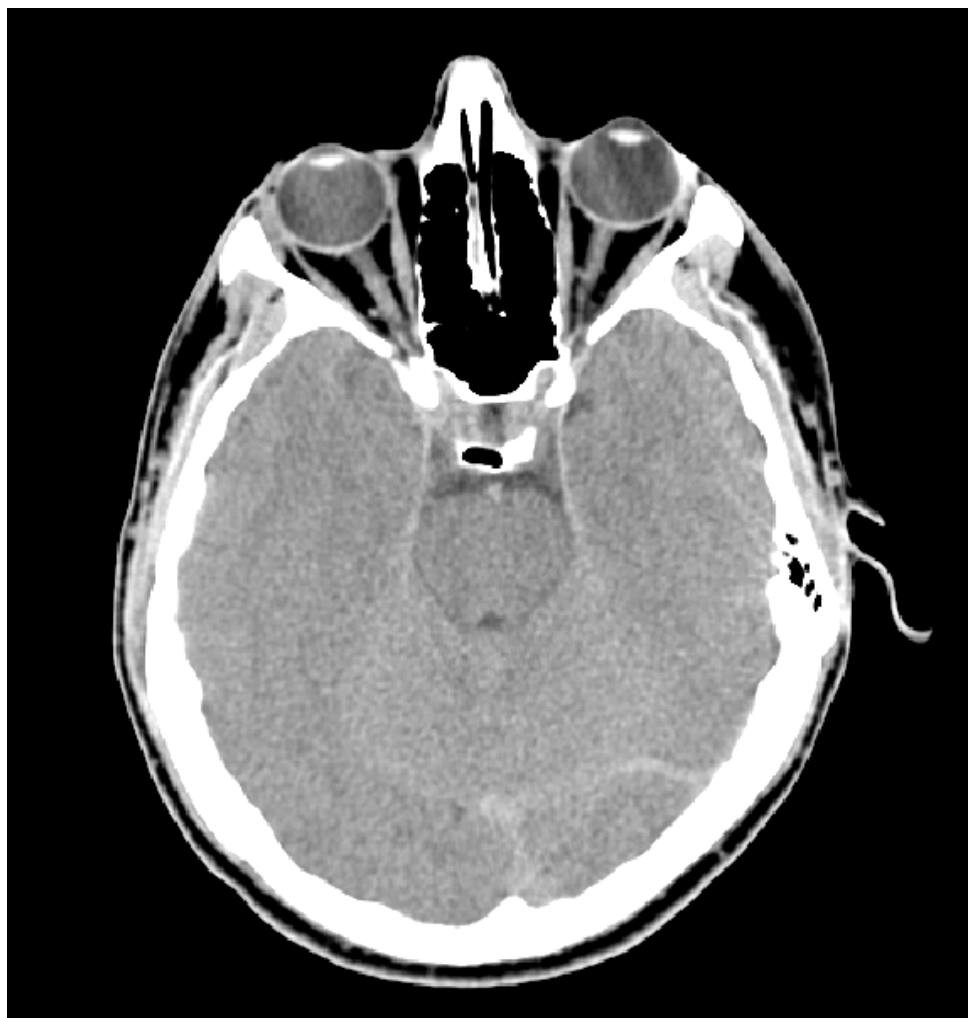


Figure 10.7: Segmentation by Region Growing



# 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**

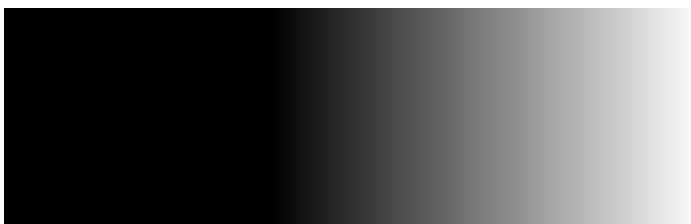
Illustration of the Marr Hildreth Edge Detection Methods

---



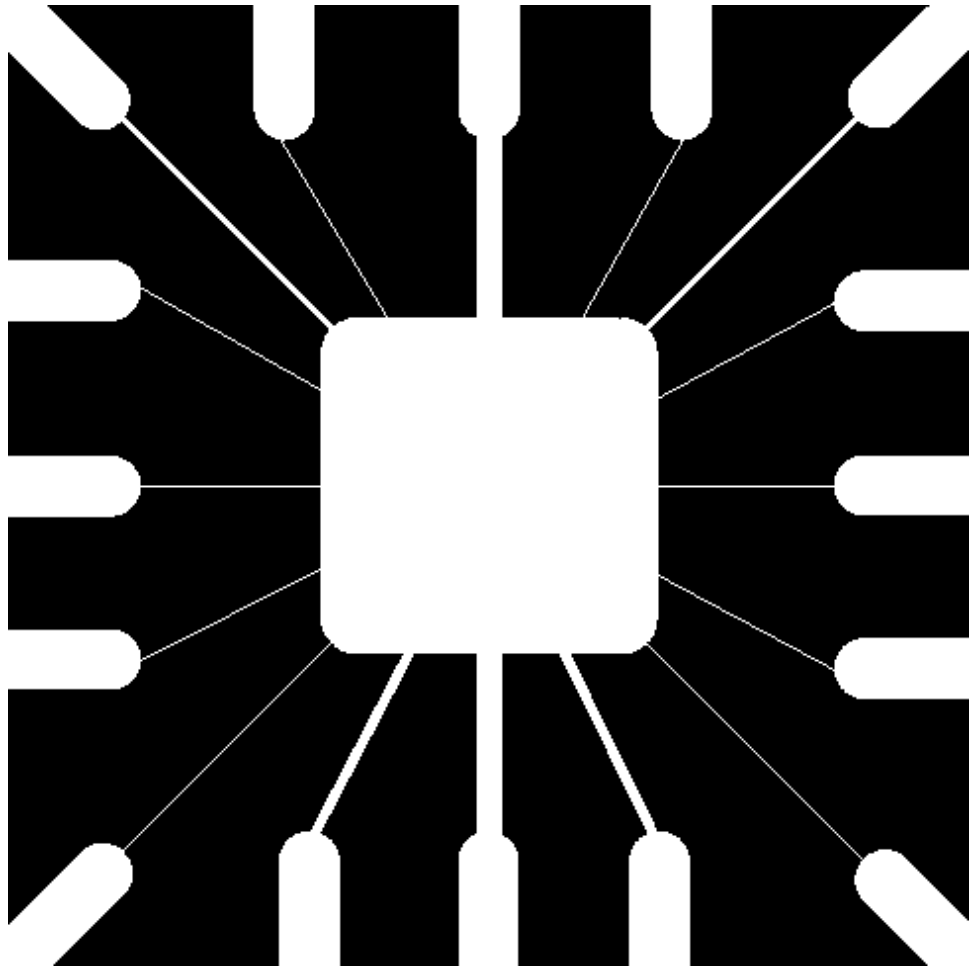
**Scilab code AP 4**

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

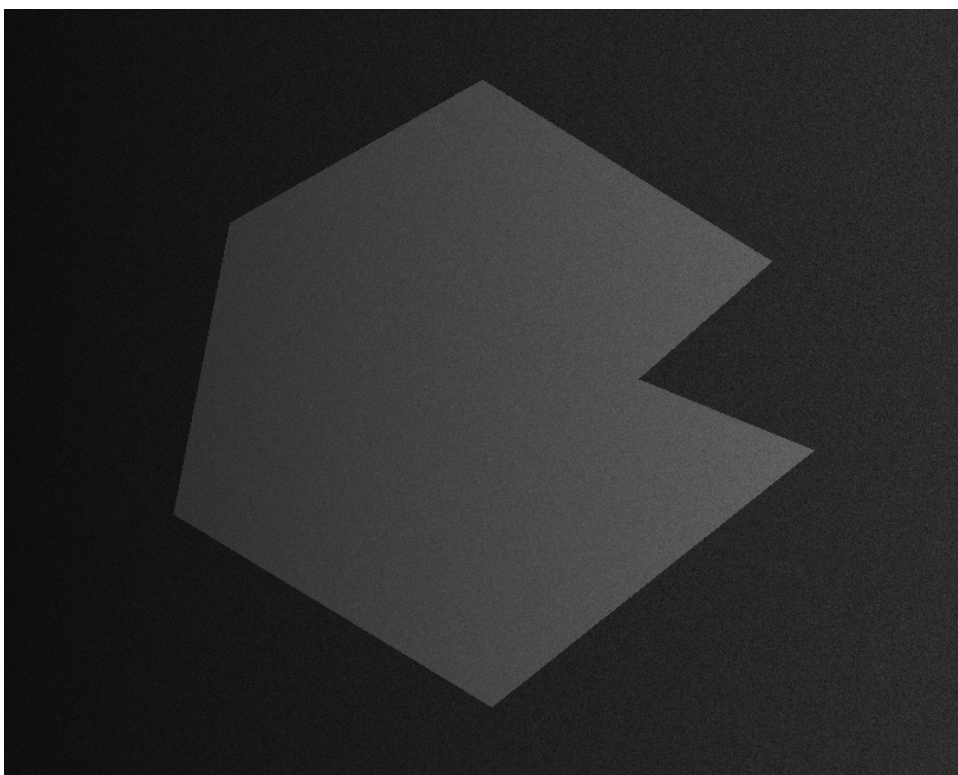
---

and ninety six between Stockley  
of Knox And State of Tennessee  
Andrew Jackson of the County  
State 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  
self alien enfeof and confir  
Jackson his heirs and a  
certain tracts or parcels of La  
and acres one thousand acre  
more or less being and his

Scilab code AP 7

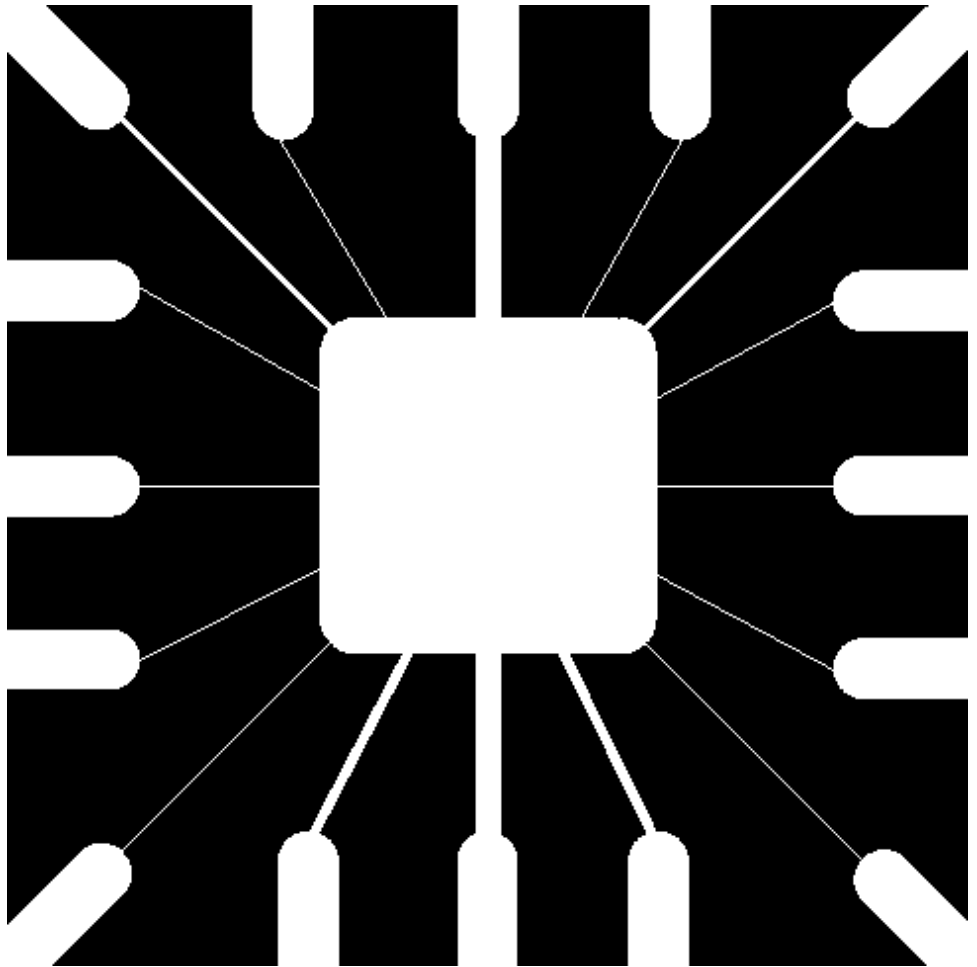
Document Thresholding Using Moving Averages

---



**Scilab code AP 8**

Variable Thresholding Via Image Partitioning



Scilab code AP 9  
Using the Laplacian for the Detection

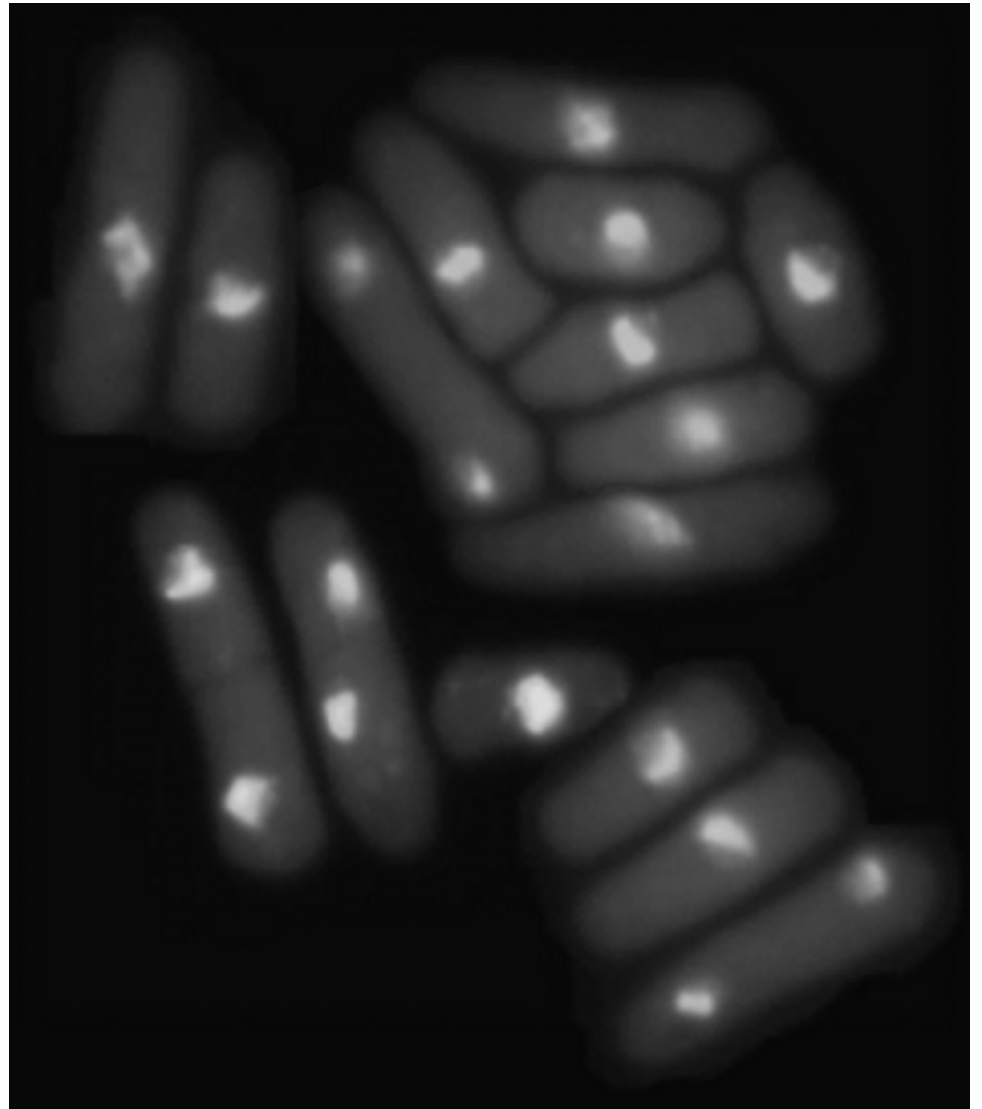
---





**Scilab code AP 10**  
Multiple Global Thresholding

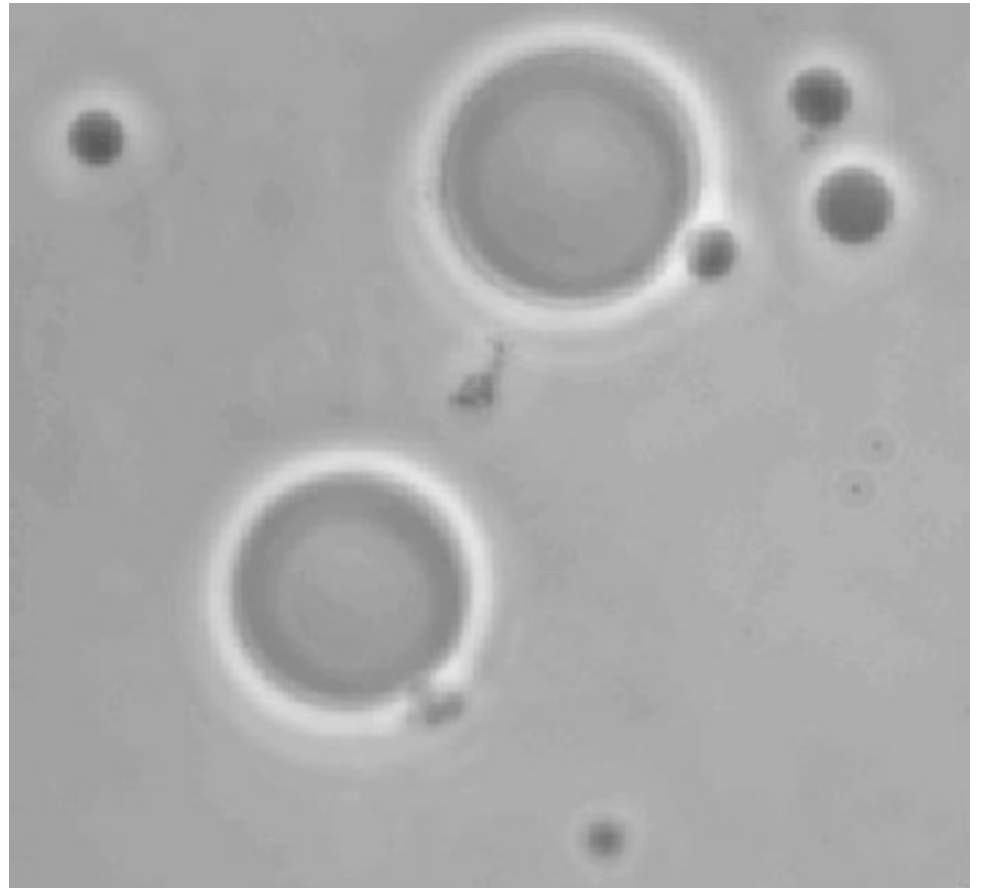
---



**Scilab code AP 11**

Using Edge Information Based on the Laplacian to Improve Global Thresholding

---



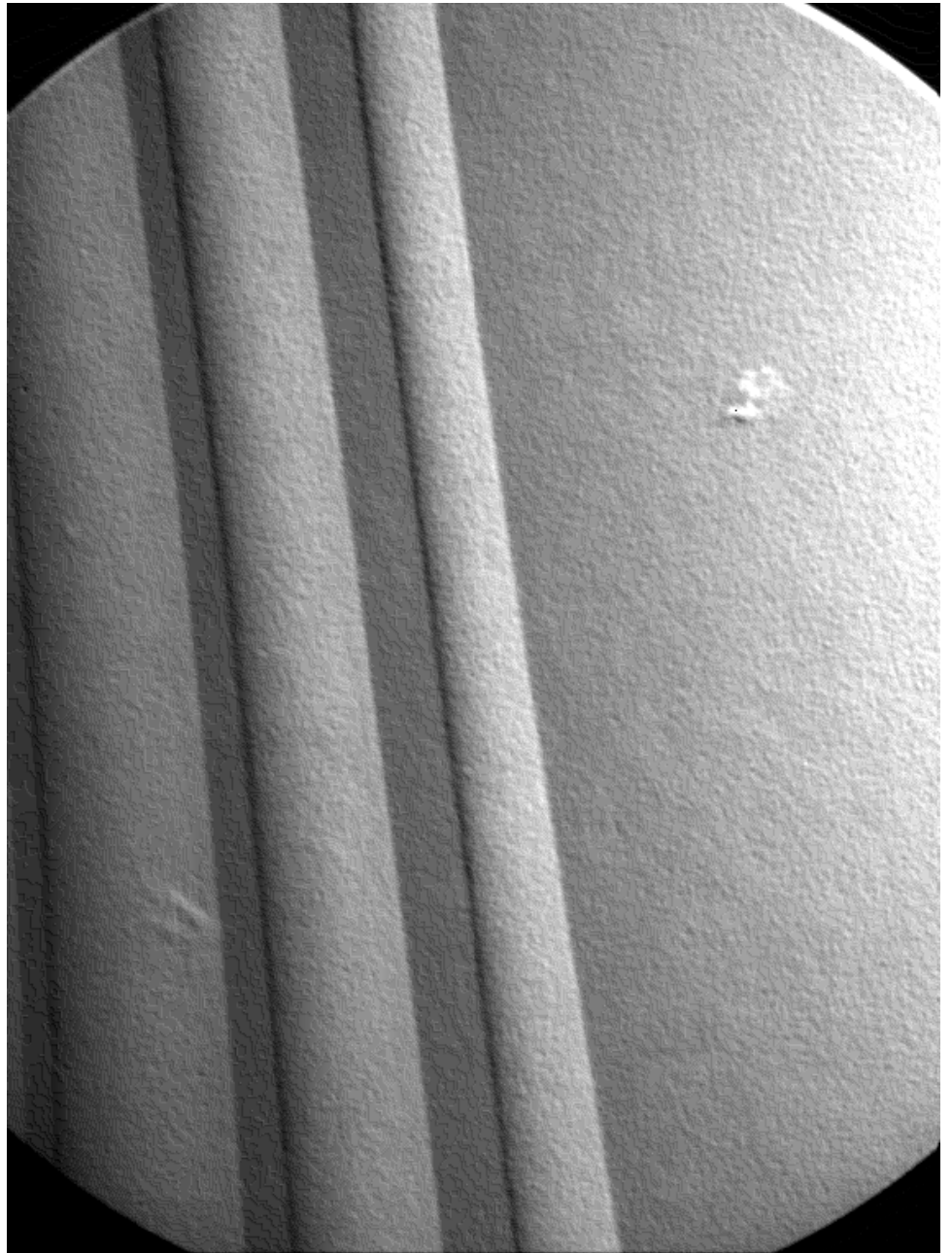
**Scilab code AP 12**

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

Historically, certain computer programs were written using only two digits rather than four to define the applicable year. Accordingly, the company's software may recognize a date using "00" as 1900 rather than the year 2000.

Scilab code AP 19  
An Illustration of Dilation



Boundary Extraction by Morphological Processing



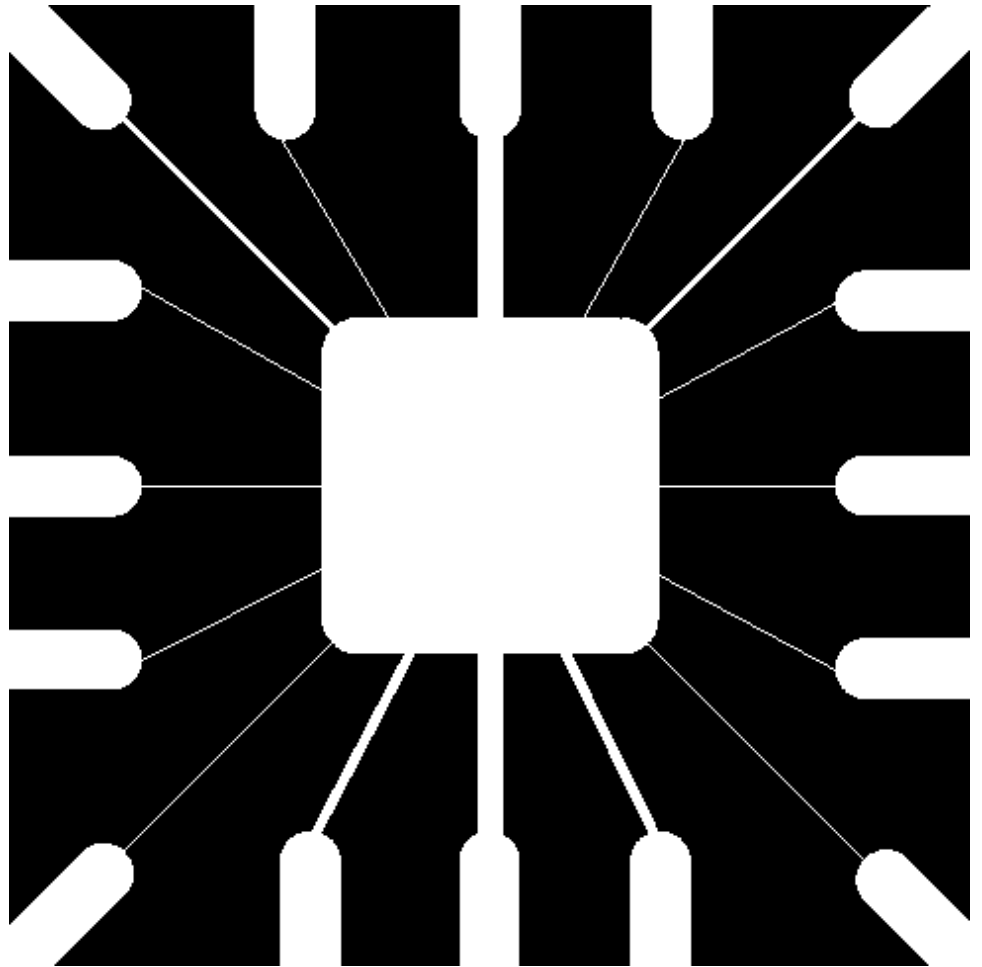


Use

of opening and closing for Morphological Filtering

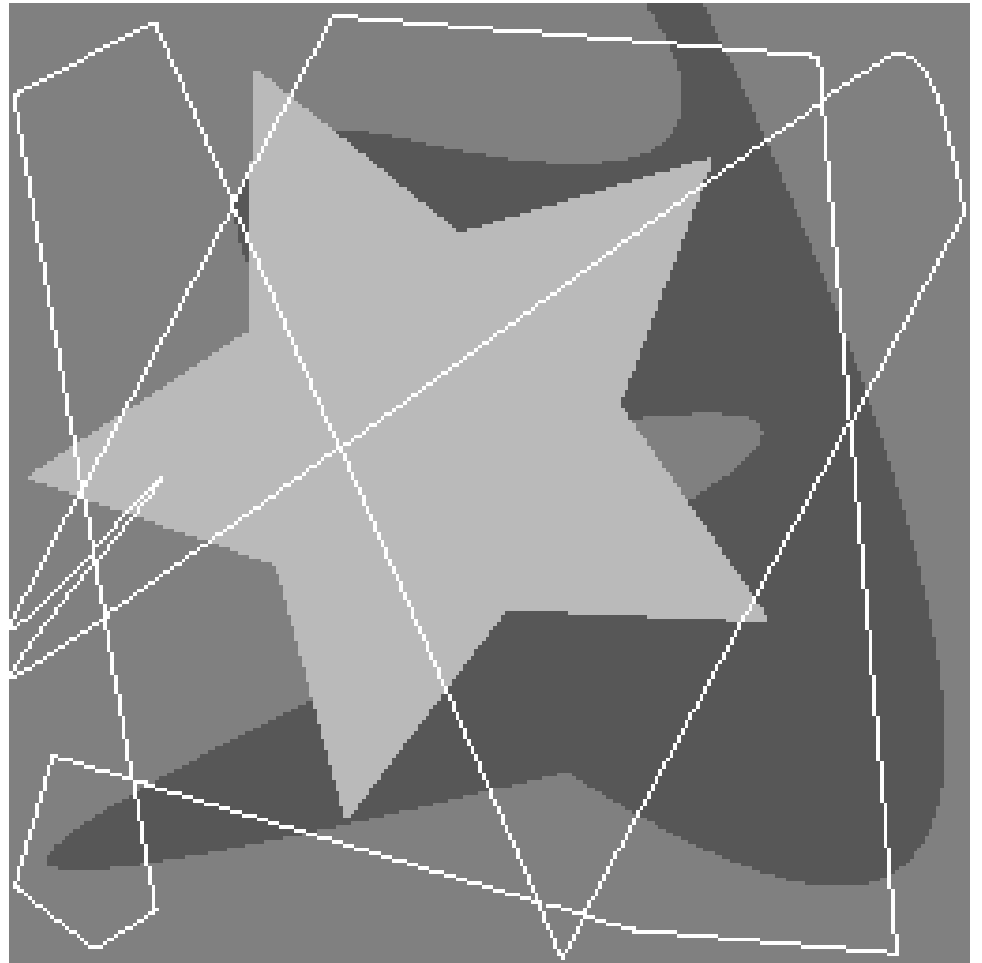


Illustration of Gray Scale Opening and Closing



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

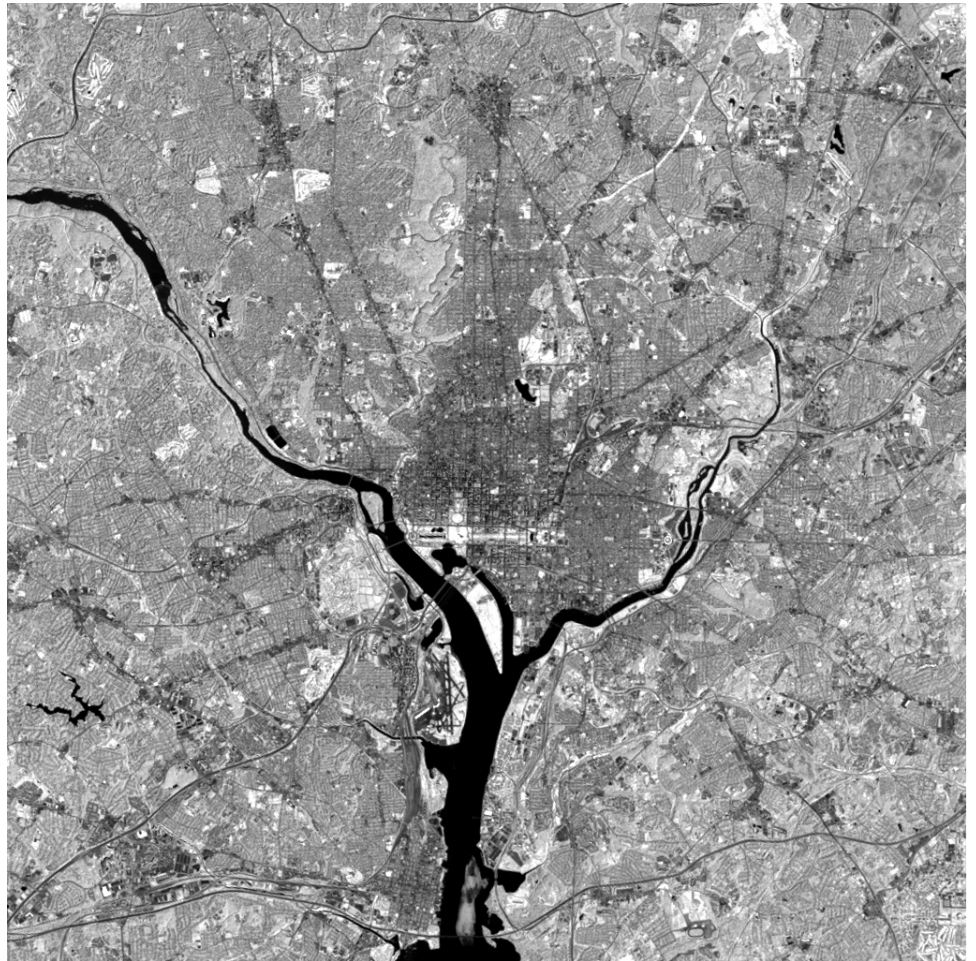
---



**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

---



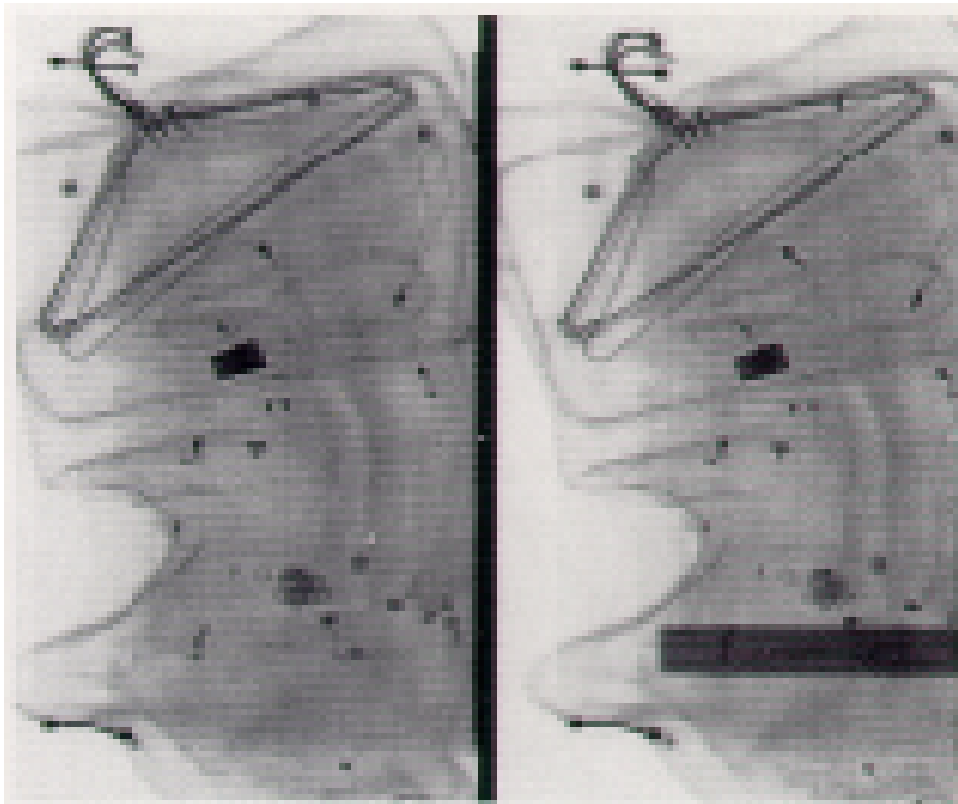
Scilab code AP 29  
Color Coding of Multi Spectral Images

---



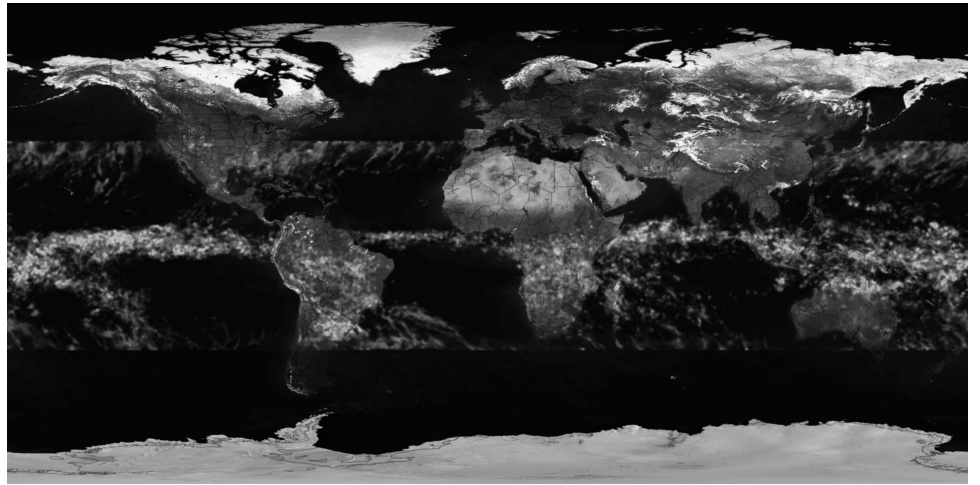
Scilab code AP 30  
Color Coding of Multi Spectral Images

---



Use

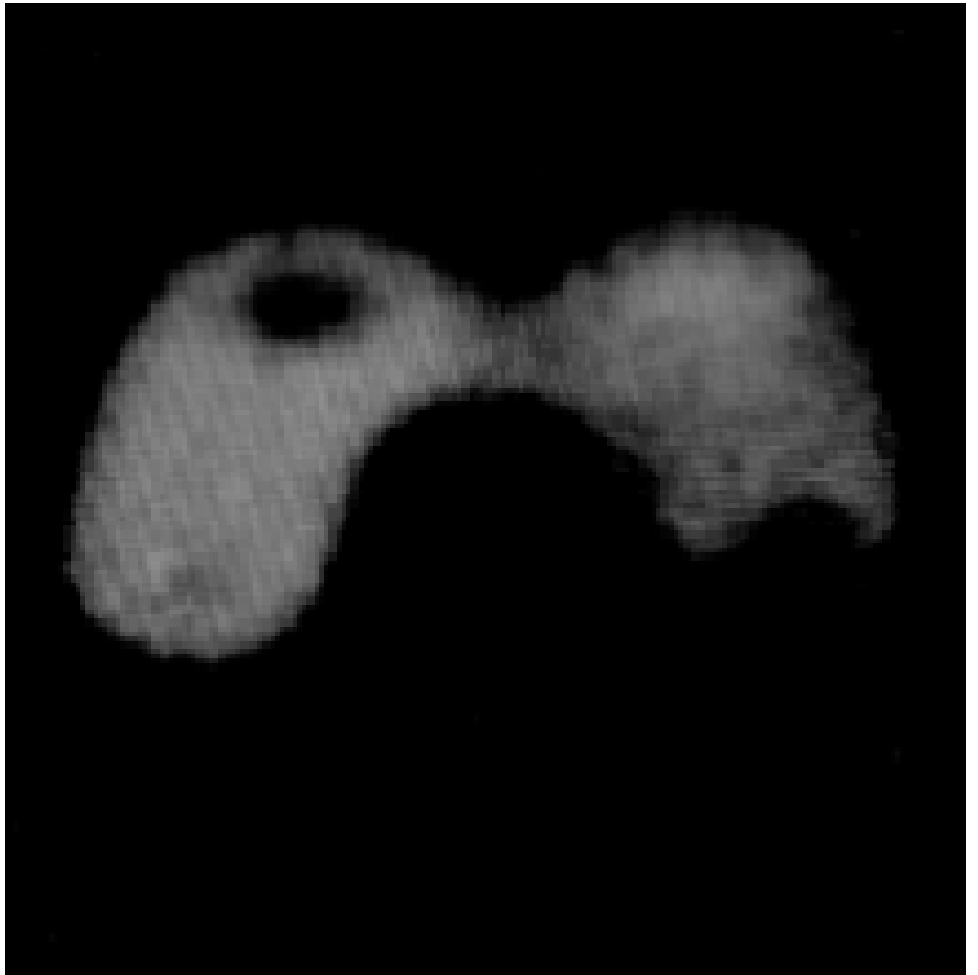
of Pseudocolor for highlighting Explosives Contained in Luggage



**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





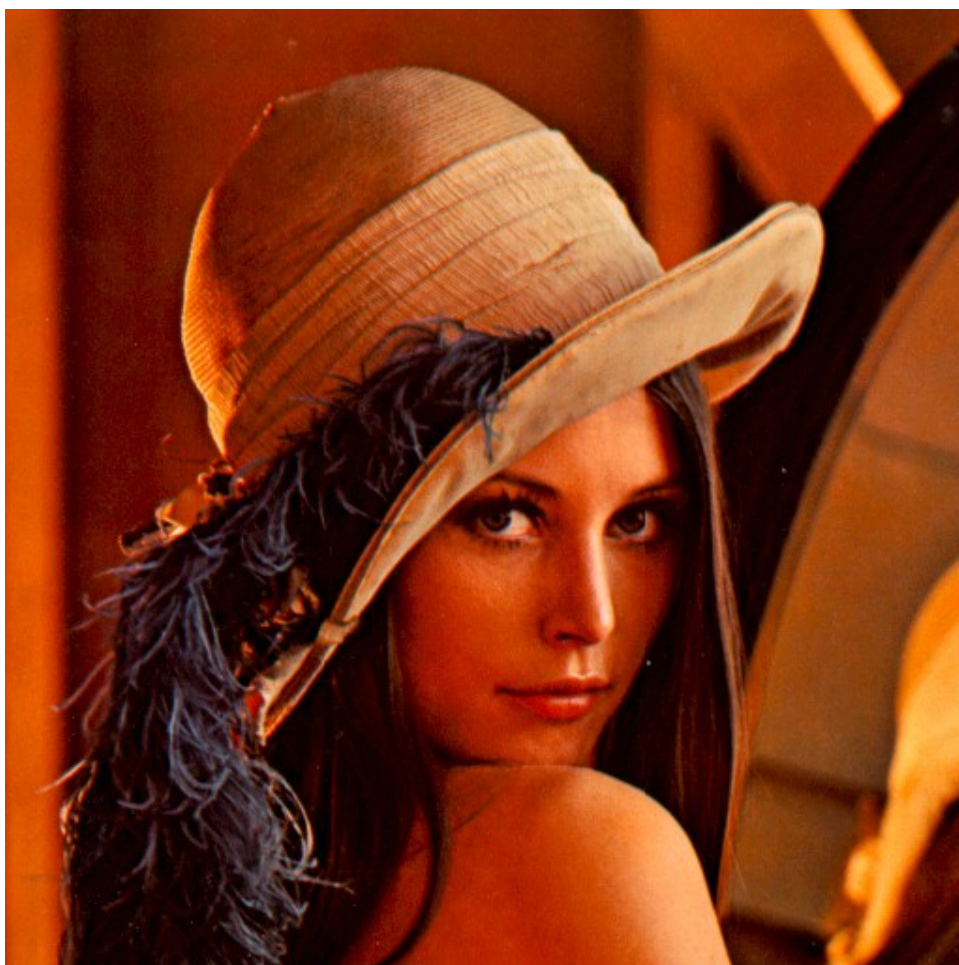
**Scilab code AP 35**

Illustration of the effects of converting noisy RGB Images to HSI



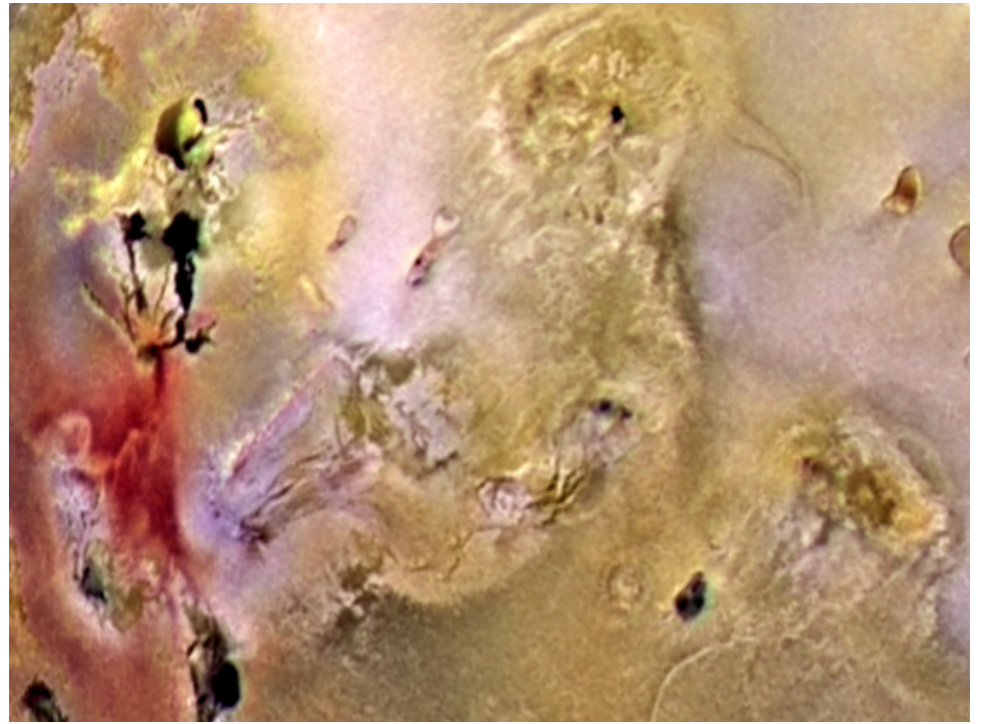
**Scilab code AP 36**

Illustration of the effects of converting noisy RGB Images to HSI



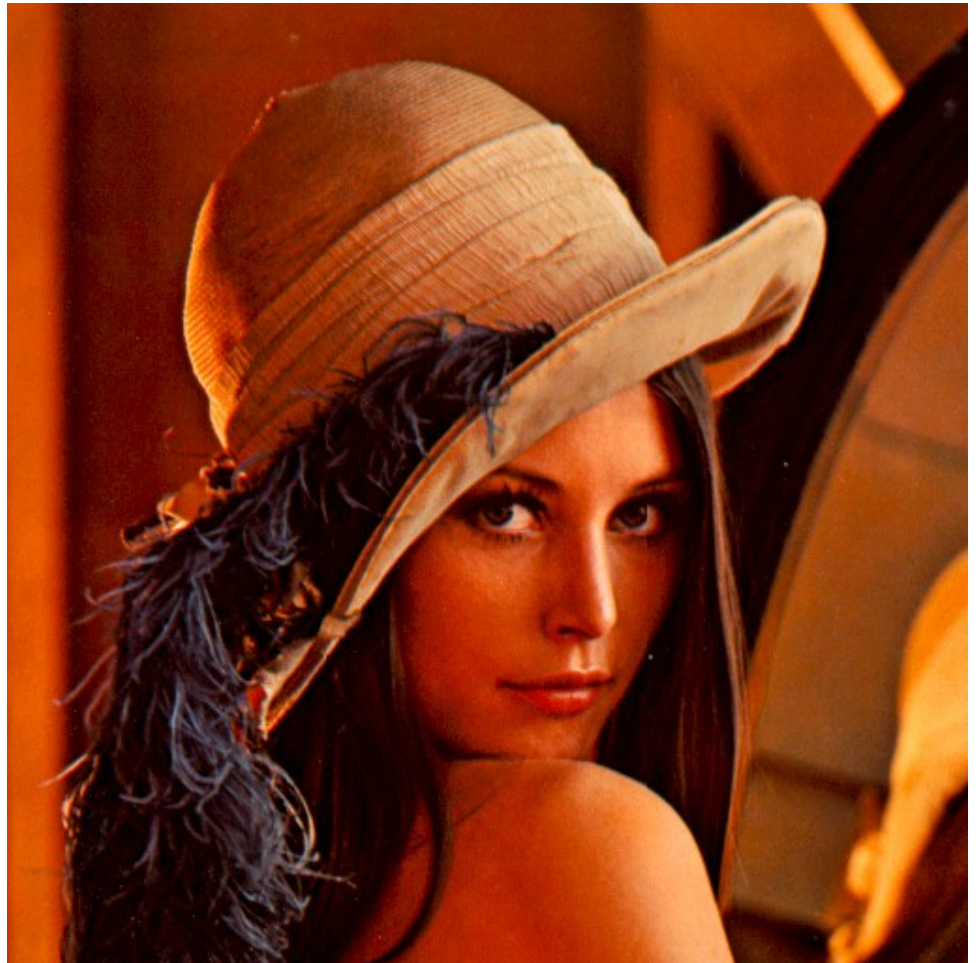
**Scilab code AP 37**  
Edge Detection Vector Space

---



Scilab code AP 38  
segmentation in HSI Space

---



Scilab code AP 39  
Sharpning with the Laplacian

---



**Scilab code AP 40**

Color Image Smoothing by Neighbourhood Averaging



Scilab code AP 41  
Histogram Equalization in the HSI Color Space

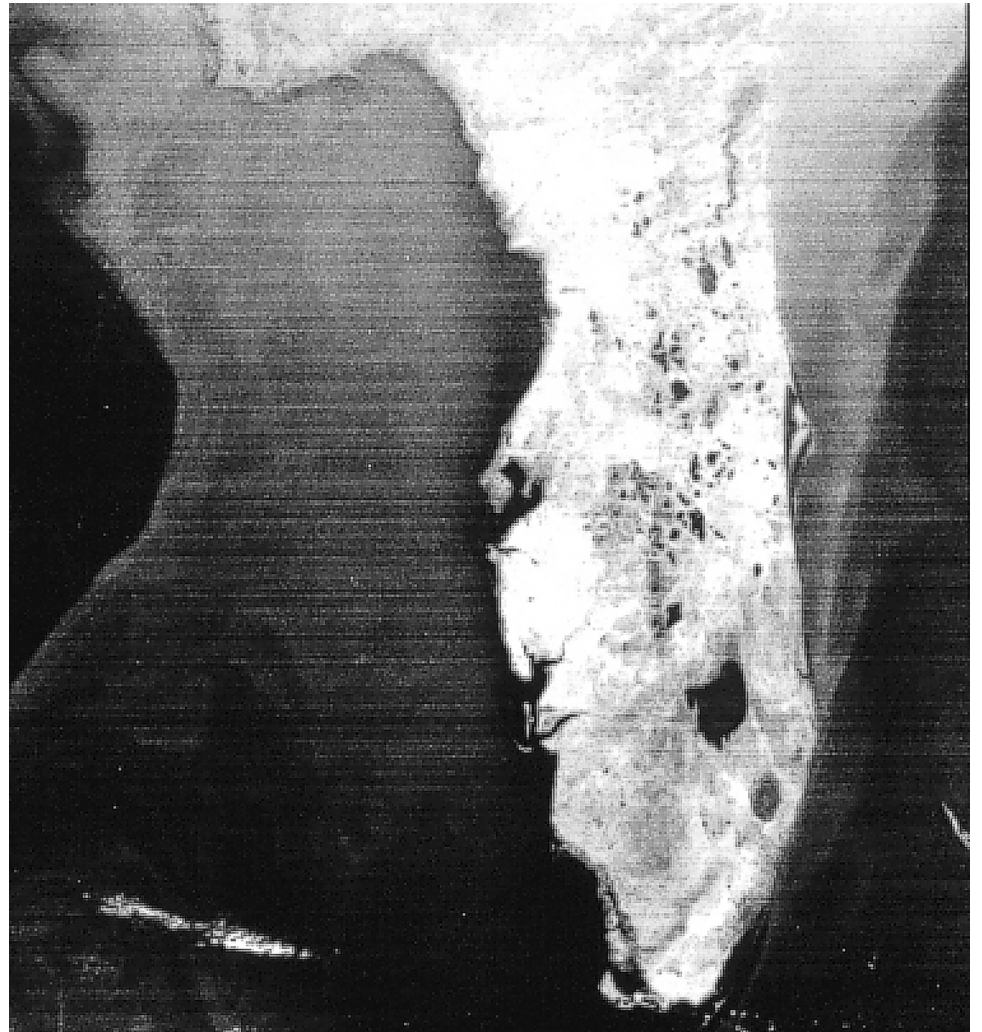
---



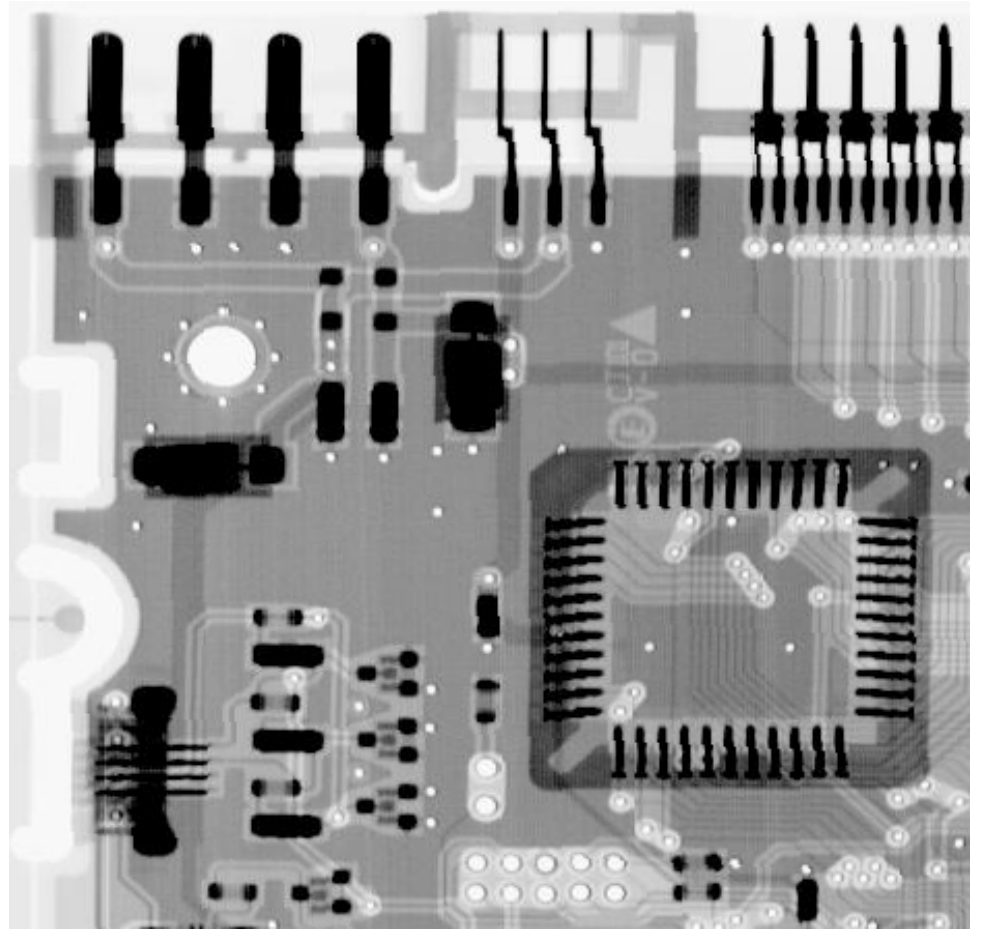
Scilab code AP 42  
Color Balancing

---



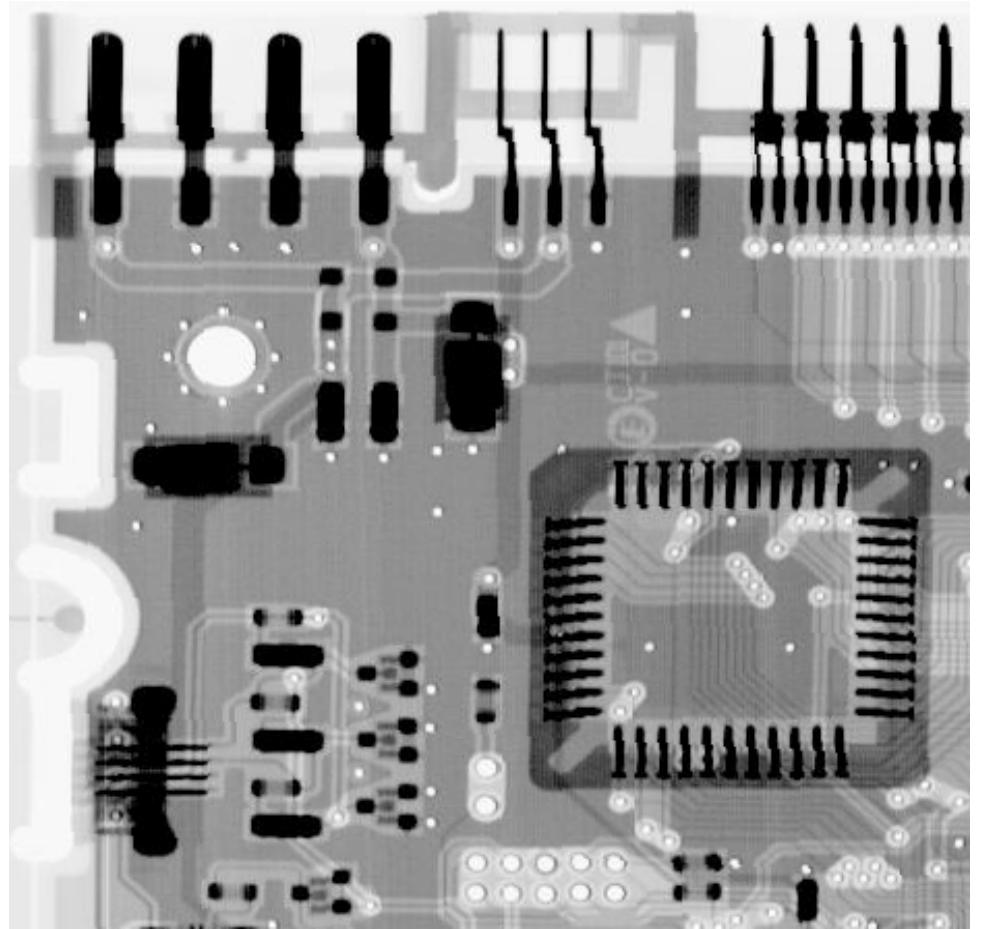


**Scilab code AP 43**  
Removal of Periodic Noise by Notch Filtering



Scilab code AP 44  
Illustration of Adaptive Median Filter

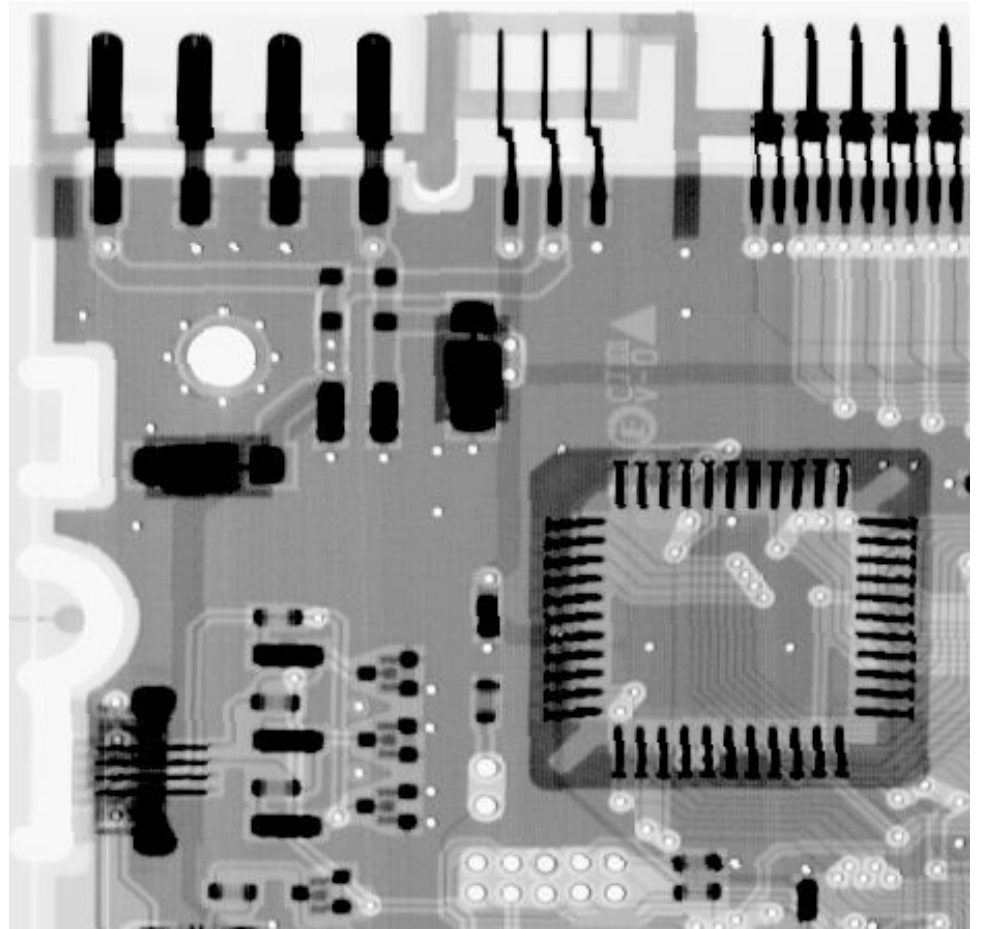
---



Scilab code AP 45

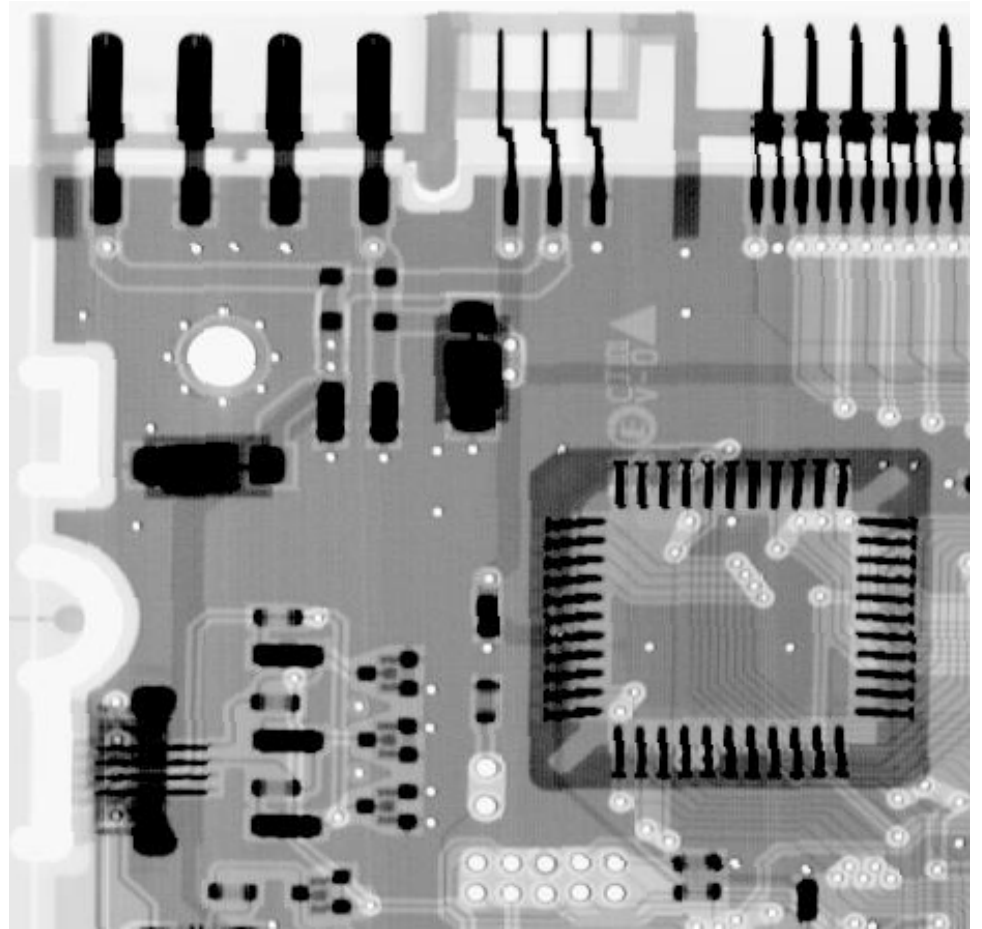
Illustration of Adaptive Local Noise Reduction Filtering

---



Scilab code AP 46  
Illustration of Order Statistic filter

---



Scilab code AP 47  
Illustration of Mean Filters

---



Comparison of Inverse Filtering and Wiener Filtering



Inverse Filtering

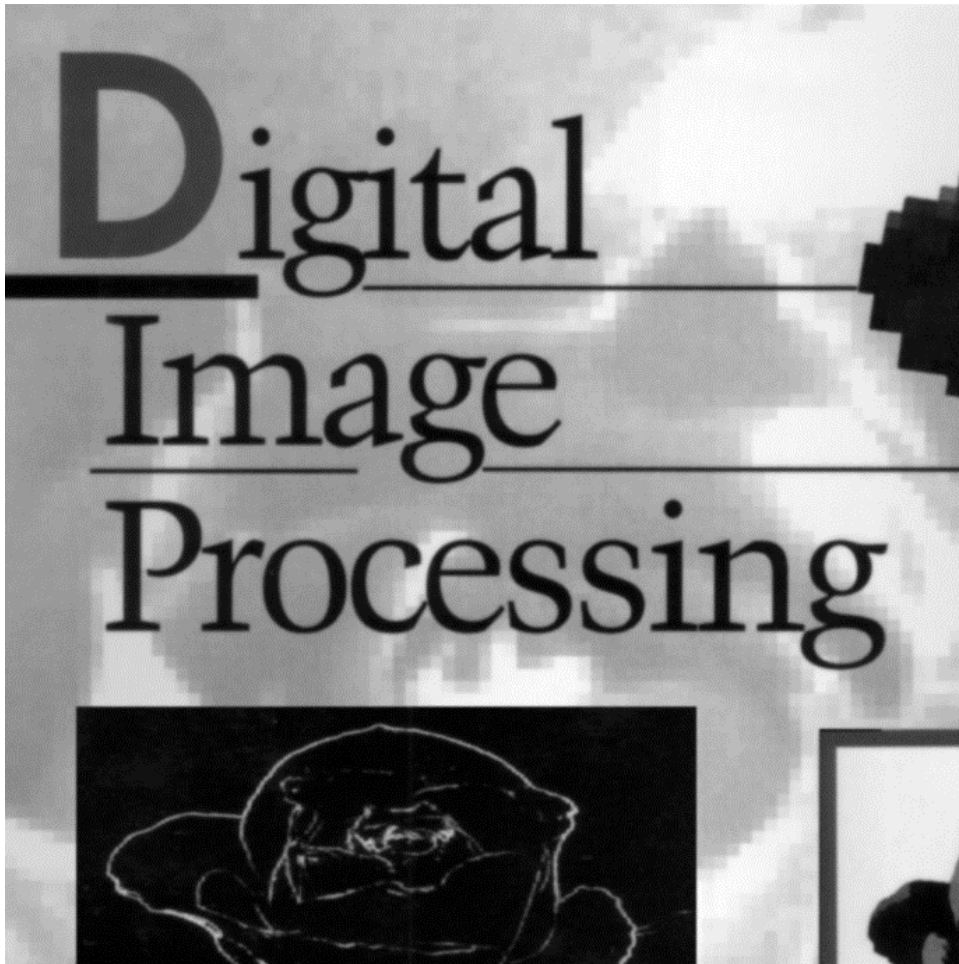
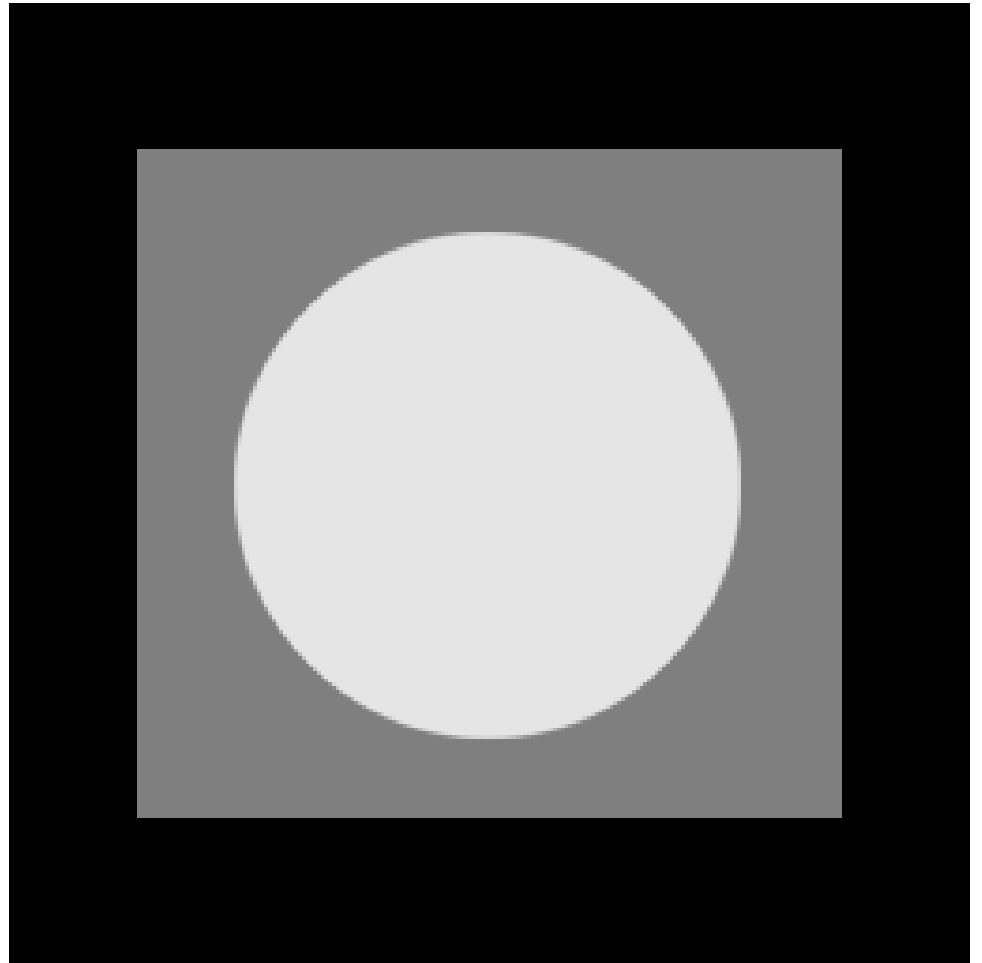


Image Blurring Due to Motion

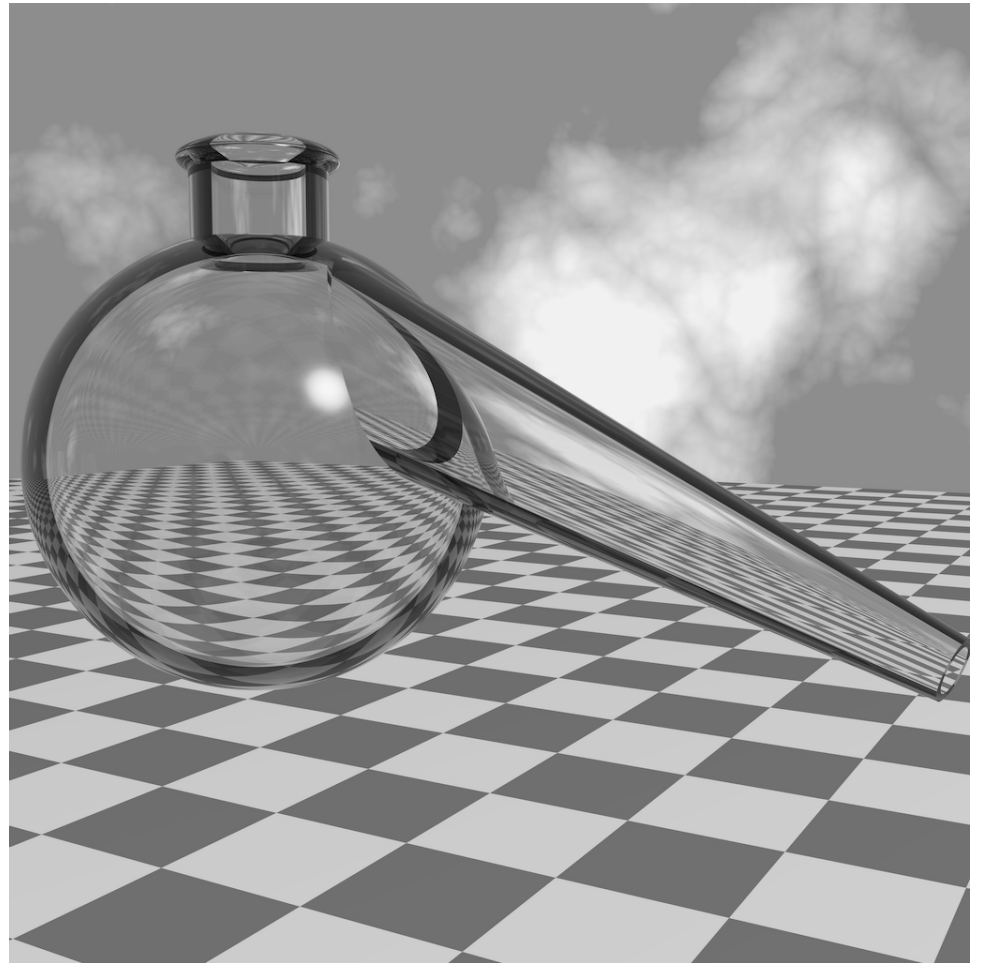
Im-





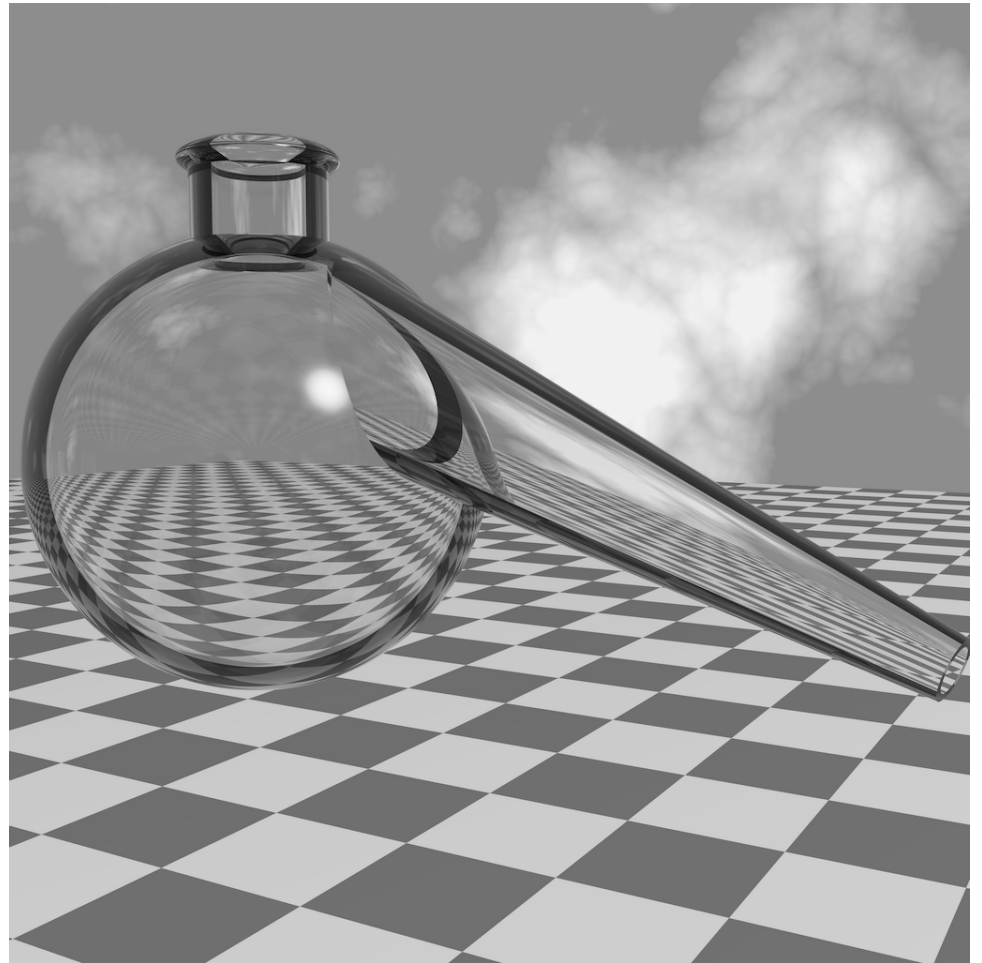
Scilab code AP 51  
Noisy Images and their Histogram

---



Scilab code AP 52  
Illustration of Jaggies in Image Zooming

---

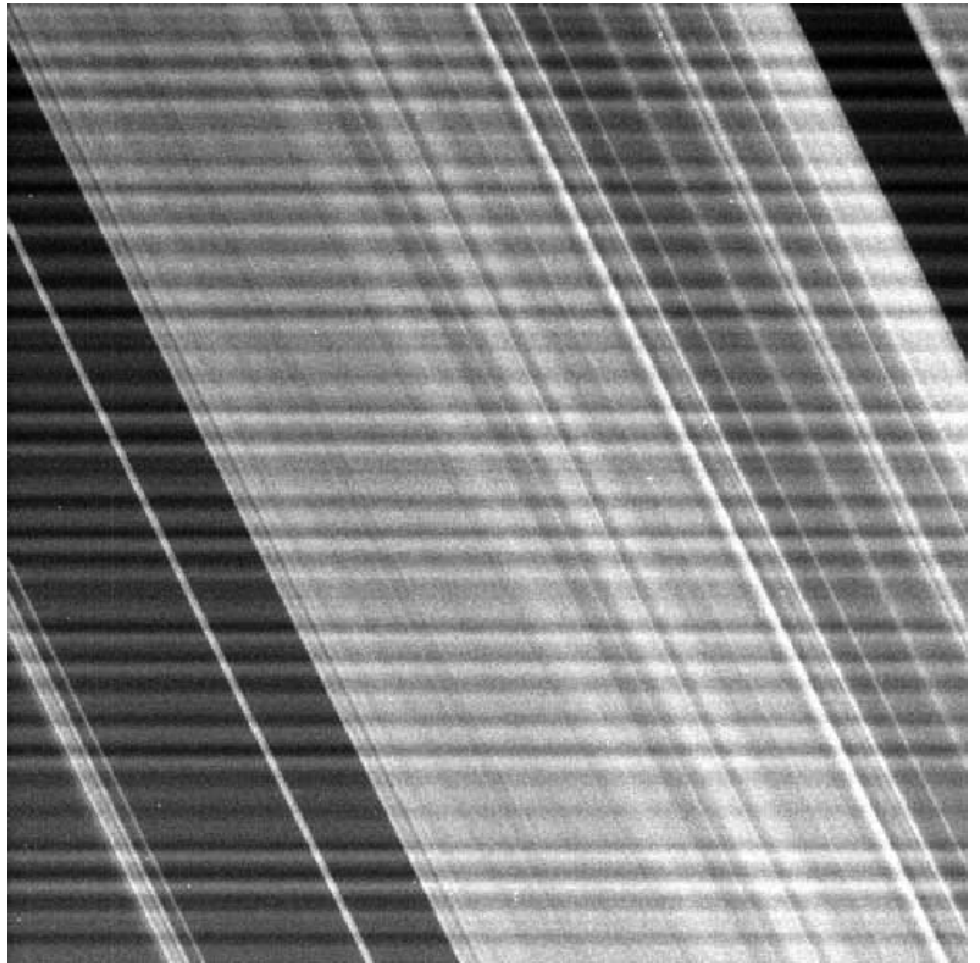


**Scilab code AP 53**  
Illustration of Jaggies in Image Shrinking

---

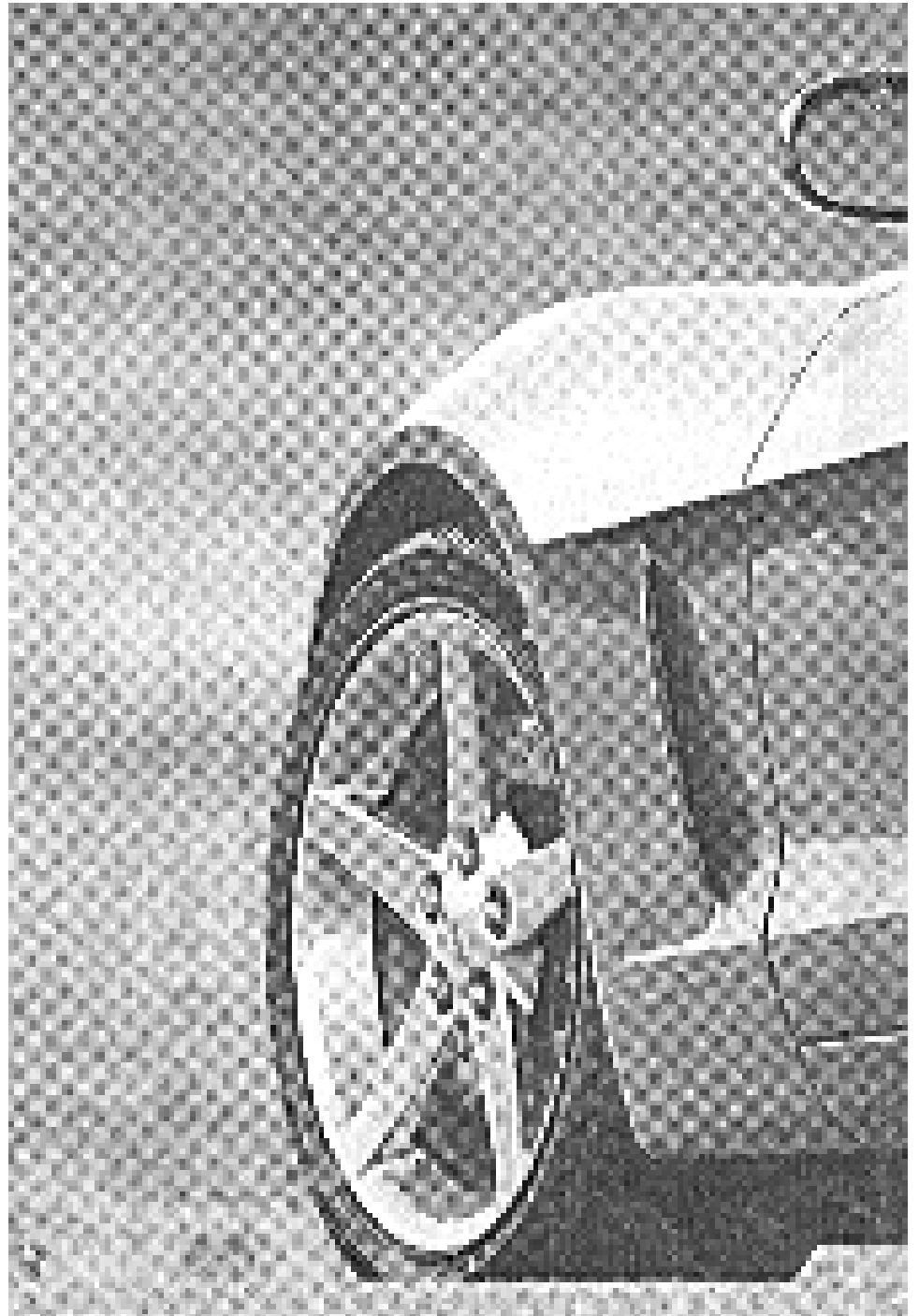


**Scilab code AP 54**  
Illustration of Aliasing in Resampled Images



**Scilab code AP 55**

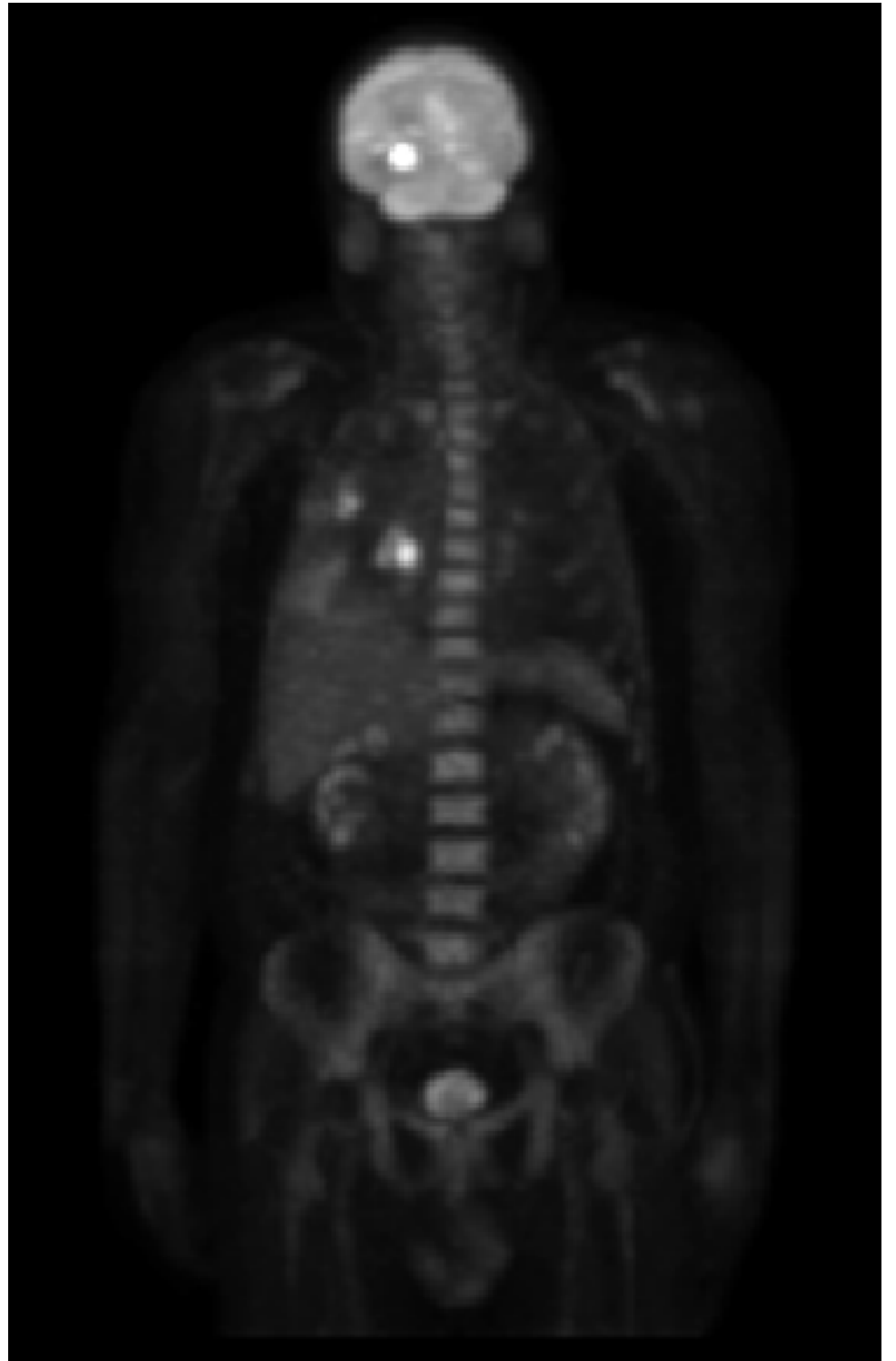
Enhancement of Corrupted Cassini Saturn Image by Notch Filtering



Scilab code AP 56

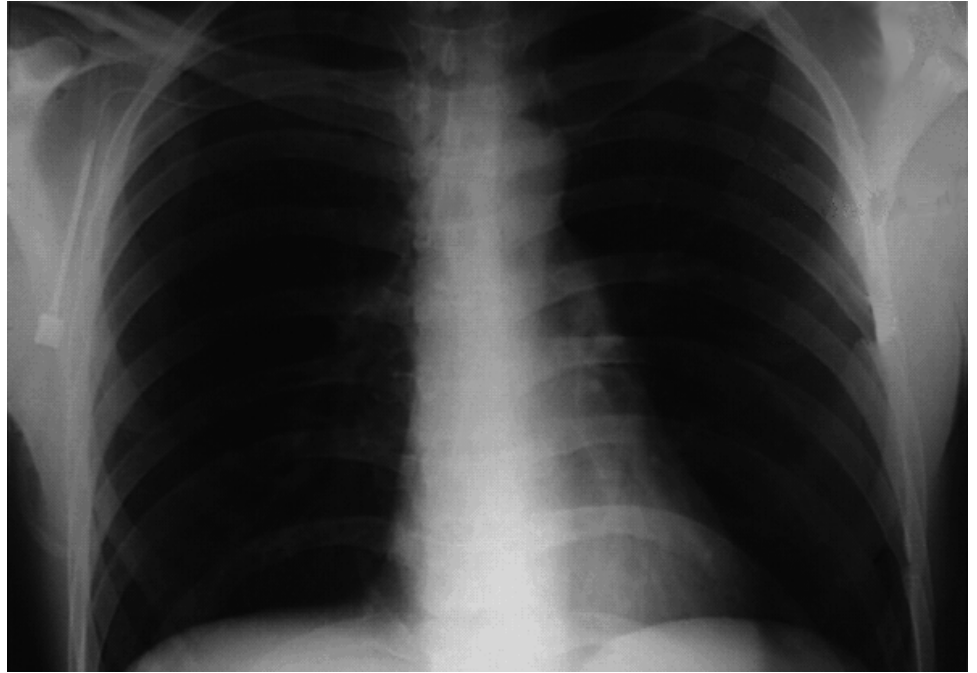
Reduction of Moire Patterns Using Notch Filtering

---



Scilab code AP 57

Image Enhancement using Homomorphic Filtering



**Scilab code AP 58**

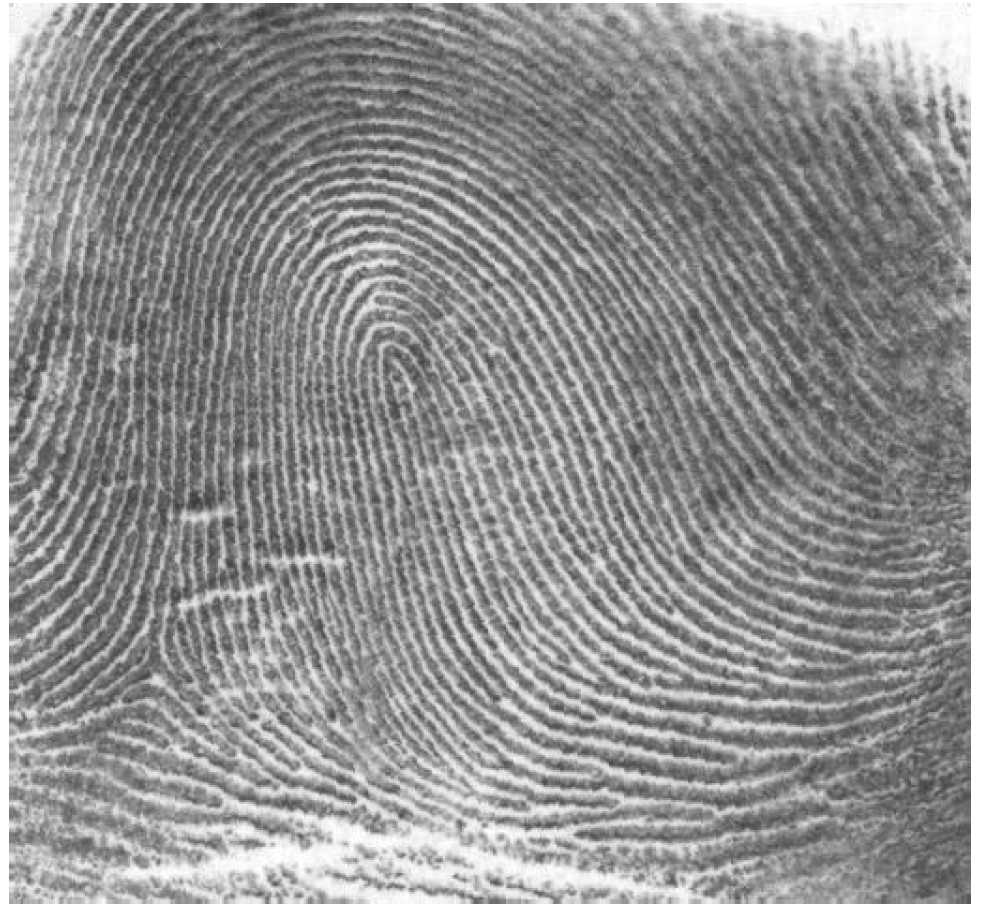
Image Enhancement using High Frequency Emphasis Filtering





**Scilab code AP 59**

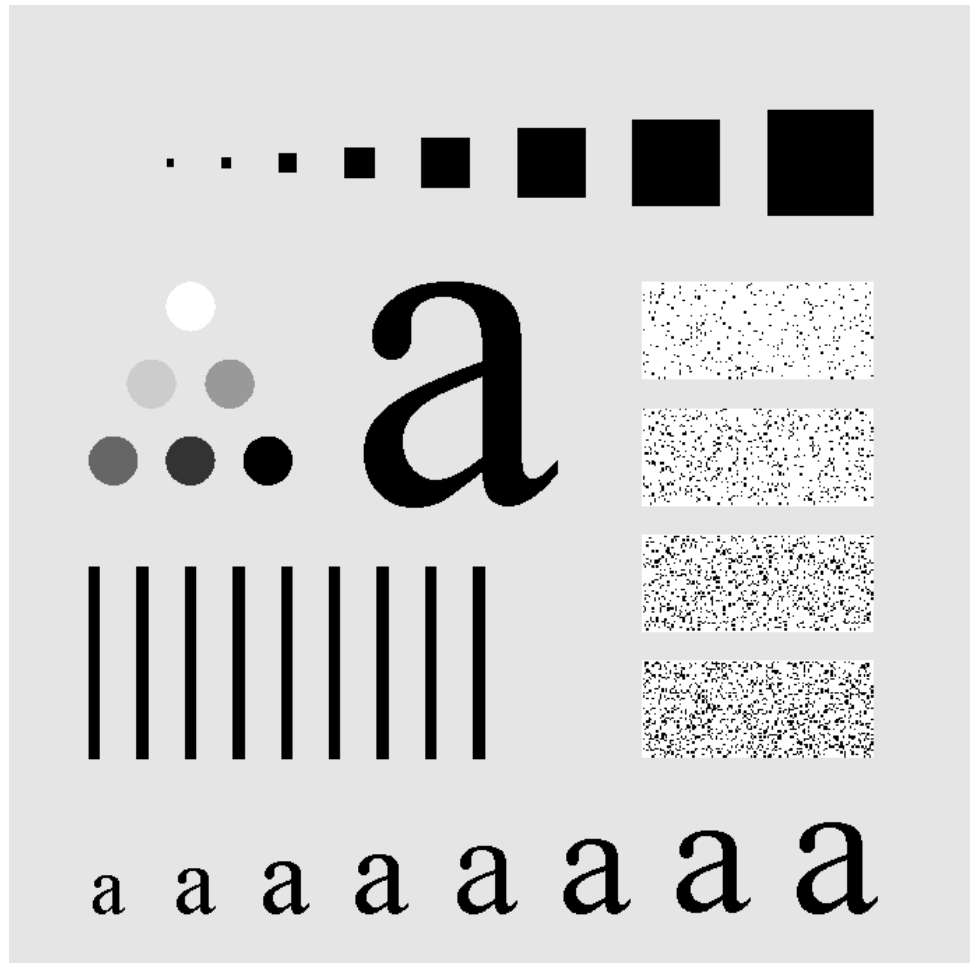
Image Sharpening in the Frequency Domain using the Laplacian



**Scilab code AP 60**

Using Highpass Filter and Thresholding for Image enhancement

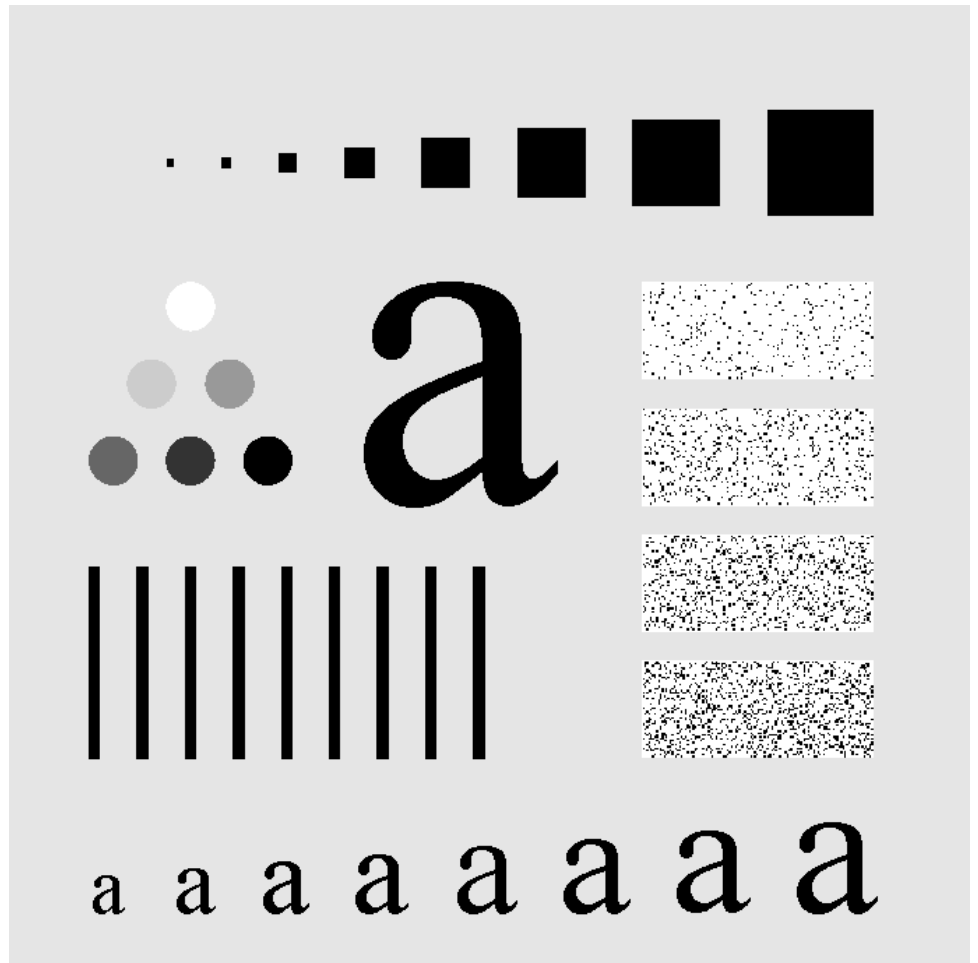
---



Scilab code AP 61

Image Smoothing using Gaussian Lowpass Filter

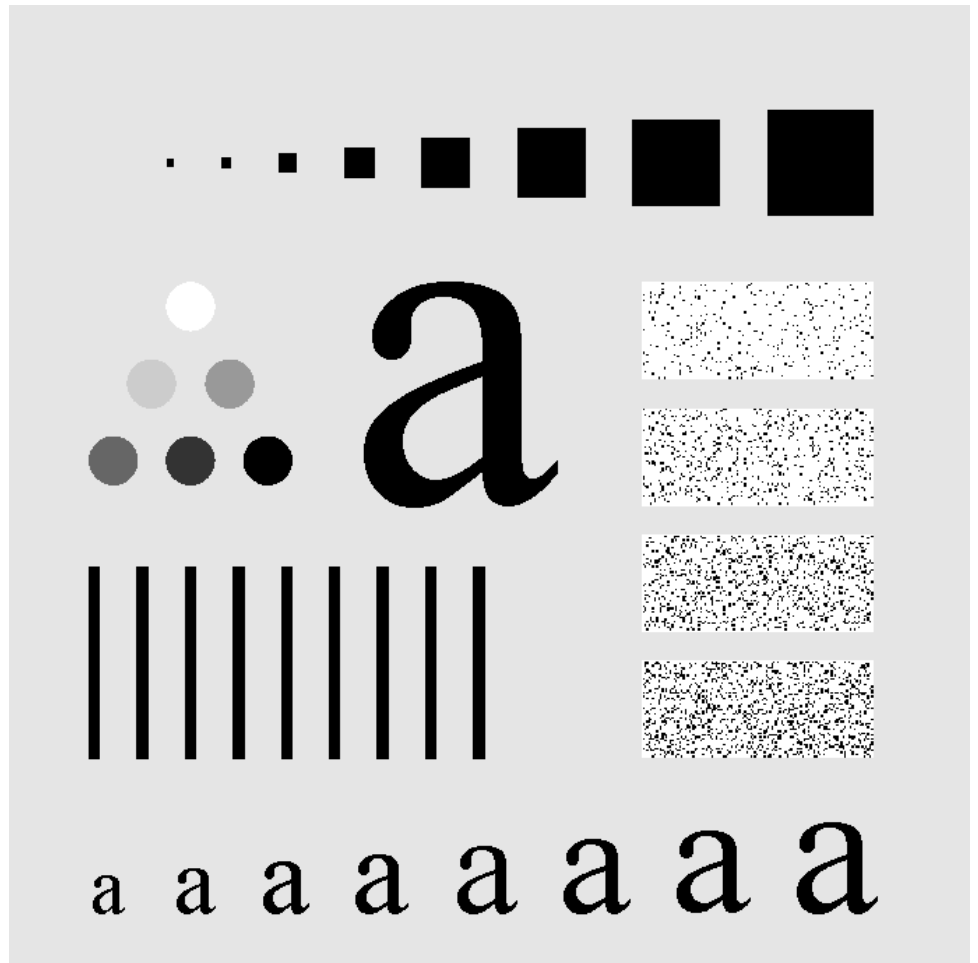
---



Scilab code AP 62

Image Smoothing with a Butterworth Lowpass Filter

---



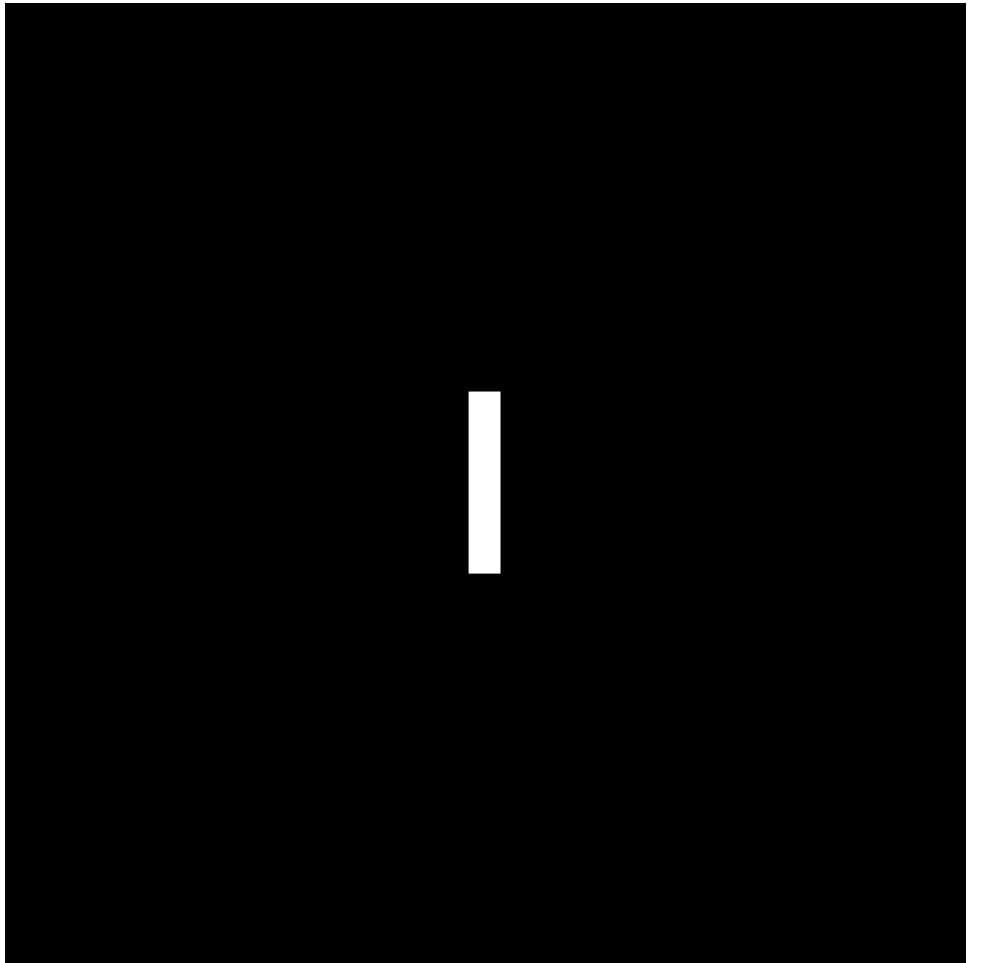
Scilab code AP 63  
Image Smoothing using an ILPF

---



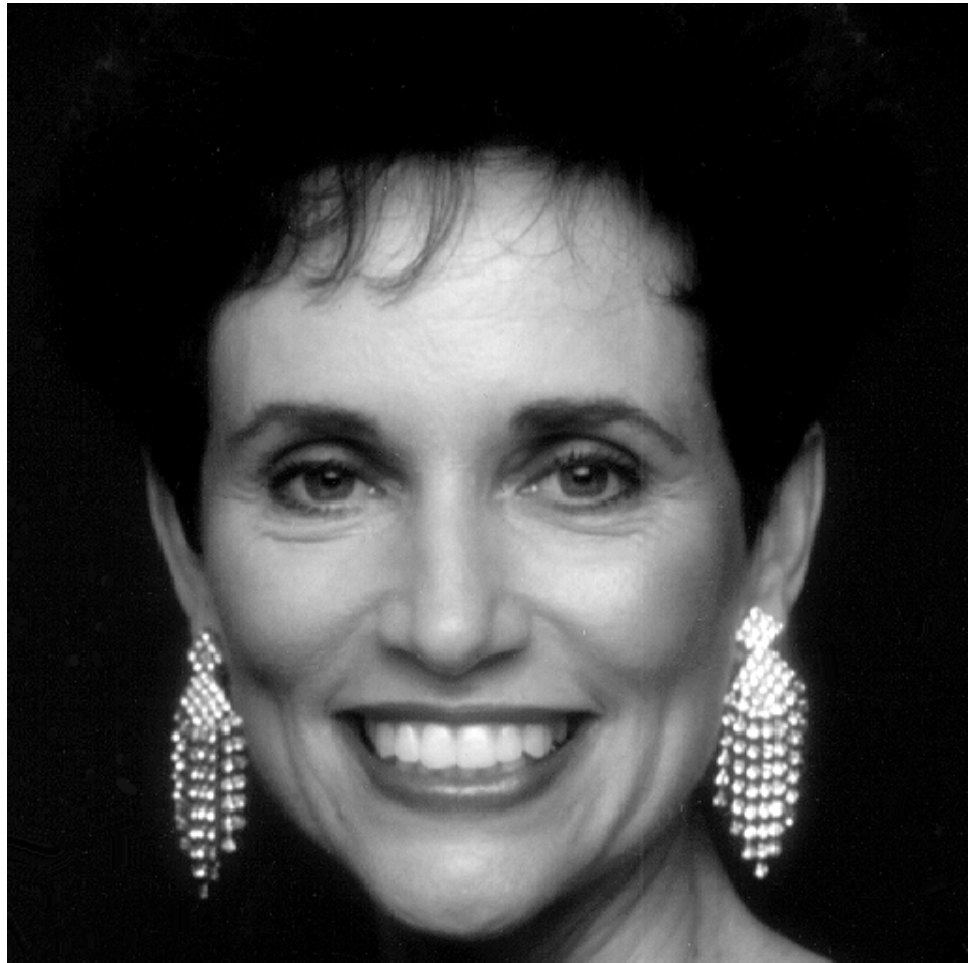
**Scilab code AP 64**

Obtaining a Frequency Domain Filtering from a Small Spatial Mask



**Scilab code AP 65**

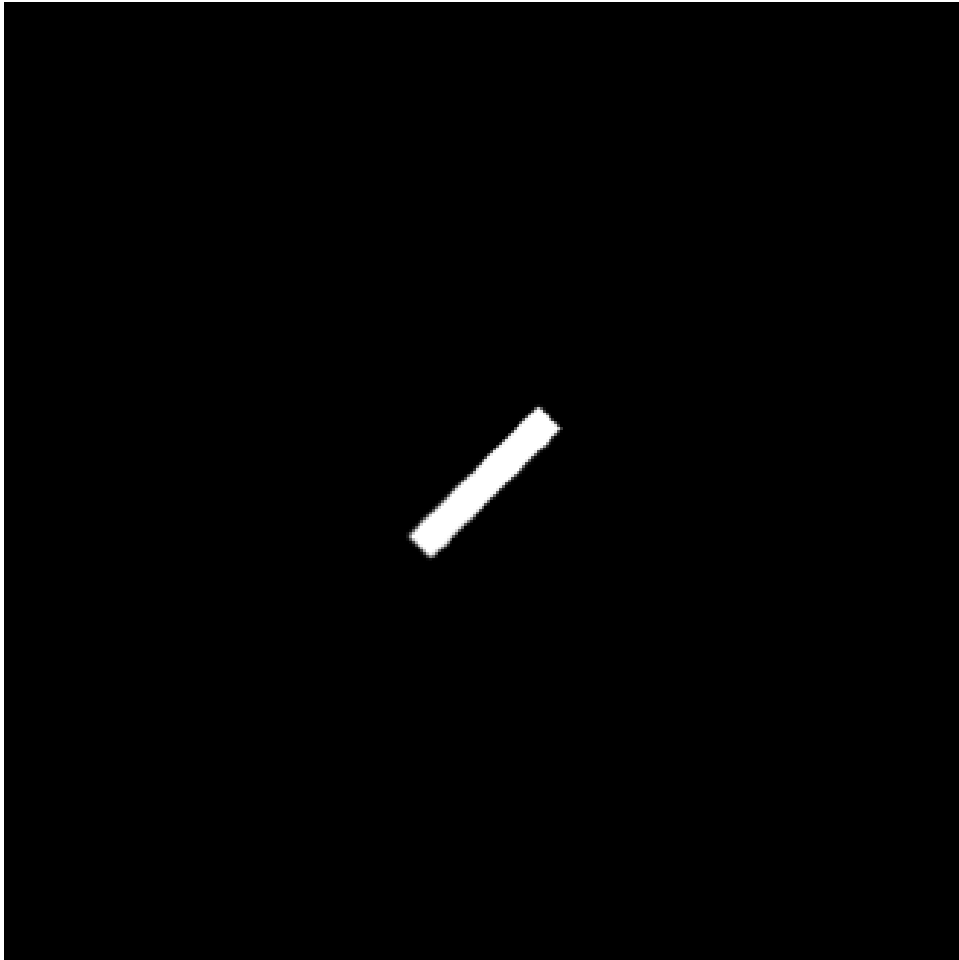
Illustration of the Properties of the Fourier Spectrum and Phase Angle



Scilab code AP 66

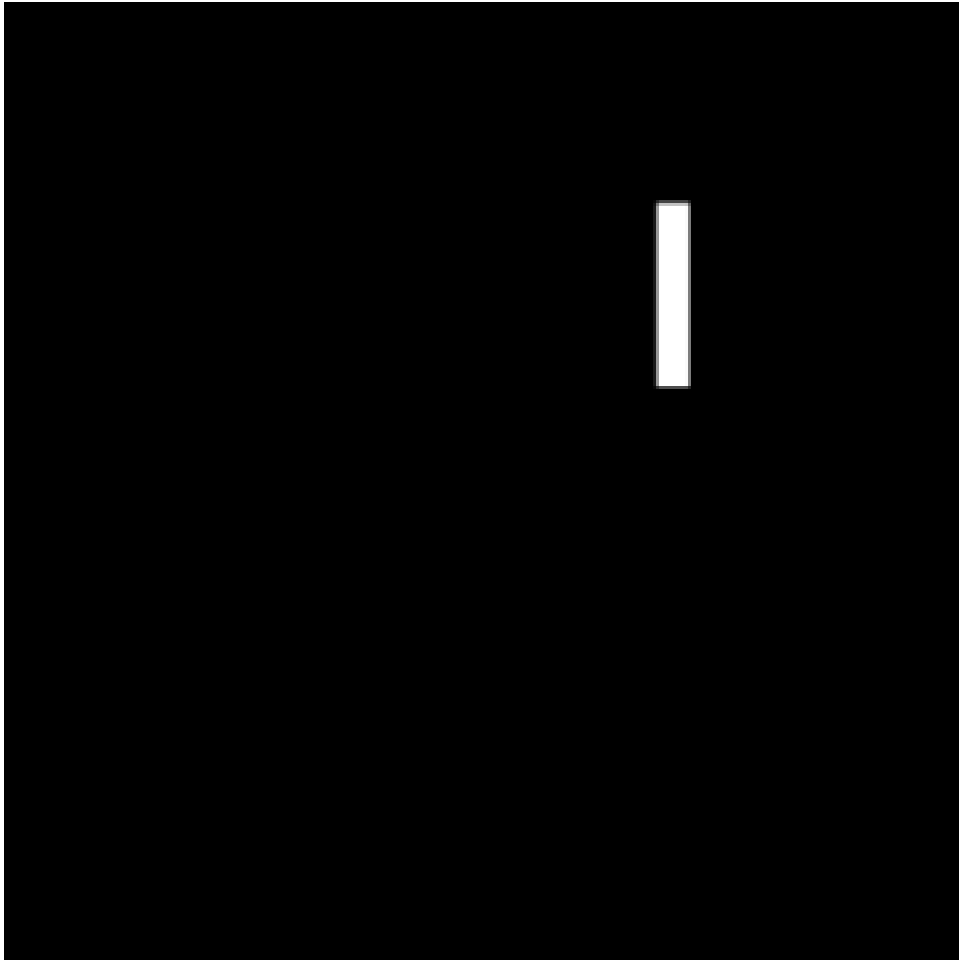
Illustration of the Properties of the Fourier Spectrum and Phase Angle





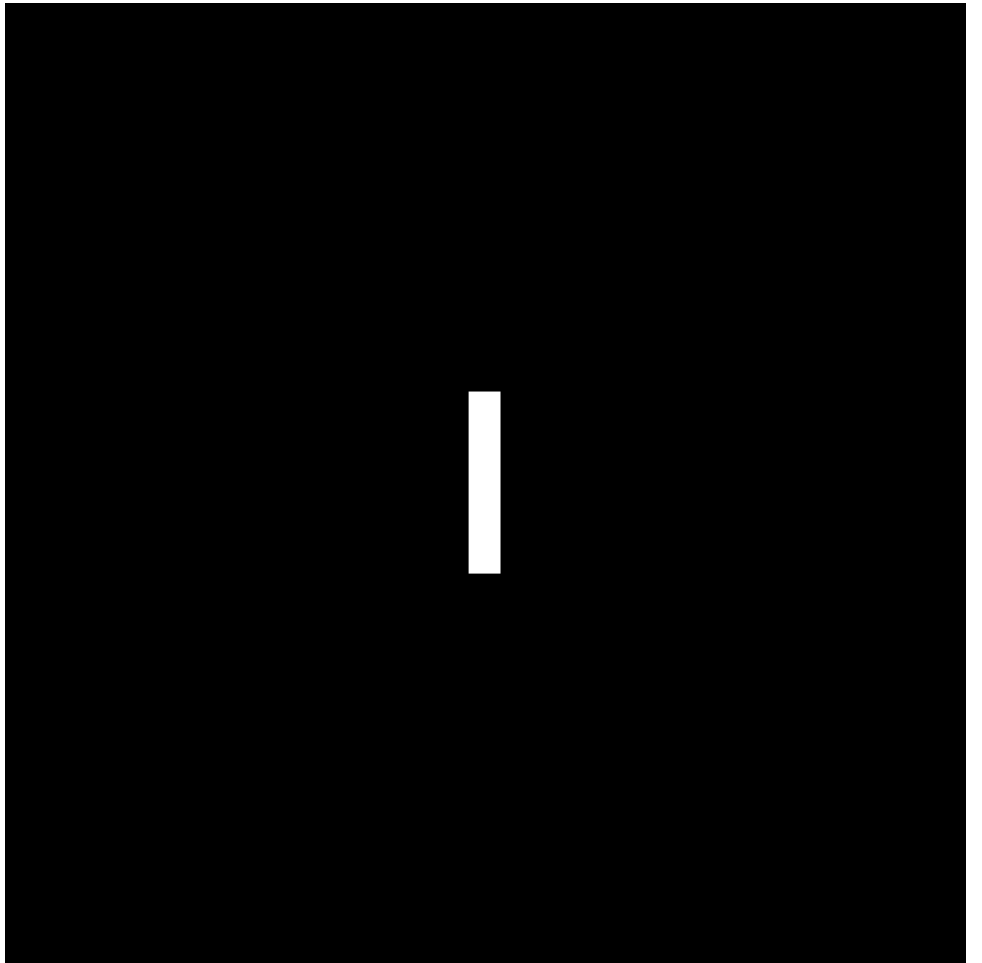
2 D

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

---



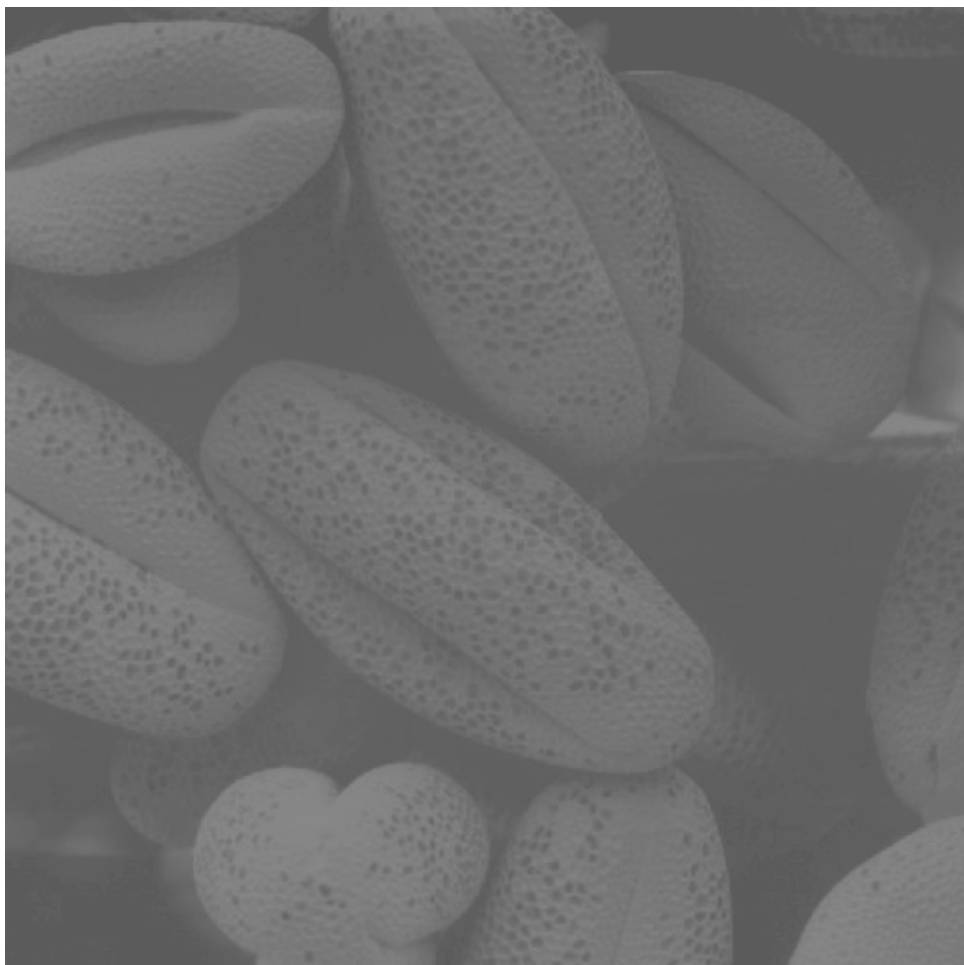
Scilab code AP 70  
Histogram Equalization

---



Scilab code AP 71  
Histogram Equalization

---



Scilab code AP 72  
Histogram Equalization

---



**Scilab code AP 73**  
Intensity Level Slicing

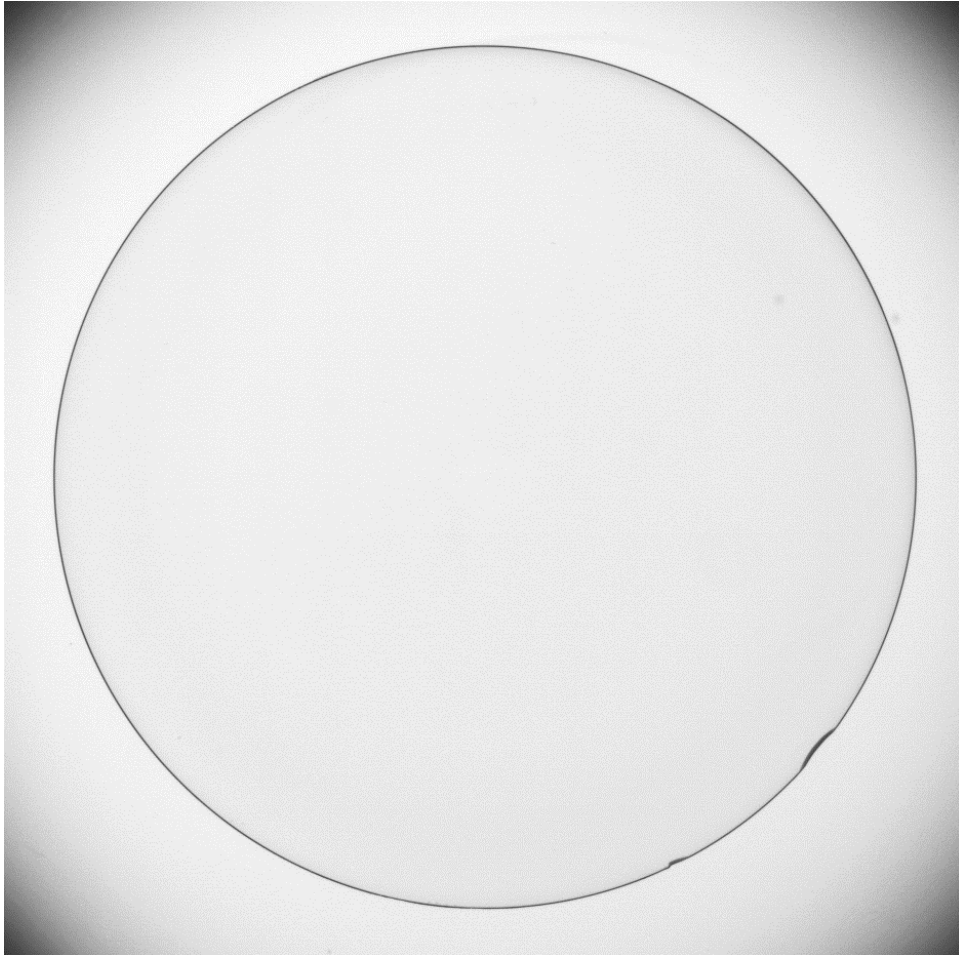
---



Scilab code AP 74  
Illustration of Power Law Transformation

---





of gradient for Edge Enhancement

Use



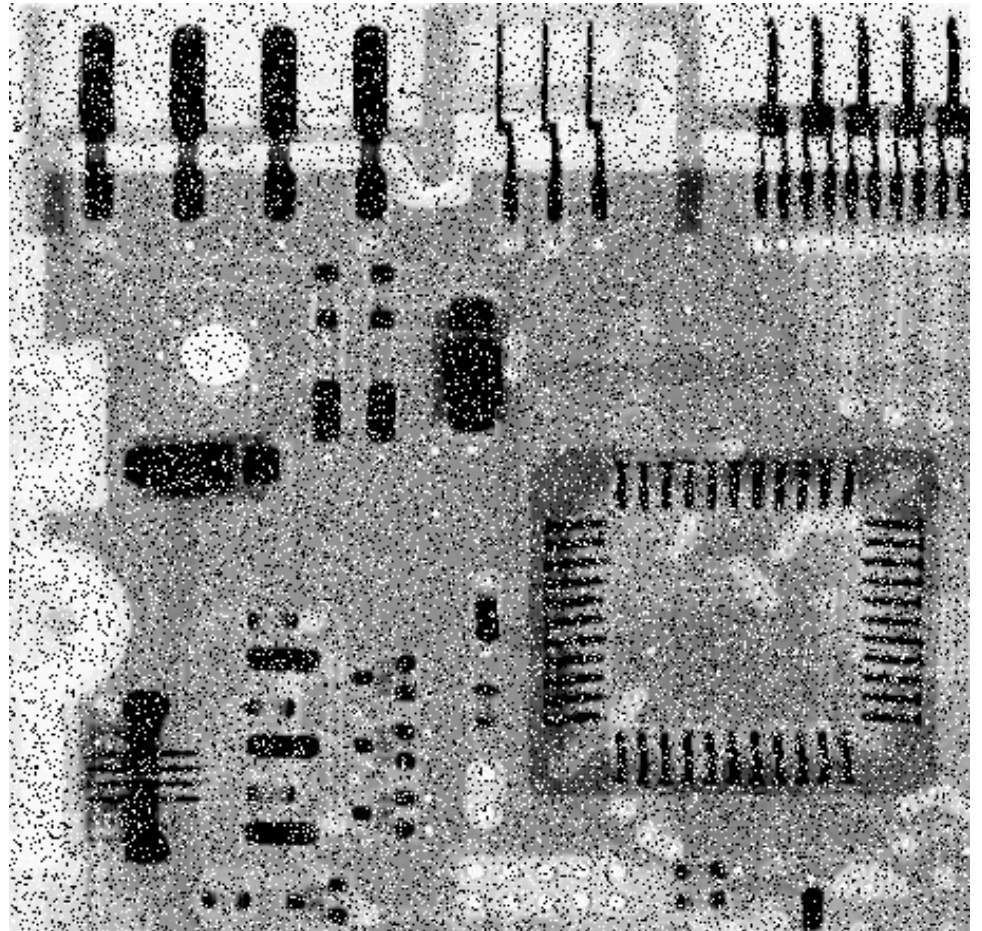
**Scilab code AP 76**

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



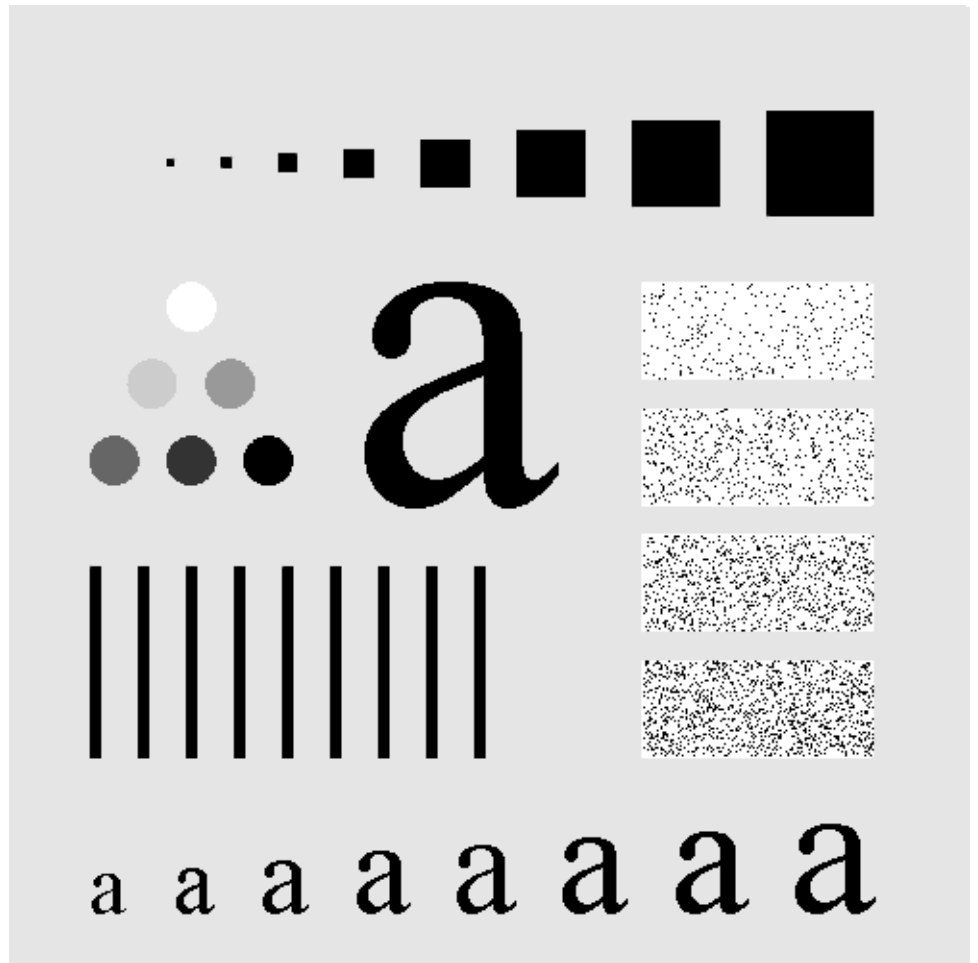
Scilab code AP 77  
Image Sharpning using Laplacian

---



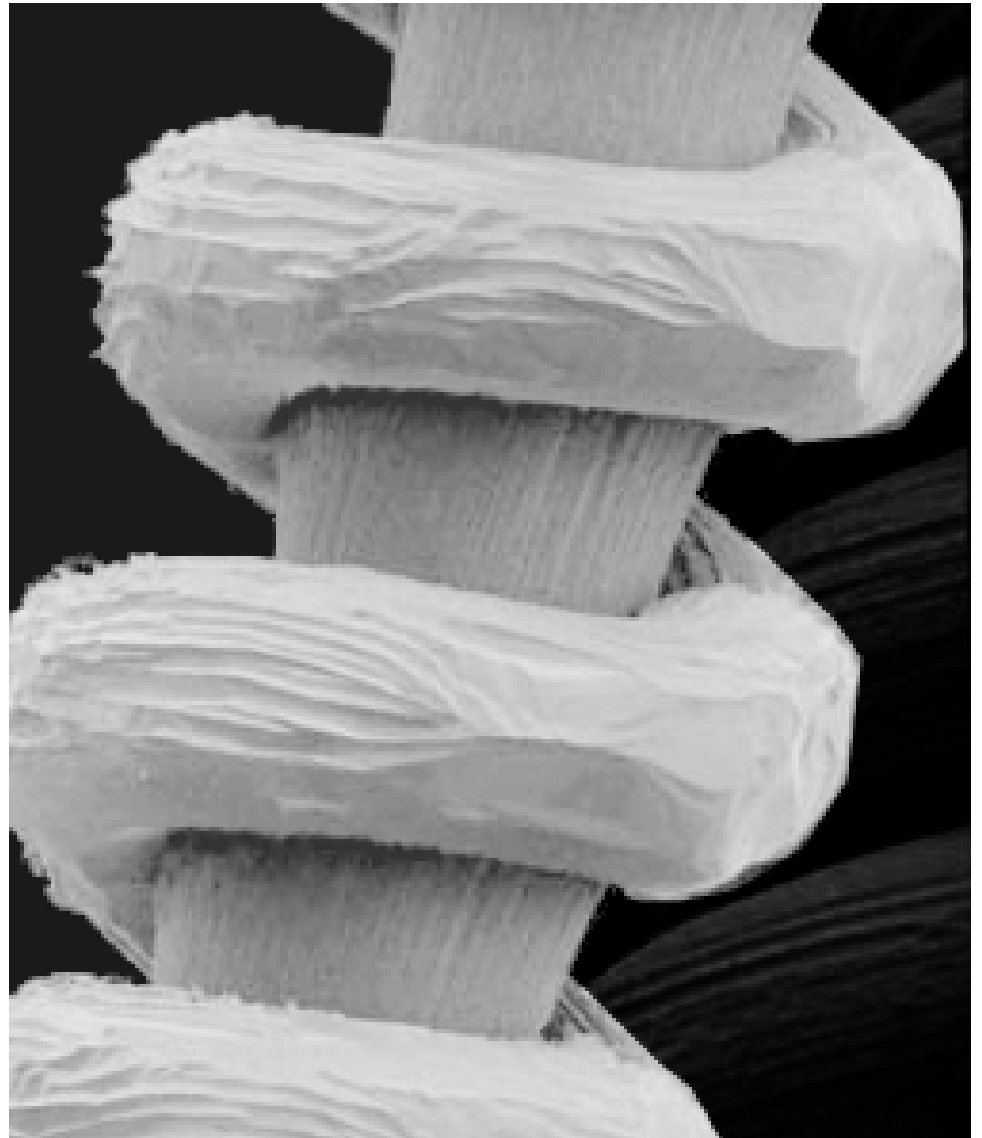
Scilab code AP 78  
Median Filtering for Noise Reduction

---



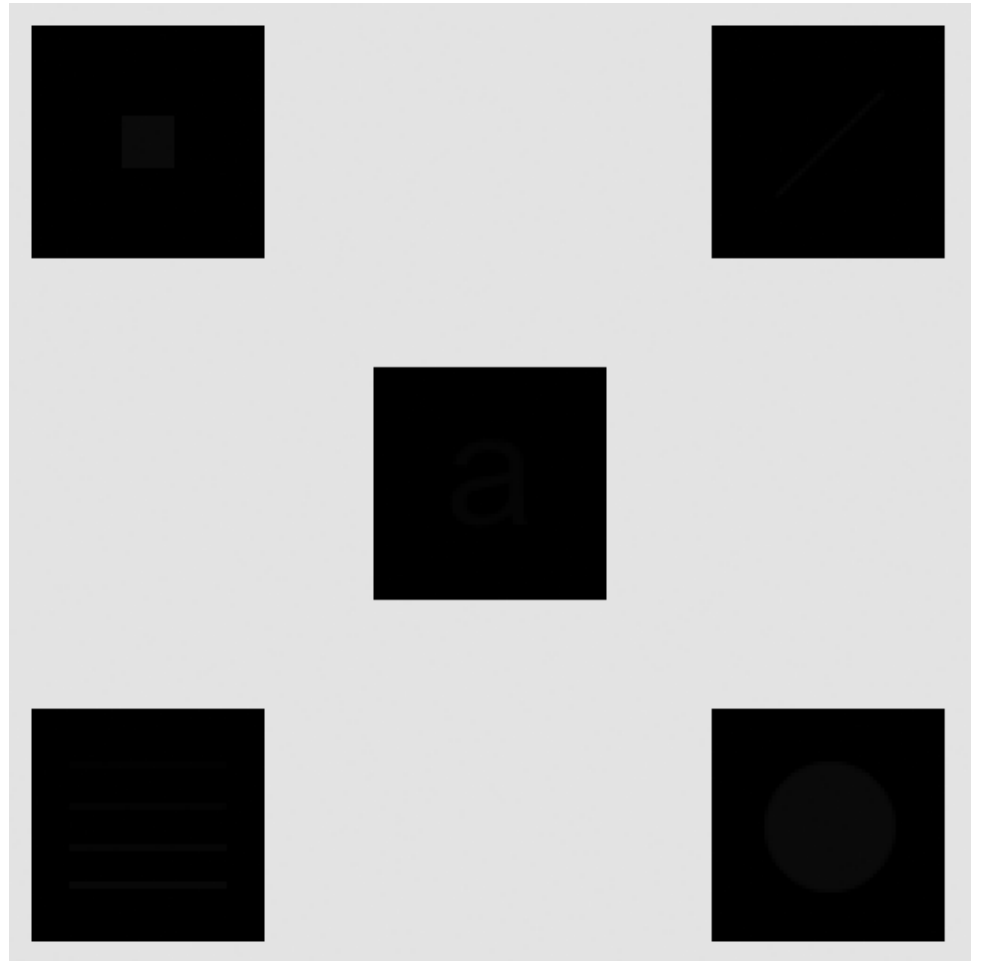
Scilab code AP 79  
Image Smoothing

---



**Scilab code AP 80**  
Local Enhancement using Histogram Statistic

---



Scilab code AP 81  
Local Histogram Equalization

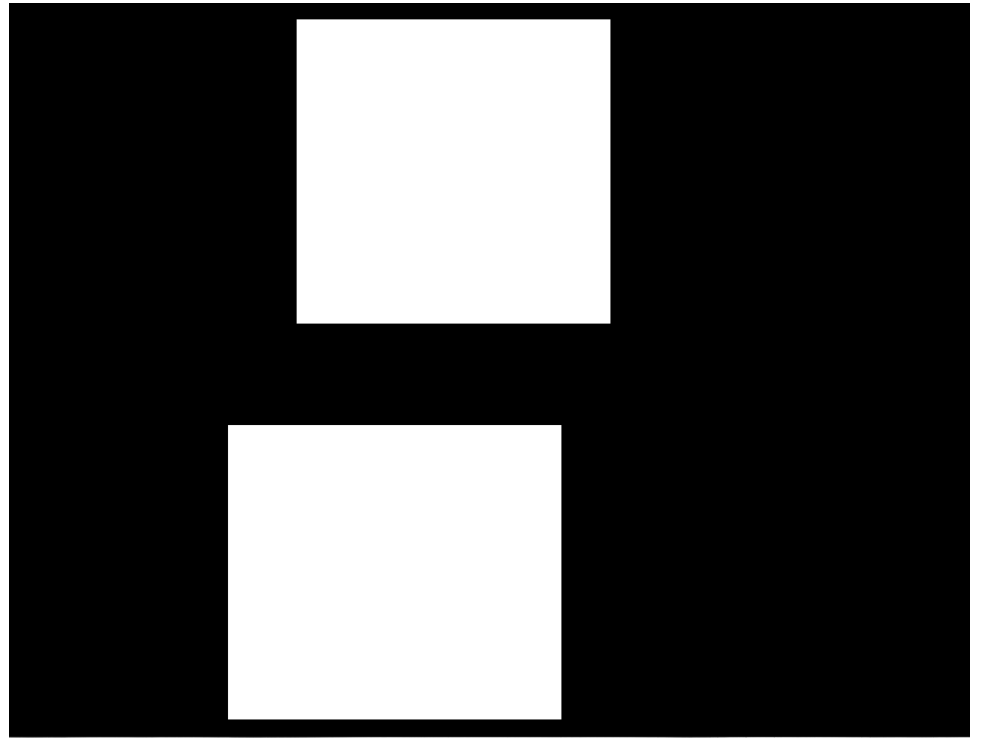
---



Scilab code AP 82  
Gamma Intensity transformation

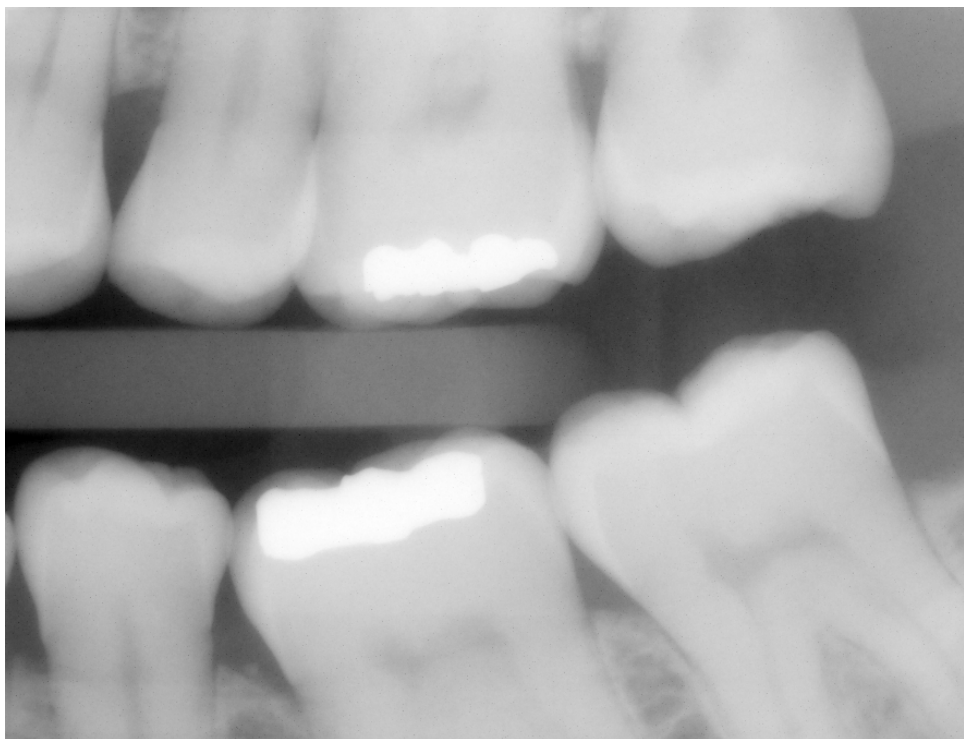
---





**Scilab code AP 83**  
Image Multiplication for Shadding Correction

---



**Scilab code AP 84**

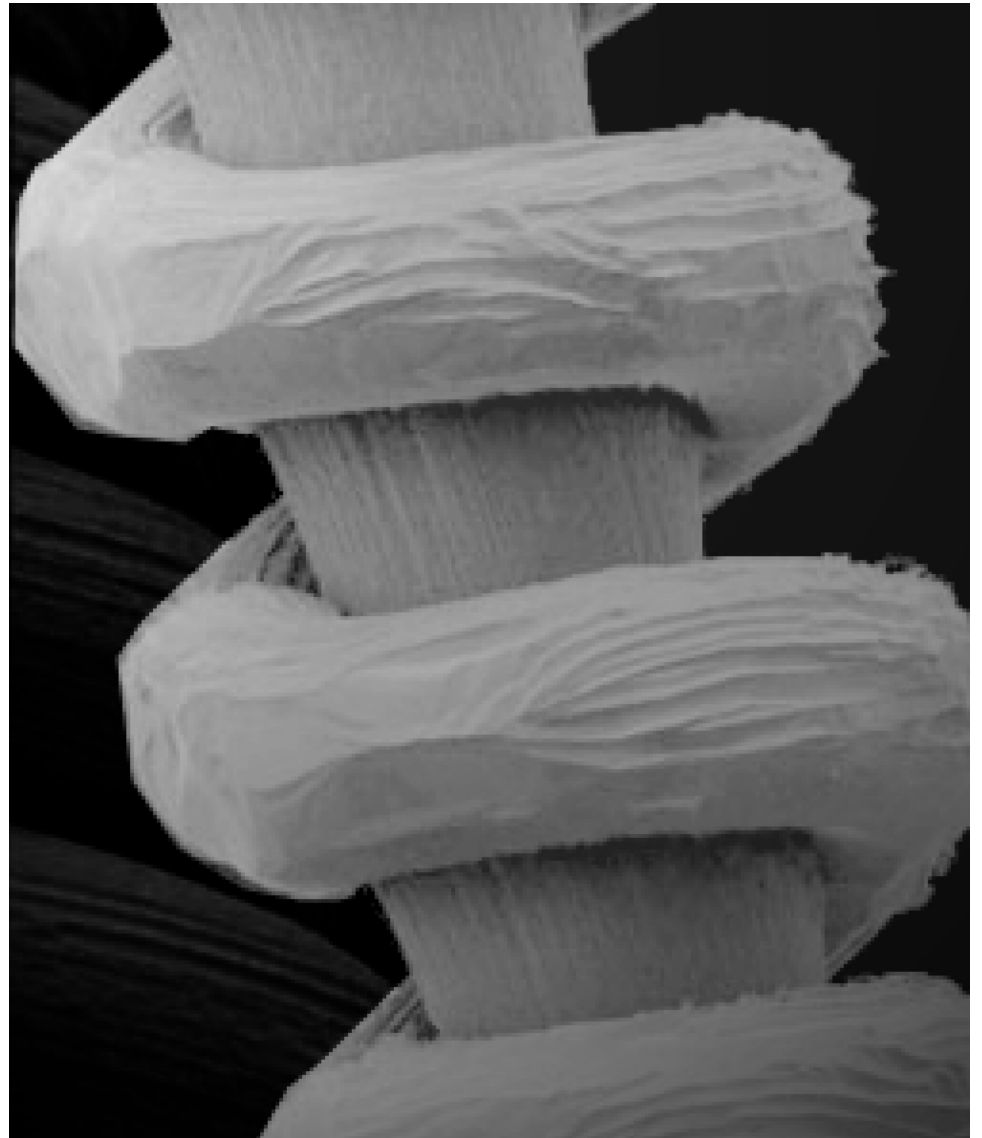
Image Multiplication for Shadding Correction



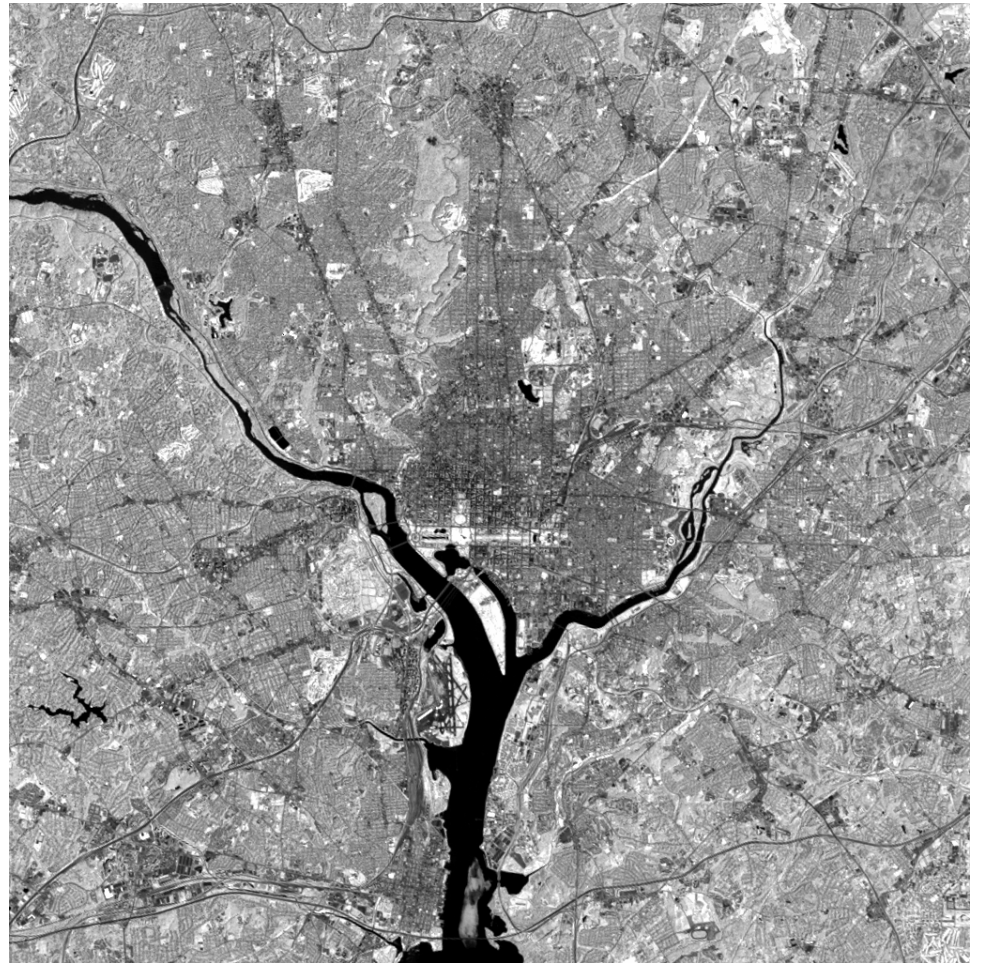
**Scilab code AP 85**

Image Multiplication for Shading Correction

---

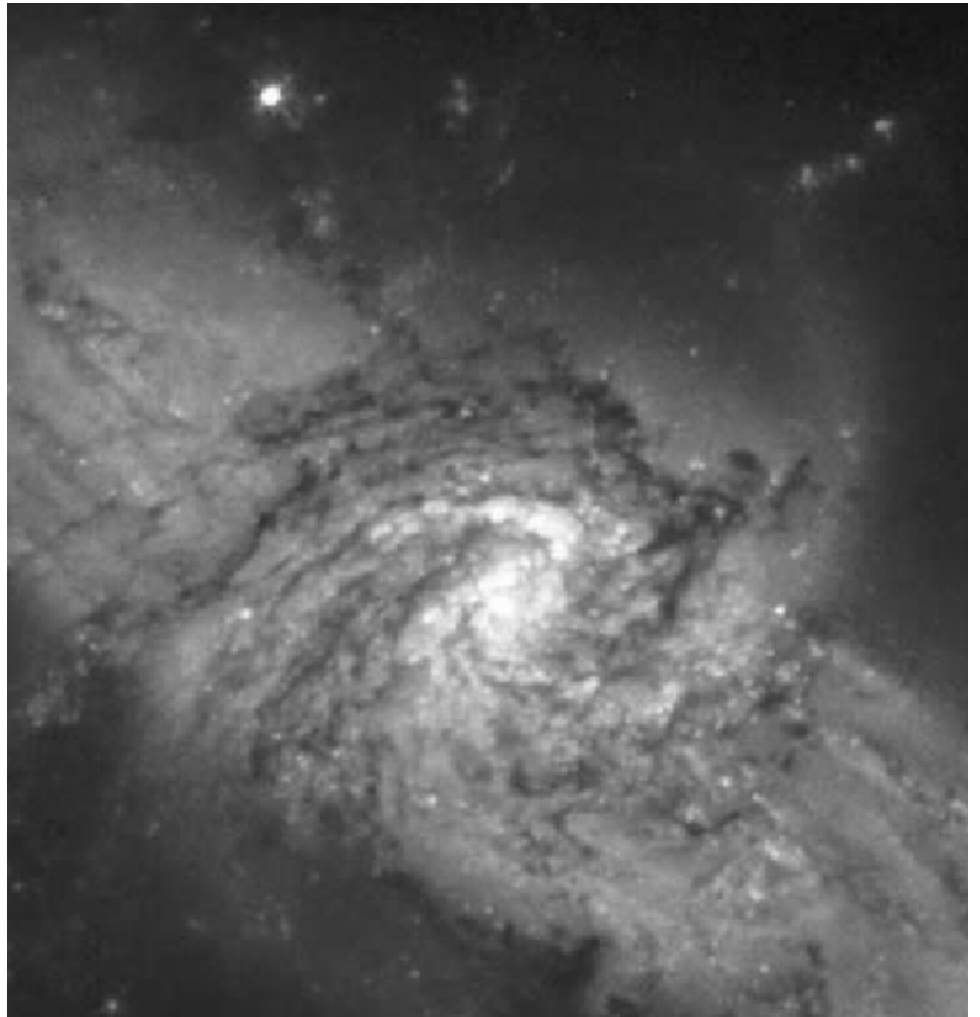


**Scilab code AP 86**  
Image Multiplication for Shading Correction



Scilab code AP 87  
Image Subtraction for Enhancing differences

---



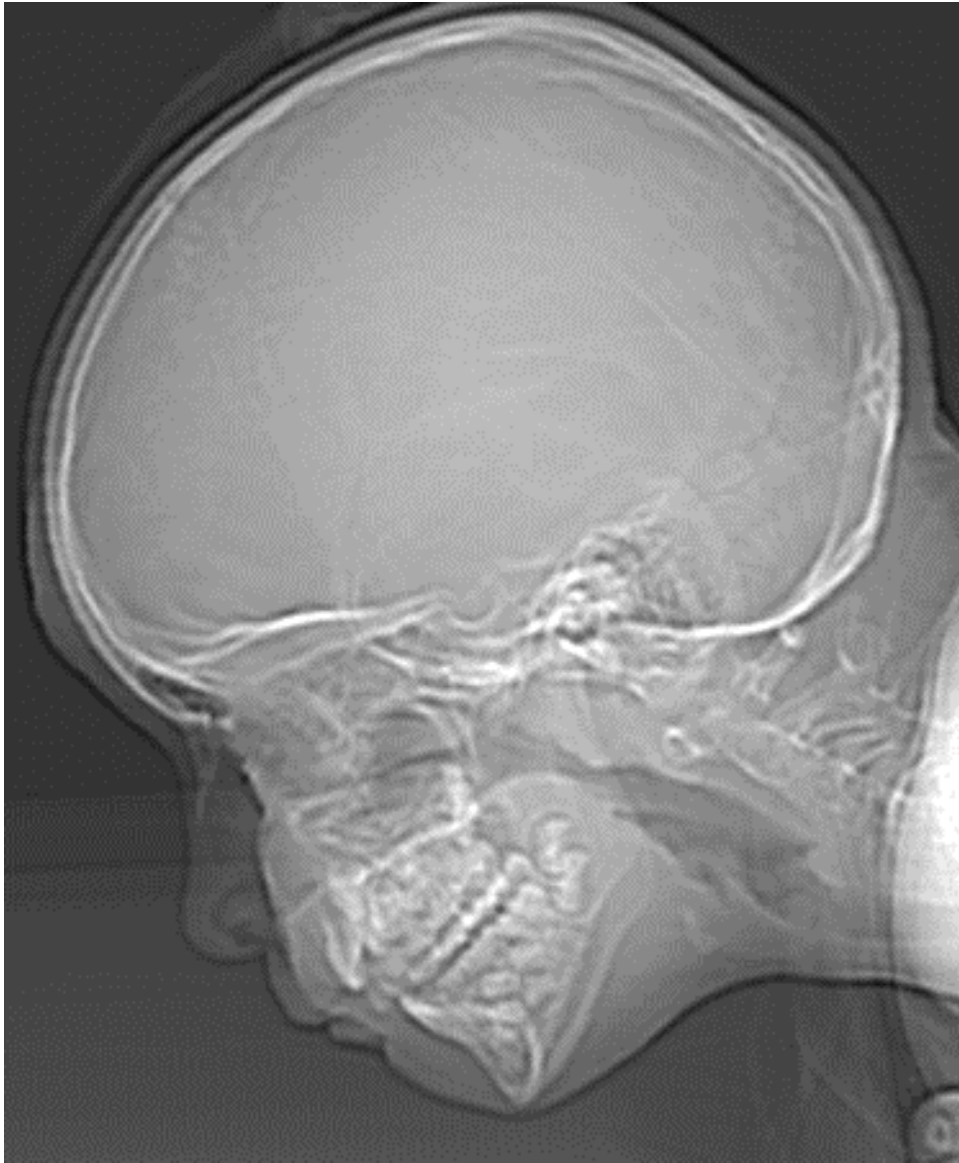
**Scilab code AP 88**

Addition of Noisy Images for Noise Reduction



Scilab code AP 89

Comparison of Interpolation Approaches for Image Shrinking and Zooming



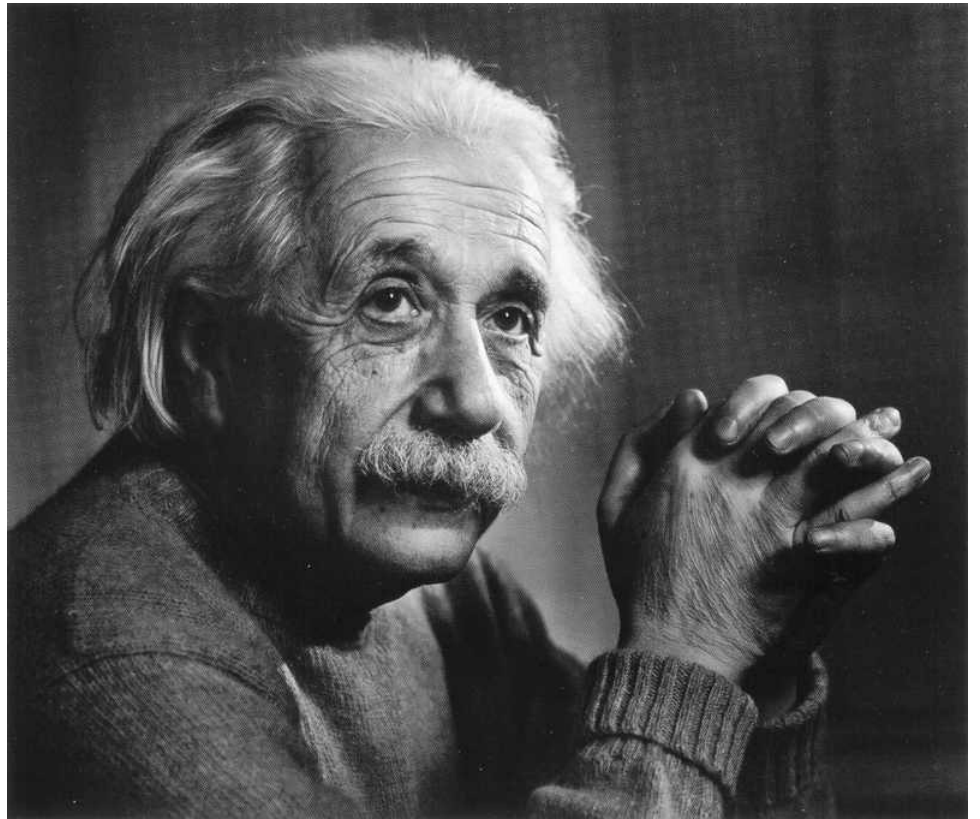
Typical Effects of Varying the Number of Intensity Levels in a digital Image





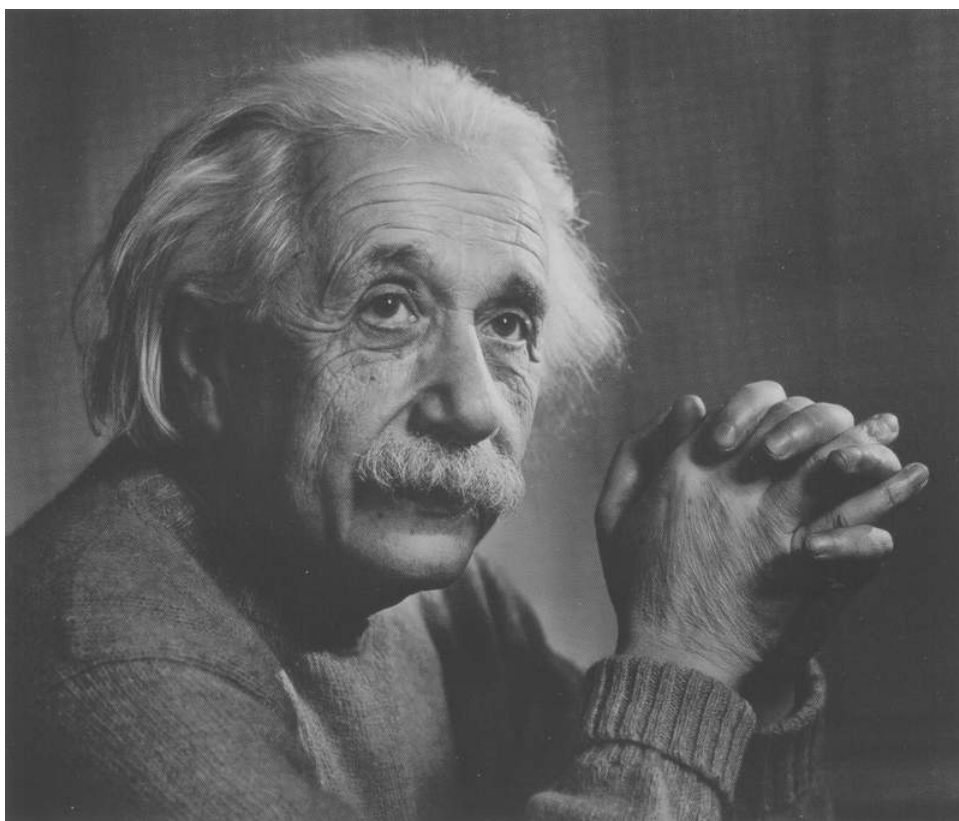
Scilab code AP 91

Illustration of the Effects of Reducing Image Spatial Resolution



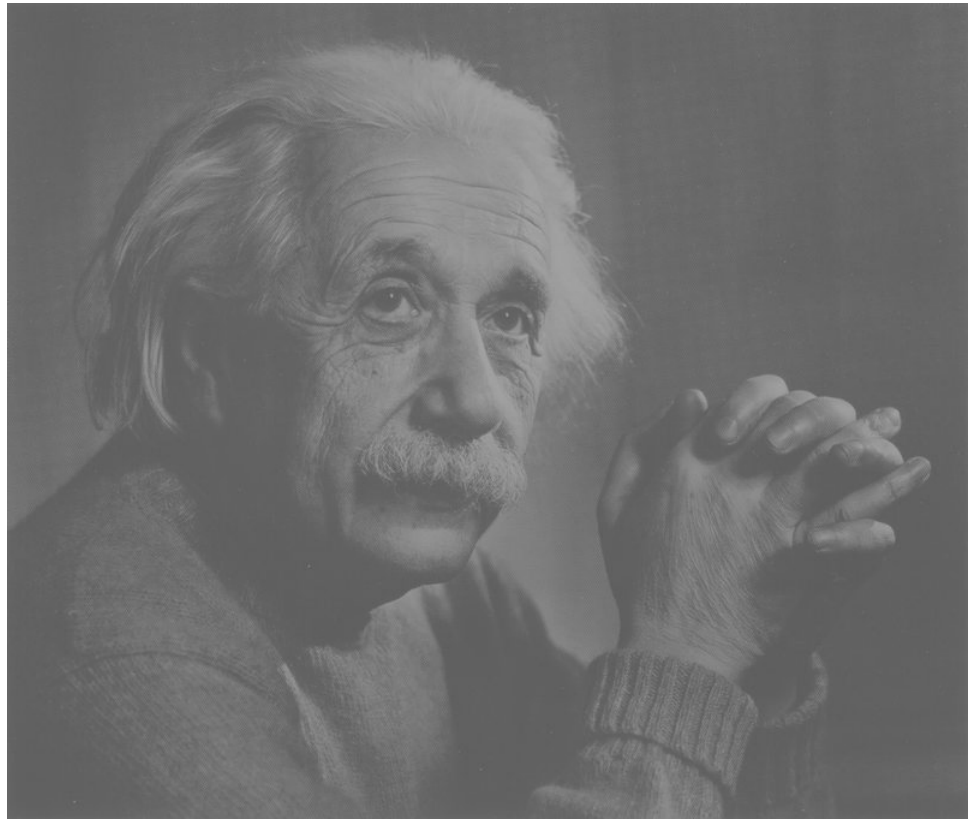
Scilab code AP 92  
Standard Deviation

---



**Scilab code AP 93**  
Standard Deviation

---



**Scilab code AP 94**  
Standard Deviation

---