

Scilab Textbook Companion for  
Digital Image Processing  
by S. Jayaraman, S. Esakkirajan And T.  
Veerakumar<sup>1</sup>

Created by  
R.Senthilkumar  
M.E., Assistant Professor  
Electronics Engineering  
Inst. of Road and Transport Tech., Erode  
College Teacher  
Dr. R.K.Gnanamurthy, Principal, V. C. E. W.  
Cross-Checked by  
Anuradha Amrutkar, IIT Bombay

July 31, 2019

<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:** S. Jayaraman, S. Esakkirajan And T. Veerakumar

**Publisher:** Tata McGraw - Hill Education Pvt. Ltd, New Delhi

**Edition:** 3

**Year:** 2010

**ISBN:** 978-0-07-014479-8

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
1 Introduction to Image Processing System	6
2 2D Signals and Systems	9
3 Convolution and Correlation	12
4 Image Transforms	20
5 Image Enhancement	30
6 Image Restoration and Denoising	40
7 Image Segmentation	60
8 Object Recognition	69
9 Image Compression	73
10 Binary Image Processing	77
11 Colour Image Processing	81
12 Wavelet based Image Processing	95

# List of Scilab Codes

Exa 1.3	Program to calculate number of samples required for an image . . . . .	6
Exa 1.13	False contouring Scilab code . . . . .	7
Exa 2.12	Frequency Response . . . . .	9
Exa 2.16	Frequency Response . . . . .	9
Exa 3.1	2D Linear Convolution . . . . .	12
Exa 3.2	2D Linear Convolution . . . . .	12
Exa 3.3	2D Linear Convolution . . . . .	13
Exa 3.7	2D Linear Convolution . . . . .	14
Exa 3.8	2D Linear Convolution . . . . .	14
Exa 3.11	Linear Convolution of any signal with an impulse signal given rise to the same signal . . . . .	15
Exa 3.12	Circular Convolution between two 2D matrices . . . . .	15
Exa 3.13	Circular Convolution expressed as linear convolution plus alias . . . . .	16
Exa 3.14	Linear Cross correlation of a 2D matrix . . . . .	17
Exa 3.15	Circular correlation between two signals . . . . .	17
Exa 3.16	Circular correlation between two signals . . . . .	18
Exa 3.17	Linear auto correlation of a 2D matrix . . . . .	18
Exa 3.18	Linear Cross correlation of a 2D matrix . . . . .	19
Exa 4.4	DFT of 4x4 grayscale image . . . . .	20
Exa 4.5	2D DFT of 4X4 grayscale image . . . . .	20
Exa 4.6	Scilab code to intergchange phase information between two images . . . . .	21
Exa 4.10	Program to compute discrete cosine transform . . . . .	22
Exa 4.12	Program to perform KL tranform for the given 2D matrix . . . . .	24

Exa 4.13	Program to find the singular value decomposition of given matrix . . . . .	28
Exa 5.5	Scilab code for brightness enhancement . . .	30
Exa 5.7	Scilab code for brightness suppression . . . .	32
Exa 5.9	Scilab code for Contrast Manipulation . . . .	32
Exa 5.13	Scilab code to determine image negative . . .	34
Exa 5.16	Scilab code that performs threshold operation	34
Exa 5.20	Program performs gray level slicing without background . . . . .	36
Exa 6.1	Scilab code to create motion blur . . . . .	40
Exa 6.5	Scilab code performs inverse filtering . . . . .	41
Exa 6.7	Scilab code performs inverse filtering . . . . .	44
Exa 6.9	Scilab code performs Pseudo inverse filtering	46
Exa 6.13	Scilab code to perform wiener filtering of the corrupted image . . . . .	50
Exa 6.18	Scilab code to Perform Average Filtering operation . . . . .	51
Exa 6.21	Scilab code to Perform median filtering . . .	53
Exa 6.23	Scilab code to Perform median filtering of colour image . . . . .	55
Exa 6.24	Scilab code to Perform Trimmed Average Filter . . . . .	57
Exa 7.23	Scilab code for Differentiation of Gaussian function . . . . .	60
Exa 7.25	Scilab code for Differentiation of Gaussian Filter function . . . . .	63
Exa 7.27	Scilab code for Edge Detection using Different Edge detectors . . . . .	65
Exa 7.30	Scilab code to perform watershed transform	65
Exa 8.4	To verify the given matrix is a covariance matrix . . . . .	69
Exa 8.5	To compute the covariance of the given 2D data . . . . .	69
Exa 8.9	Develop a perceptron AND function with bipolar inputs and targets . . . . .	71
Exa 9.9	Program performs Block Truncation Coding BTC . . . . .	73
Exa 9.59	Program performs Block Truncation Coding	75

Exa 10.17	Scilab Code for dilation and erosion process	77
Exa 10.19	Scilab Code to perform an opening and closing operation on the image . . . . .	78
Exa 11.4	Read an RGB image and extract the three colour components red green blue . . . . .	81
Exa 11.12	Read a Colour image and separate the colour image into red green and blue planes . . . . .	82
Exa 11.16	Compute the histogram of the colour image	84
Exa 11.18	Perform histogram equalisation of the given RGB image . . . . .	86
Exa 11.21	This program performs median filtering of the colour image . . . . .	87
Exa 11.24	Filtering only the luminance component . . .	89
Exa 11.28	Perform gamma correction for the given colour image . . . . .	90
Exa 11.30	Perform Pseudo Colouring Operation . . . . .	91
Exa 11.32	Read an RGB image and segment it using the threshold method . . . . .	93
Exa 12.9	Scilab code to perform wavelet decomposition	95
Exa 12.42	Scilab code to generate different levels of a Gaussian pyramid . . . . .	96
Exa 12.57	Scilab code to implement watermarking in spatial domain . . . . .	97
Exa 12.63	Scilab code to implement wavelet based watermarking . . . . .	100

# List of Figures

1.1	False contouring Scilab code . . . . .	8
2.1	Frequency Response . . . . .	10
2.2	Frequency Response . . . . .	11
4.1	Scilab code to intergchange phase information between two images . . . . .	23
4.2	Program to compute discrete cosine transform . . . . .	25
4.3	Program to compute discrete cosine transform . . . . .	26
5.1	Scilab code for brightness enhancement . . . . .	31
5.2	Scilab code for brightness suppression . . . . .	33
5.3	Scilab code for Contrast Manipulation . . . . .	35
5.4	Scilab code to determine image negative . . . . .	35
5.5	Scilab code that performs threshold operation . . . . .	37
5.6	Program performs gray level slicing without background . . . . .	39
6.1	Scilab code to create motion blur . . . . .	41
6.2	Scilab code performs inverse filtering . . . . .	43
6.3	Scilab code performs inverse filtering . . . . .	45
6.4	Scilab code performs inverse filtering . . . . .	47
6.5	Scilab code performs Pseudo inverse filtering . . . . .	49
6.6	Scilab code to perform wiener filtering of the corrupted image . . . . .	52
6.7	Scilab code to Perform Average Filtering operation . . . . .	54



6.8	Scilab code to Perform median filtering . . . . .	56
6.9	Scilab code to Perform median filtering of colour image . . . . .	57
6.10	Scilab code to Perform Trimmed Average Filter . . . . .	59
7.1	Scilab code for Differentiation of Gaussian function . . . . .	61
7.2	Scilab code for Differentiation of Gaussian function . . . . .	62
7.3	Scilab code for Differentiation of Gaussian Filter function . . . . .	63
7.4	Scilab code for Differentiation of Gaussian Filter function . . . . .	64
7.5	Scilab code for Edge Detection using Different Edge detectors . . . . .	66
7.6	Scilab code to perform watershed transform . . . . .	68
9.1	Program performs Block Truncation Coding . . . . .	76
10.1	Scilab Code for dilation and erosion process . . . . .	78
10.2	Scilab Code to perform an opening and closing operation on the image . . . . .	80
11.1	Read an RGB image and extract the three colour components red green blue . . . . .	83
11.2	Read a Colour image and separate the colour image into red green and blue planes . . . . .	85
11.3	This program performs median filtering of the colour image . . . . .	88
11.4	Filtering only the luminance component . . . . .	90
11.5	Perform gamma correction for the given colour image . . . . .	91
11.6	Perform Pseudo Colouring Operation . . . . .	93
11.7	Read an RGB image and segment it using the threshold method . . . . .	94
12.1	Scilab code to generate different levels of a Gaussian pyramid . . . . .	98
12.2	Scilab code to implement watermarking in spatial domain . . . . .	100

# Chapter 1

## Introduction to Image Processing System

Scilab code Exa 1.3 Program to calculate number of samples required for an image

```
1 //Caption:Program to calculate number of samples
   required for an image
2 //Example1.3
3 //page 12
4 clc;
5 close;
6 //dimension of the image in inches
7 m = 4;
8 n = 6;
9 N = 400; //number of dots per inch in each direction
10 N2 = 2*N; //number of dots per inch in both
   horizontal & vertical
11 Fs = m*N2*n*N2;
12 disp(Fs, 'Number of samples required to preserve the
   information in the image=')
13 //Result
14 //Number of samples required to preserve the
   information in the image=
15 //15360000.
```

---

check Appendix [AP 4](#) for dependency:

gray.sci

check Appendix [AP 5](#) for dependency:

grayscale.sci

### Scilab code Exa 1.13 False contouring Scilab code

```
1 //Caption: False contouring Scilab code
2 //Fig1.13
3 //page 13
4 clc;
5 close;
6 a =ReadImage('E:\DIP_JAYARAMAN\Chapter1\tigerpub.jpg
7 ');
8 a = uint8(a);
9 figure
10 imshow(a)
11 title('Original image');
12 //using 128 gray levels
13 figure
14 a_128 = grayscale(a,128);
15 gray_128 = gray(128);
16 ShowImage(a_128,'Image with 128 gray levels',
17 gray_128);
18 //using 64 gray levels
19 figure
20 a_64 = grayscale(a,64);
21 gray_64 = gray(64);
22 ShowImage(a_64,'Image with 64 gray levels',gray_64);
23 //using 32 gray levels
24 figure
25 a_32 = grayscale(a,32);
```

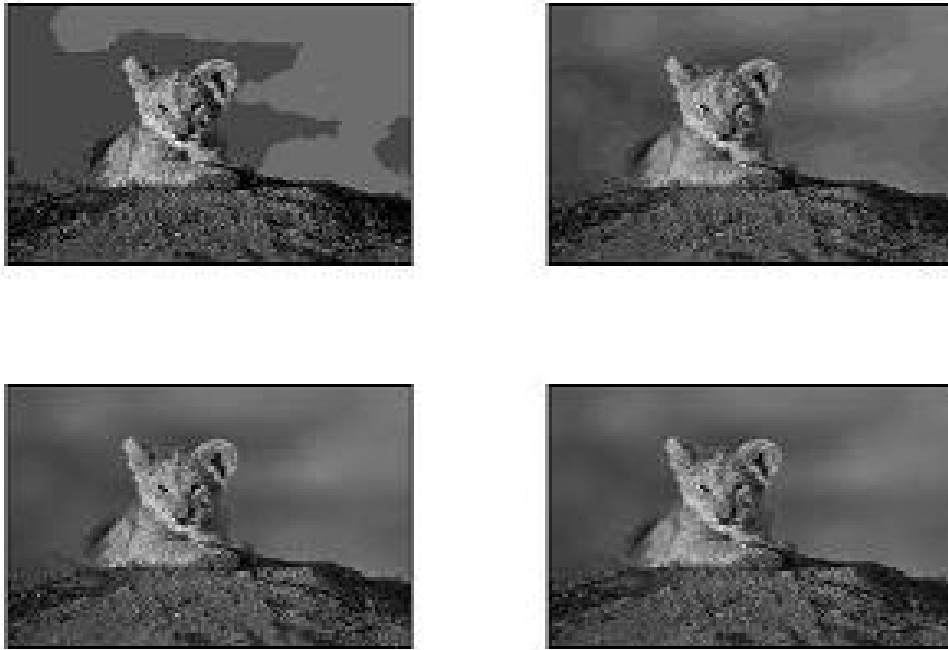


Figure 1.1: False contouring Scilab code

```
24 gray_32 = gray(32);
25 ShowImage(a_32, 'Image with 32 gray levels', gray_32);
26 //using 16 gray levels
27 figure
28 a_16 = grayslice(a,16);
29 gray_16 = gray(16);
30 ShowImage(a_16, 'Image with 16 gray levels', gray_16);
31 //using 8 gray levels
32 a_8 = grayslice(a,8);
33 gray_8 = gray(8);
34 ShowImage(a_8, 'Image with 8 gray levels', gray_8);
```

---

# Chapter 2

## 2D Signals and Systems

Scilab code Exa 2.12 Frequency Response

```
1 //Caption: Frequency Response
2 //Fig2.12
3 //page 60
4 clc;
5 close;
6 [X, Y] = meshgrid(-%pi:.09:%pi);
7 Z = 2*cos(X)+2*cos(Y);
8 surf(X,Y,Z);
9 xgrid(1)
```

---

Scilab code Exa 2.16 Frequency Response

```
1 //Caption: Frequency Response
2 //Fig2.16
3 //page 64
```

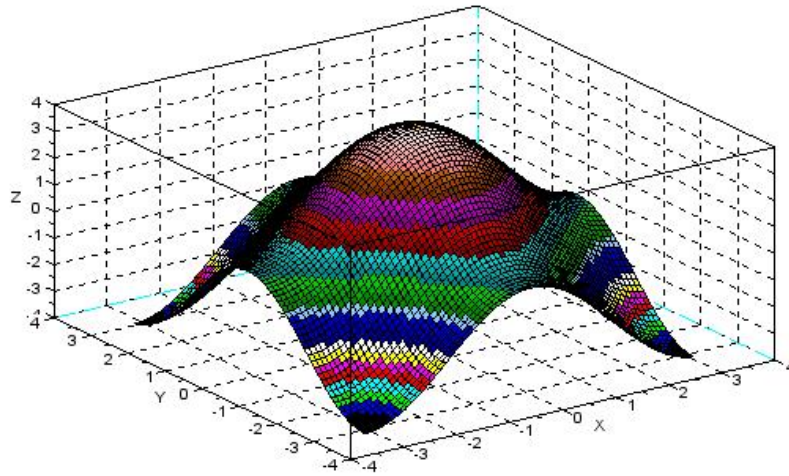


Figure 2.1: Frequency Response

```
4 clc;  
5 close;  
6 [X, Y] = meshgrid(-%pi:.05:%pi);  
7 Z = 2-cos(X)-cos(Y);  
8 surf(X,Y,Z);  
9 xgrid(1)
```

---

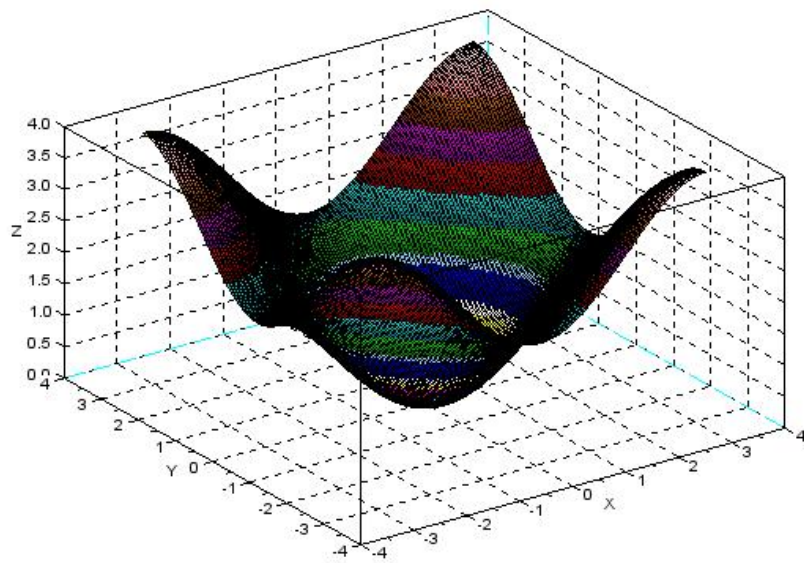


Figure 2.2: Frequency Response

# Chapter 3

## Convolution and Correlation

Scilab code Exa 3.1 2D Linear Convolution

```
1 //Caption: 2-D Linear Convolution
2 //Example3.1 & Example3.4
3 //page 85 & page 107
4 clc;
5 x =[4,5,6;7,8,9];
6 h = [1;1;1];
7 disp(x, 'x=')
8 disp(h, 'h=')
9 [y,X,H] = conv2d2(x,h);
10 disp(y, 'Linear 2D convolution result y =')
11 //Result
12 //Linear 2D convolution result y =
13 //
14 //      4.      5.      6.
15 //      11.     13.     15.
16 //      11.     13.     15.
17 //      7.      8.      9.
```

---

Scilab code Exa 3.2 2D Linear Convolution



```

1 //Caption: 2-D Linear Convolution
2 //Example3.2 & Example3.5 & Example3.9
3 //page 91 & page 108 & page 116
4 clc;
5 x =[1,2,3;4,5,6;7,8,9];
6 h = [1,1;1,1;1,1];
7 y = conv2d2(x,h);
8 disp(y,'Linear 2D convolution result y =')
9 //Result
10 // Linear 2D convolution result y =
11 //
12 //      1.      3.      5.      3.
13 //      5.      12.     16.     9.
14 //      12.     27.     33.     18.
15 //      11.     24.     28.     15.
16 //      7.      15.     17.     9.
17 //

```

---

### Scilab code Exa 3.3 2D Linear Convolution

```

1 //Caption: 2-D Linear Convolution
2 //Example3.3 & Example3.6 & Example3.10
3 //page 100 & page 109 & page 119
4 clc;
5 x =[1,2,3;4,5,6;7,8,9];
6 h = [3,4,5];
7 y = conv2d2(x,h);
8 disp(y,'Linear 2D convolution result y =')
9 //Result
10 //Linear 2D convolution result y =
11 //
12 //      3.      10.     22.     22.     15.
13 //      12.     31.     58.     49.     30.
14 //      21.     52.     94.     76.     45.

```

---

### Scilab code Exa 3.7 2D Linear Convolution

```
1 //Caption: 2-D Linear Convolution
2 //Example3.7
3 //page 111
4 clc;
5 x =[1,2;3,4];
6 h = [5,6;7,8];
7 y = conv2d2(x,h);
8 disp(y,'Linear 2D convolution result y =')
9 //Result
10 // Linear 2D convolution result y =
11 //Linear 2D convolution result y =
12 //
13 //      5.      16.      12.
14 //      22.      60.      40.
15 //      21.      52.      32
```

---

### Scilab code Exa 3.8 2D Linear Convolution

```
1 //Caption: 2-D Linear Convolution
2 //Example3.8
3 //page 113
4 clc;
5 x =[1,2,3;4,5,6;7,8,9];
6 h = [1;1;1];
7 y = conv2d2(x,h);
8 disp(y,'Linear 2D convolution result y =')
9 //Result
10 // Linear 2D convolution result y =
11 ///// 1.      2.      3.
12 //      5.      7.      9.
```

```
13 //      12.      15.      18.
14 //      11.      13.      15.
15 //      7.       8.       9.
```

---

**Scilab code Exa 3.11** Linear Convolution of any signal with an impulse signal given

```
1 //Caption: Linear CONvolution of any signal with an
  //impulse signal gives
2 //rise to the same signal
3 //Example3.11
4 //page 121
5 clc;
6 x =[1,2;3,4];
7 h = 1;
8 y = conv2d2(x,h);
9 disp(y,'Linear 2D convolution result y =')
10 //Result
11 //Linear 2D convolution result y =
12 //// Linear 2D convolution result y =
13 //
14 //      1.      2.
15 //      3.      4.
```

---

**Scilab code Exa 3.12** Circular Convolution between two 2D matrices

```
1 //Caption: Circular Convolution between two 2D
  //matrices
2 //Example3.12
3 //page 122
4 clc;
5 x = [1,2;3,4];
6 h = [5,6;7,8];
7 X = fft2d(x); //2D FFT of x matrix
```

```

8 H = fft2d(h); //2D FFT of h matrix
9 Y = X.*H; //Element by Element multiplication
10 y = ifft2d(Y);
11 disp(y,'Circular Convolution Result y =')
12 //Result
13 //Circular Convolution Result y =
14 //
15 //      70.      68.
16 //      62.      60.

```

---

**Scilab code Exa 3.13** Circular Convolution expressed as linear convolution plus alias

```

1 //Caption: Circular Convolution expressed as linear
   convolution plus alias
2 //Example3.13
3 //page 123
4 clc;
5 x = [1,2;3,4];
6 h = [5,6;7,8];
7 y = conv2d(x,h);
8 y1 = [y(:,1)+y(:,2),y(:,2)];
9 y2 = [y1(1,:)+y1(2,:);y1(2,:)]
10 disp(y,'Linear Convolution result y=')
11 disp(y2,'circular convolution expressed as linear
   convolution plus alias =')
12 //Result
13 // Linear Convolution result y=
14 //
15 //      5.      16.      12.
16 //      22.      60.      40.
17 //      21.      52.      32.
18 //
19 // circular convolution expressed as linear
   convolution plus alias =
20 //

```

```
21 //      70.      68.
22 //      62.      60.
23 //
```

---

**Scilab code Exa 3.14** Linear Cross correlation of a 2D matrix

```
1 //Caption: linear cross correlation of a 2D matrix
2 //Example3.14
3 //page 129
4 clc;
5 x = [3,1;2,4];
6 h1 = [1,5;2,3];
7 h2 = h1(:, $:-1:1);
8 h = h2($:-1:1, :);
9 y = conv2d(x,h)
10 disp(y, 'Linear cross Correlation result y=')
11 //Result
12 //Linear cross Correlation result y=
13 //
14 //      9.      9.      2.
15 //      21.     24.     9.
16 //      10.     22.     4.
```

---

**Scilab code Exa 3.15** Circular correlation between two signals

```
1 //Caption: Circular correlation between two signals
2 //Example3.15
3 //page 131
4 clc;
5 x = [1,5;2,4];
6 h = [3,2;4,1];
7 h = h(:, $:-1:1);
8 h = h($:-1:1, :);
```

```

 9 X = fft2d(x);
10 H = fft2d(h);
11 Y = X.*H;
12 y = ifft2d(Y);
13 disp(y, 'Circular Correlation result y=')
14 //Result
15 //Circular Correlation result y=
16 //
17 //      37.      23.
18 //      35.      25.

```

---

**Scilab code Exa 3.16** Circular correlation between two signals

```

1 //Caption: Circular correlation between two signals
2 //Example3.16
3 //page 134
4 clc;
5 x = [5,10;15,20];
6 h = [3,6;9,12];
7 h = h(:, $:-1:1);
8 h = h($:-1:1, :);
9 X = fft2d(x);
10 H = fft2d(h);
11 Y = X.*H;
12 y = ifft2d(Y);
13 disp(y, 'Circular Correlation result y=')
14 //Result
15 // Circular Correlation result y=
16 //
17 //      300.      330.
18 //      420.      450.

```

---

**Scilab code Exa 3.17** Linear auto correlation of a 2D matrix

```

1 //Caption: linear auto correlation of a 2D matrix
2 //Example3.17
3 //page 136
4 clc;
5 x1 = [1,1;1,1];
6 x2 = x1(:, $:-1:1);
7 x2 = x2($:-1:1, :);
8 x = conv2d(x1,x2)
9 disp(x, 'Linear auto Correlation result x=')
10 //Result
11 //Linear auto Correlation result x=
12 //
13 //      1.      2.      1.
14 //      2.      4.      2.
15 //      1.      2.      1.

```

---

**Scilab code Exa 3.18** Linear Cross correlation of a 2D matrix

```

1 //Caption: linear cross correlation of a 2D matrix
2 //Example3.18
3 //page 141
4 clc;
5 x = [1,1;1,1];
6 h1 = [1,2;3,4];
7 h2 = h1(:, $:-1:1);
8 h = h2($:-1:1, :);
9 y = conv2d(x,h)
10 disp(y, 'Linear cross Correlation result y=')
11 //Result
12 //Linear cross Correlation result y=
13 //
14 //      4.      7.      3.
15 //      6.     10.      4.
16 //      2.      3.      1.

```

---

# Chapter 4

## Image Transforms

Scilab code Exa 4.4 DFT of 4x4 grayscale image

```
1 //Caption: 2D DFT of 4x4 grayscale image
2 //Example4.4
3 //page 170
4 clc;
5 f = [1,1,1,1;1,1,1,1;1,1,1,1;1,1,1,1];
6 N =4; //4-point DFT
7 kernel = dft_mtx(N);
8 F = kernel*(f*kernel');
9 disp(F,'2D DFT of given 2D image =')
10 //Result
11 //2D DFT of given 2D image =
12 //
13 //      16.      0      0      0
14 //      0       0      0      0
15 //      0       0      0      0
16 //      0       0      0      0
```

---

Scilab code Exa 4.5 2D DFT of 4X4 grayscale image



```

1 //Caption: 2D DFT of 4x4 grayscale image
2 //Example4.5
3 //page 171
4 clc;
5 F = [16,0,0,0;0,0,0,0;0,0,0,0;0,0,0,0];
6 N =4; //4-point DFT
7 kernel = dft_mtx(N);
8 f = (kernel*(F*kernel'))/(N^2);
9 f = real(f);
10 disp(f, 'Inverse 2D DFT of the transformed image f ='
    )
11 //Result
12 //Inverse 2D DFT of the transformed image f =
13 //
14 //      1.      1.      1.      1.
15 //      1.      1.      1.      1.
16 //      1.      1.      1.      1.
17 //      1.      1.      1.      1.

```

---

check Appendix [AP 1](#) for dependency:

fft2d.sce

check Appendix [AP 2](#) for dependency:

ifft2d.sce

**Scilab code Exa 4.6** Scilab code to intergchange phase information between two imag

```

1 //Caption:Scilab code to intergchange phase
  information between two images
2 //Example4.6
3 //page 174-175
4 clc;
5 close;
6 a = imread('E:\DIP_JAYARAMAN\Chapter4\lena.png');
  //SIVP toolbox

```

```

7 b = imread('E:\DIP_JAYARAMAN\Chapter4\baboon.png');
8 a = rgb2gray(a);
9 b = rgb2gray(b);
10 a = imresize(a,0.5);
11 b = imresize(b,0.5);
12 figure(1)
13 ShowImage(a,'Original lena Image'); //IPD toolbox
14 title('Original lena Image');
15 figure(2)
16 ShowImage(b,'Original baboon Image');
17 title('Original baboon Image')
18 ffta = fft2d(double(a));
19 fftb = fft2d(double(b));
20 mag_a = abs(ffta);
21 mag_b = abs(fftb);
22 ph_a = atan(imag(ffta),real(ffta));
23 ph_b = atan(imag(fftb),real(fftb));
24 newfft_a = mag_a.*(exp(%i*ph_b));
25 newfft_b = mag_b.*(exp(%i*ph_a));
26 rec_a = ifft2d(newfft_a);
27 rec_b = ifft2d(newfft_b);
28 figure(3)
29 ShowImage(uint8(rec_a),'lena Image after phase
    reversal');
30 title('lena Image after phase reversal')
31 figure(4)
32 ShowImage(uint8(rec_b),'baboon Image after phase
    reversal');
33 title('baboon Image after phase reversal')

```

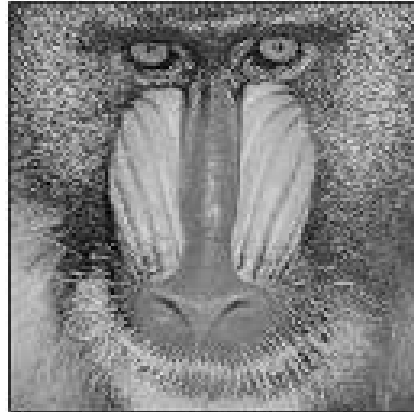
---

Scilab code Exa 4.10 Program to compute discrete cosine transform

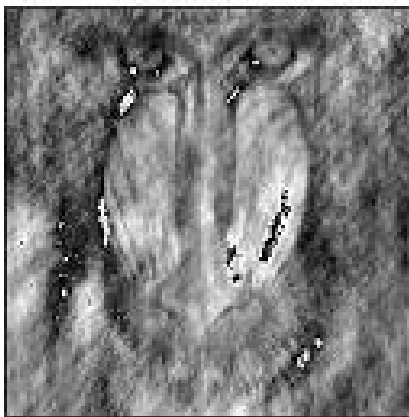
Original lena image



Original baboon image



lena image after phase reversal



baboon image after phase reversal



Figure 4.1: Scilab code to intergchange phase information between two images

```

1 //Caption: Program to compute discrete cosine
  transform
2 //Example4.10
3 //page 198
4 clc;
5 N =4; //DCT matrix of order four
6 X = dct_mtx(N);
7 disp(X, 'DCT matrix of order four')
8 //Result
9 //DCT matrix of order four
10 //
11 //      0.5          0.5          0.5          0.5
12 //      0.6532815    0.2705981  - 0.2705981  -
13 //      0.6532815
14 //      0.5          - 0.5          - 0.5          0.5
15 //      0.2705981  - 0.6532815    0.6532815  -
16 //      0.2705981

```

---

**Scilab code Exa 4.12** Program to perform KL tranform for the given 2D matrix

```

1 //Caption: Program to perform KL transform for the
  given 2D matrix
2 //Example4.12
3 //page 208
4 clear;
5 clc;
6 X = [4,3,5,6;4,2,7,7;5,5,6,7];
7 [m,n]= size(X);
8 A = [];
9 E = [];
10 for i =1:n

```

lena Image after phase reversal

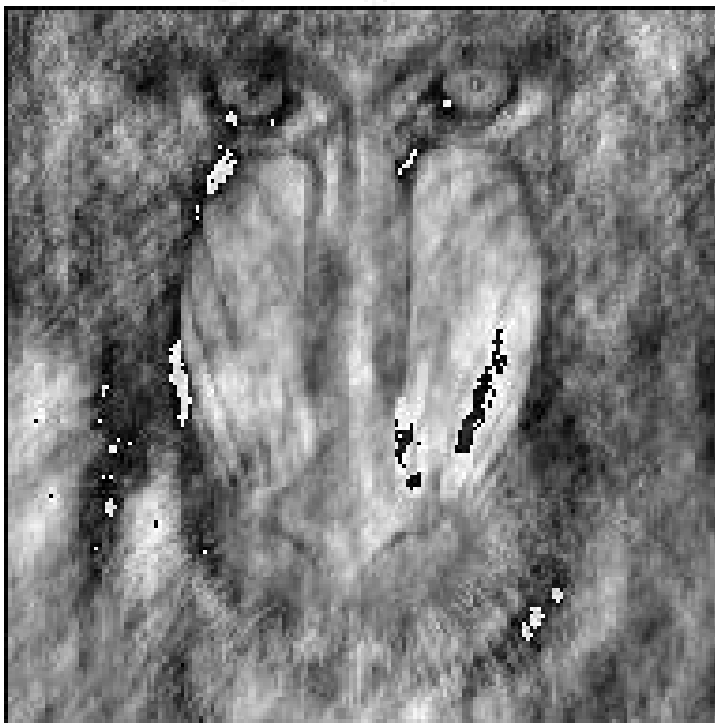


Figure 4.2: Program to compute discrete cosine transform

**baboon Image after phase reversal**



Figure 4.3: Program to compute discrete cosine transform

```

11     A = A+X(:,i);
12     E = E+X(:,i)*X(:,i)';
13 end
14 mx = A/n;    //mean matrix
15 E = E/n;
16 C = E - mx*mx'; //covariance matrix C = E[xx']-mx*mx'
17 [V,D] = spec(C); //eigen values and eigen vectors
18 d = diag(D); //diagonal elements od eigen values
19 [d,i] = gsort(d); //sorting the elements of D in
    descending order
20 for j = 1:length(d)
21     T(:,j)= V(:,i(j));
22 end
23 T =T'
24 disp(d,'Eigen Values are U = ')
25 disp(T,'The eigen vector matrix T =')
26 disp(T,'The KL tranform basis is =')
27 //KL transform
28 for i = 1:n
29     Y(:,i)= T*X(:,i);
30 end
31 disp(Y,'KL transformation of the input matrix Y =')
32 //Reconstruction
33 for i = 1:n
34     x(:,i)= T'*Y(:,i);
35 end
36 disp(x,'Reconstruct matrix of the given sample
    matrix X =')
37 //Result
38 // Eigen Values are U =
39 //     6.1963372
40 //     0.2147417
41 //     0.0264211
42 // The eigen vector matrix T =
43 //     0.4384533     0.8471005     0.3002988
44 //     0.4460381 - 0.4951684     0.7455591
45 // - 0.7802620     0.1929481     0.5949473
46 // The KL tranform basis is =

```

```

47 //      0.4384533      0.8471005      0.3002988
48 //      0.4460381  - 0.4951684      0.7455591
49 //  - 0.7802620      0.1929481      0.5949473
50 // KL transformation of the input matrix Y =
51 //      6.6437095      4.5110551      9.9237632
      10.662515
52 //      3.5312743      4.0755729      3.2373664
      4.4289635
53 //      0.6254808      1.0198466      1.0190104
      0.8336957
54 // Reconstruct matrix of the given sample matrix x =
55 //      4.      3.      5.      6.
56 //      4.      2.      7.      7.
57 //      5.      5.      6.      7.

```

---

**Scilab code Exa 4.13** Program to find the singular value decomposition of given mat

```

1 //Caption: Program to find the singular value
      decomposition of given matrix
2 //Example4.13
3 //page 210
4 clear;
5 clc;
6 A = [1,-2,3;3,2,-1];
7 [U,S,V]= svd(A);
8 A_recon = U*S*V';
9 disp(U, 'U =')
10 disp(S, 'S =')
11 disp(V, 'V =')
12 disp(A_recon, 'A matrix from svd =')
13 //Result
14 // U =
15 //
16 //  - 0.7071068      0.7071068
17 //      0.7071068      0.7071068

```



```
18 //
19 // S =
20 //
21 //      4.2426407      0.      0.
22 //      0.      3.1622777      0.
23 //
24 // V =
25 //
26 //      0.3333333      0.8944272 - 0.2981424
27 //      0.6666667 - 2.776D-16      0.7453560
28 // - 0.6666667      0.4472136      0.5962848
29 //
30 // A matrix from svd =
31 //
32 //      1. - 2.      3.
33 //      3.      2. - 1.
```

---

# Chapter 5

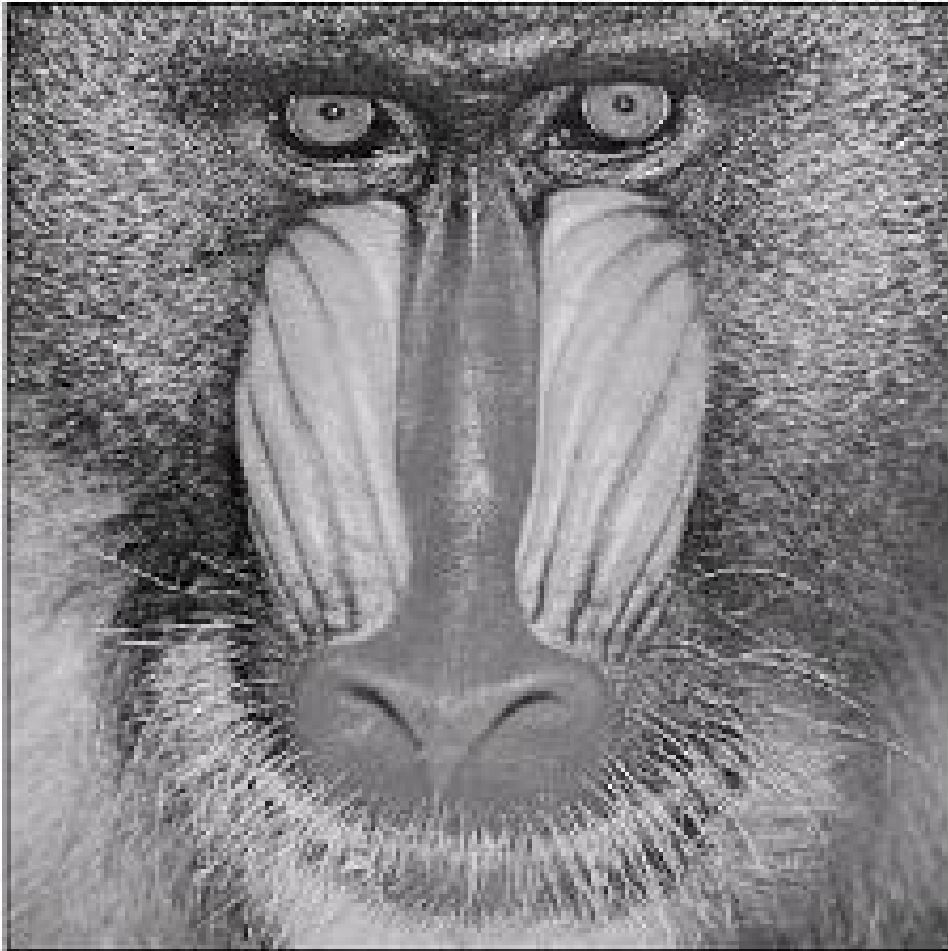
## Image Enhancement

Scilab code Exa 5.5 Scilab code for brightness enhancement

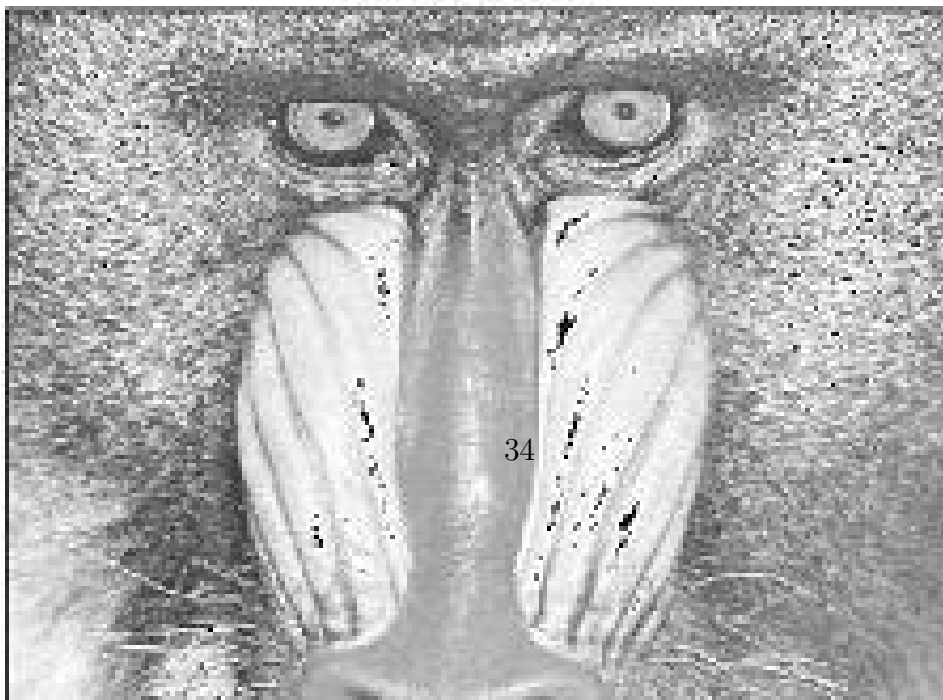
```
1 //Caption:Scilab code for brightness enhancement
2 //Fig5.5
3 //page 246
4 clc;
5 close;
6 //a = imread('E:\DIP_JAYARAMAN\Chapter5\plate.GIF');
   //SIVP toolbox
7 a = imread('E:\DIP_JAYARAMAN\Chapter4\baboon.png');
8 a = rgb2gray(a);
9 b = double(a)+50;
10 b = uint8(b);
11 figure(1)
12 ShowImage(a,'Original Image');
13 title('Original Image')
14 figure(2)
15 ShowImage(b,'Enhanced Image');
16 title('Enhanced Image')
```

---

Original Image



Enhanced Image



### Scilab code Exa 5.7 Scilab code for brightness suppression

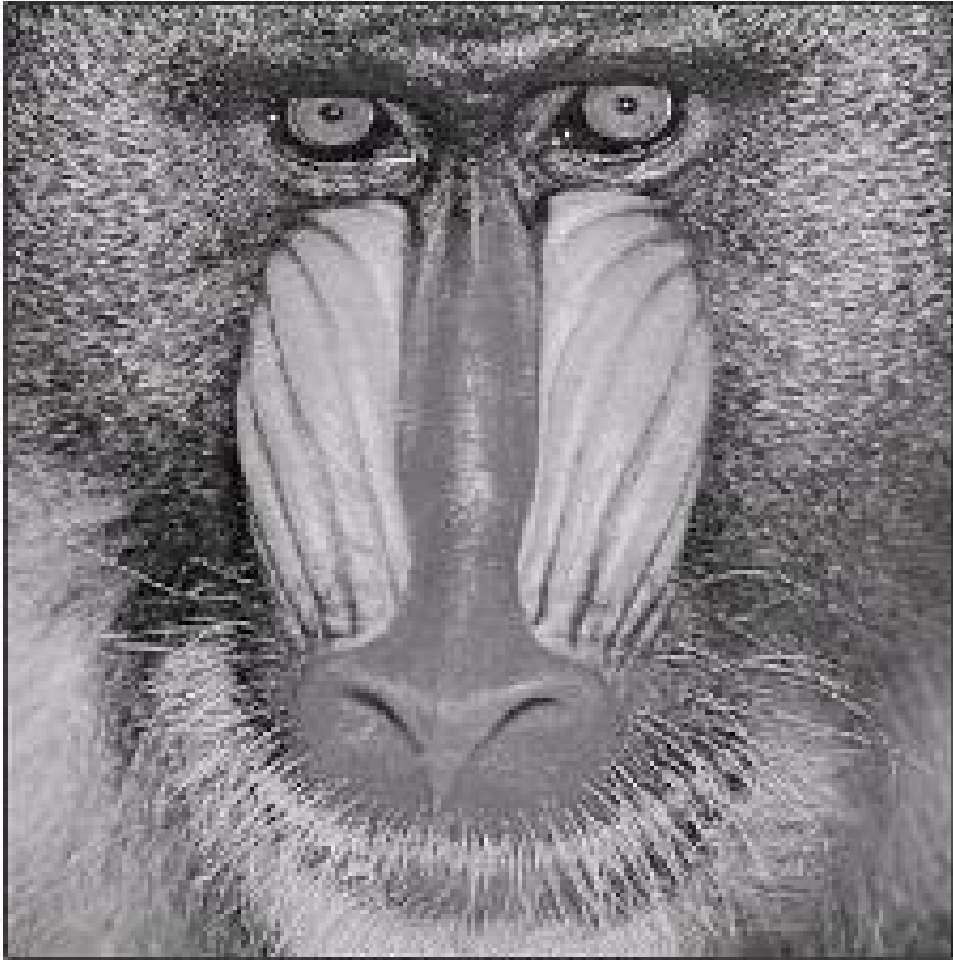
```
1 //Caption:Scilab code for brightness suppression
2 //Fig5.7
3 //page 247
4 clc;
5 close;
6 a = imread('E:\DIP_JAYARAMAN\Chapter4\baboon.png');
7 a = rgb2gray(a);
8 b = double(a)-50;
9 b = uint8(b);
10 figure(1)
11 ShowImage(a,'Original Image');
12 title('Original Image')
13 figure(2)
14 ShowImage(b,'Brightness Supressed Image');
15 title('Brightness Supressed Image')
```

---

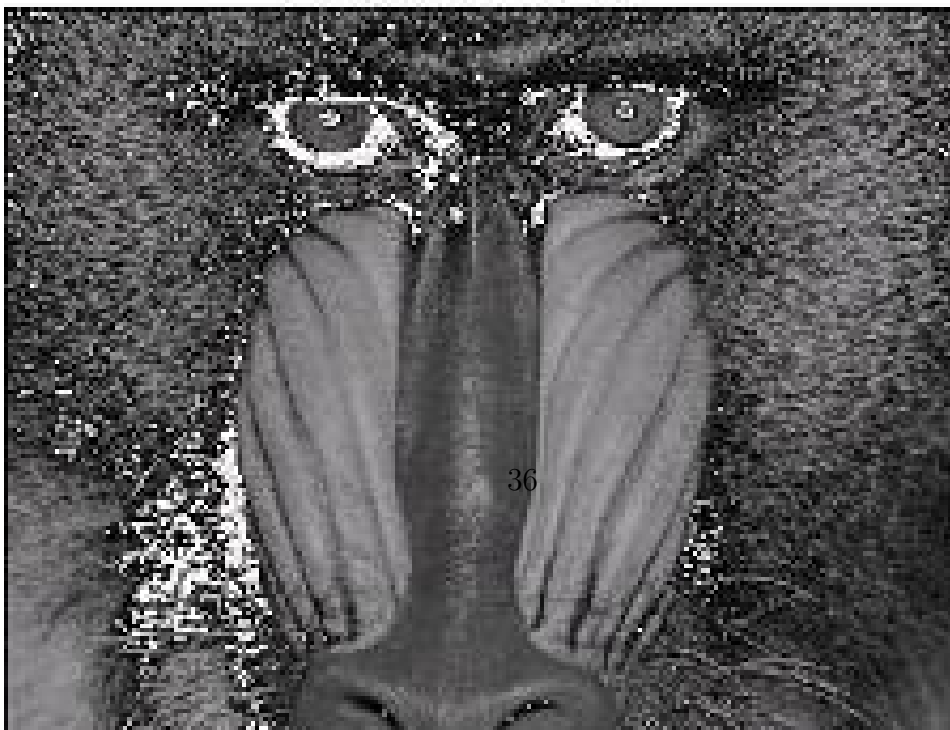
### Scilab code Exa 5.9 Scilab code for Contrast Manipulation

```
1 //Caption:Scilab code for Contrast Manipulation
2 //Fig5.9
3 //page 248
4 clc;
5 close;
6 a = imread('E:\DIP_JAYARAMAN\Chapter4\lena.png');
7 a = rgb2gray(a);
8 b = double(a)*0.5;
9 b = uint8(b)
10 c = double(b)*2;
```

Original Image



Brightness Supressed Image



```
11 c = uint8(c)
12 figure(1)
13 ShowImage(a, 'Original Image');
14 title('Original Image')
15 figure(2)
16 ShowImage(b, 'Decrease in Contrast');
17 title('Decrease in Contrast')
18 figure(3)
19 ShowImage(c, 'Increase in Contrast');
20 title('Increase in Contrast')
```

---

**Scilab code Exa 5.13** Scilab code to determine image negative

```
1 //Caption:Scilab code to determine image negative
2 //Fig.5.13
3 //page 252
4 clc;
5 close;
6 a = imread('E:\DIP_JAYARAMAN\Chapter5\label.jpg');
7 k = 255-double(a);
8 k = uint8(k);
9 imshow(a);
10 title('Original onca Image')
11 imshow(k);
12 title('Negative of Original Image')
```

---

**Scilab code Exa 5.16** Scilab code that performs threshold operation



Figure 5.3: Scilab code for Contrast Manipulation

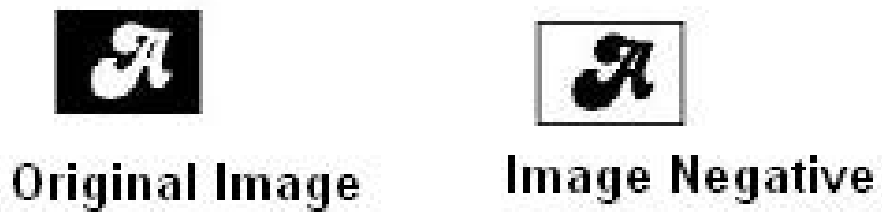


Figure 5.4: Scilab code to determine image negative

```

1 //Caption:Scilab code that performs threshold
  operation
2 //Fig5.16
3 //page 254
4 clc;
5 close;
6 a = imread('E:\Digital_Image_Processing_Jayaraman\
  Chapter5\lena.png');
7 a = rgb2gray(a);
8 [m n] = size(a);
9 t = input('Enter the threshold parameter');
10 for i = 1:m
11     for j = 1:n
12         if(a(i,j)<t)
13             b(i,j)=0;
14         else
15             b(i,j)=255;
16         end
17     end
18 end
19 figure(1)
20 ShowImage(a, 'Original Image');
21 title('Original Image')
22 figure(2)
23 ShowImage(b, 'Thresholded Image');
24 title('Thresholded Image')
25 xlabel(sprintf('Threshold value is %g',t))
26 //Result
27 //Enter the threshold parameter 140

```

---

Scilab code Exa 5.20 Program performs gray level slicing without background



Original Image



Thresholded Image



```

1 //Caption: Program performs gray level slicing
   without background
2 //Fig.5.20
3 //page256
4 clc;
5 x = imread('E:\Digital_Image_Processing_Jayaraman\
   Chapter5\lena.png');
6 x = rgb2gray(x);
7 y = double(x);
8 [m,n]= size(y);
9 L = max(max(x));
10 a = round(L/2);
11 b = L;
12 for i =1:m
13     for j =1:n
14         if(y(i,j)>=a & y(i,j)<=b)
15             z(i,j) = L;
16         else
17             z(i,j)=0;
18         end
19     end
20 end
21 z = uint8(z);
22 figure(1)
23 ShowImage(x, 'Original Image');
24 title('Orginal Image')
25 figure(2)
26 ShowImage(z, 'Gray Level Slicing');
27 title('Gray Level Slicing without preserving
   background')

```

---



Figure 5.6: Program performs gray level slicing without background

# Chapter 6

## Image Restoration and Denoising

Scilab code Exa 6.1 Scilab code to create motion blur

```
1 //Caption:Scilab code to create motion blur
2 //Fig6.1
3 //page 326
4 clc;
5 close;
6 a = imread('E:\DIP_JAYARAMAN\Chapter6\hummm.jpg'); //
    SIVP toolbox
7 //filter coefficients of fspecial('motion',10,25)
8 H =[0,0,0,0,0,0,0,0,0.0032,0.0449,0.0865,0.0072;...
9 0,0,0,0,0,0.0092,0.0509,0.0925,0.0629,0.0213,0;...
10 0,0,0,0.0152,0.0569,0.0985,0.0569,0.0152,0,0,0;...
11 0,0.0213,0.0629,0.0925,0.0509,0.0092,0,0,0,0,0;...
12 0.0072,0.0865,0.0449,0.0032,0,0,0,0,0,0,0];
13 Motion_Blur = imfilter(a,H);
14 Motion_Blur =uint8(Motion_Blur);
15 ShowImage(a,'original Image')
16 title('original Image')
17 figure
18 ShowImage(Motion_Blur,'Motion Blurred Image')
```

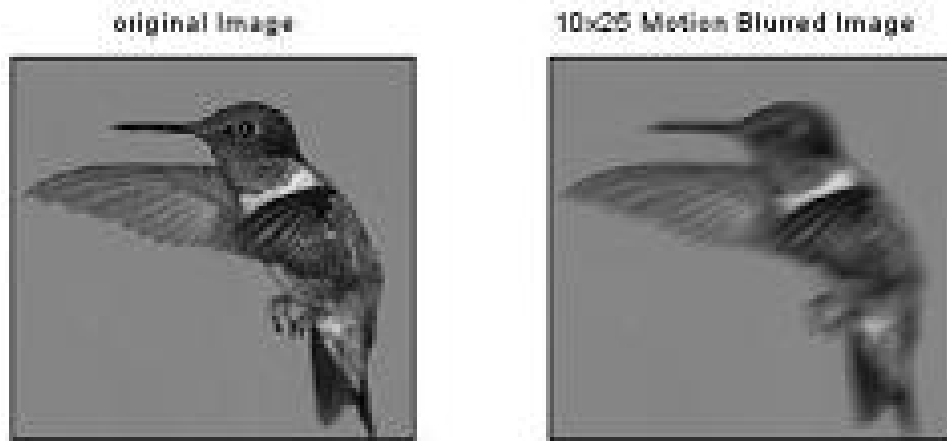


Figure 6.1: Scilab code to create motion blur

```
19 title('10x25 Motion Blurred Image')
```

---

check Appendix [AP 1](#) for dependency:

```
fft2d.sce
```

check Appendix [AP 2](#) for dependency:

```
ifft2d.sce
```

**Scilab code Exa 6.5** Scilab code performs inverse filtering

```
1 //Caption:Scilab code performs inverse filtering
2 //Degrade the image by means of a known blur
3 //Apply inverse filter to the blurred image and see
  the restored image
4 //Fig6.5
5 //page 330
6 clc;
7 close;
```

```

8 x =imread('E:\DIP_JAYARAMAN\Chapter6\flower2.jpg');
9 x=double(rgb2gray(x));
10 [M N]=size(x);
11 h = zeros(M,N);
12 for i = 1:11
13     for j = 1:11
14         h(i,j) = 1/121;
15     end
16 end
17 sigma = sqrt(4*10^(-7));
18 freqx = fft2d(x); //Fourier transform of input image
19 freqh = fft2d(h); //Fourier transform of degradation
20 y = real(ifft2d(freqh.*freqx));
21 freqy = fft2d(y);
22 powfreqx = freqx.^2/(M*N);
23 alpha = 0.5; //Indicates inverse filter
24 freqg = ((freqh.'').*abs(powfreqx)./(abs(freqh.^2)
        .*abs(powfreqx)+alpha*sigma^2));
25 Resfreqx = freqg.*freqy;
26 Resa = real(ifft2d(Resfreqx));
27 x = uint8(x);
28 y = uint8(y);
29 Resa = uint8(Resa)
30 ShowImage(x,'Original Image')
31 title('Original Image')
32 figure
33 ShowImage(y,'Degraded Image')
34 title('Degraded Image')
35 figure
36 ShowImage(Resa,'Restored Image')
37 title('Restored Image')

```

---

check Appendix [AP 1](#) for dependency:

fft2d.sce

check Appendix [AP 2](#) for dependency:

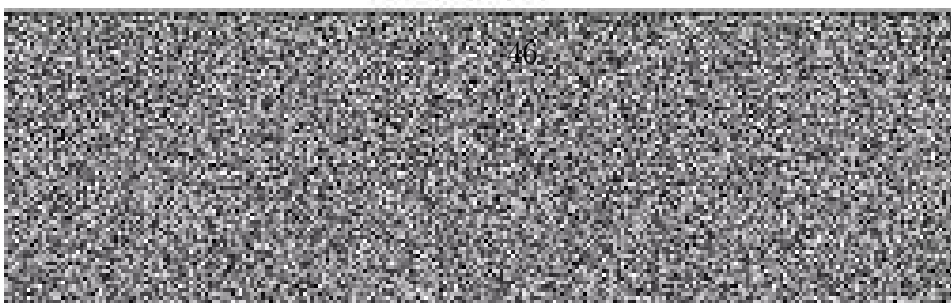
Original Image



Degraded+noise Image



Restored Image



ifft2d.sce

**Scilab code Exa 6.7** Scilab code performs inverse filtering

```
1 //Caption:Scilab code performs inverse filtering
2 //Degrade the image by means of a known blur and
  white noise
3 //The image is degraded as well as corrupted by
  noise
4 //Apply inverse filter to restore the image
5 //Fig6.7
6 //page 332
7 clc;
8 close;
9 x =imread('E:\DIP_JAYARAMAN\Chapter6\flower2.jpg');
10 x=double(rgb2gray(x));
11 [M N]=size(x);
12 h = zeros(M,N);
13 for i = 1:11
14     for j = 1:11
15         h(i,j) = 1/121;
16     end
17 end
18 sigma = sqrt(4*10(-7));
19 freqx = fft2d(x); //Fourier transform of input image
20 freqh = fft2d(h); //Fourier transform of degradation
21 y = real(ifft2d(freqh.*freqx))+10*rand(M,N, 'normal')
  ;
22 freqy = fft2d(y);
23 powfreqx = freqx.^2/(M*N);
24 alpha = 0.5; //Indicates inverse filter
25 freqg = ((freqh.'')'.*abs(powfreqx))./(abs(freqh.^2)
  .*abs(powfreqx)+alpha*sigma^2);
26 Resfreqx = freqg.*freqy;
```



Original Image



Degraded Image



Restored Image



```

27 Resa = real(iff2d(Resfreqx));
28 x = uint8(x);
29 y = uint8(y);
30 Resa = uint8(Resa)
31 ShowImage(x, 'Original Image')
32 title('Original Image')
33 figure
34 ShowImage(y, 'Degraded+noise Image')
35 title('Degraded+noise Image')
36 figure
37 ShowImage(Resa, 'Restored Image')
38 title('Restored Image')

```

---

check Appendix [AP 1](#) for dependency:

fft2d.sce

check Appendix [AP 2](#) for dependency:

iff2d.sce

**Scilab code Exa 6.9** Scilab code performs Pseudo inverse filtering

```

1 //Caption:Scilab code performs Pseudo inverse
  filtering
2 //Degrade the image by means of a known blur and
  white noise
3 //The image is degraded as well as corrupted by
  noise
4 //Apply Pseudo inverse filter to restore the image
5 //Fig6.9
6 //page 333
7 clc;
8 close;
9 x =imread('E:\DIP_JAYARAMAN\Chapter6\flower2.jpg');

```

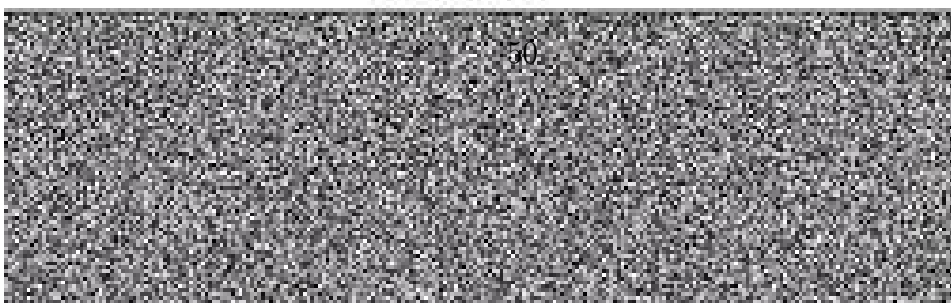
Original Image



Degraded+noise Image



Restored Image



```

10 x=double(rgb2gray(x));
11 [M N]=size(x);
12 h = zeros(M,N);
13 for i = 1:11
14     for j = 1:11
15         h(i,j) = 1/121;
16     end
17 end
18 mask_b = ones(11,11)/121;
19 [m1,n1] = size(mask_b);
20 Thr_Freq = 0.2;
21 freqx = fft2d(x); //Fourier transform of input image
22 freqh = fft2d(h); //Fourier transform of degradation
23 y = real(ifft2d(freqh.*freqx))+25*rand(M,N,'normal')
    ;
24 freqy = fft2d(y);
25 psf=zeros(M,N);
26 psf(M/2+1-(m1-1)/2:M/2+1+(m1-1)/2,N/2+1-(n1-1)/2:N
    /2+1+(n1-1)/2) = mask_b;
27 psf = fftshift(psf);
28 freq_res = fft2d(psf);
29 Inv_filt = freq_res./((abs(freq_res)).^2+Thr_Freq);
30 z = real(ifft2d(freqy.*Inv_filt));
31 x = uint8(x);
32 y = uint8(y);
33 z = uint8(z)
34 ShowImage(x,'Original Image')
35 title('Original Image')
36 figure
37 ShowImage(y,'Degraded+noise Image')
38 title('Degraded+noise Image')
39 figure
40 ShowImage(z,'Restored Image')
41 title('Restored Image')

```

---

check Appendix [AP 1](#) for dependency:

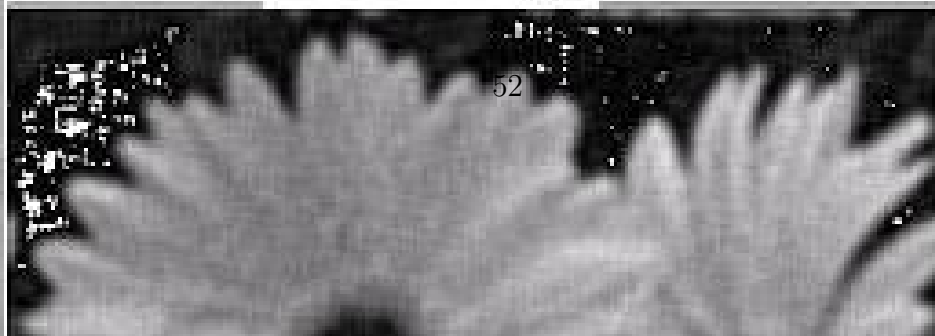
Original Image



Degraded+noise Image



Restored Image



fft2d.sce

check Appendix [AP 2](#) for dependency:

ifft2d.sce

**Scilab code Exa 6.13** Scilab code to perform wiener filtering of the corrupted image

```
1 //Caption: Scilab code to perform wiener filtering
  of the corrupted image
2 //Fig6.13
3 //Page 339
4 close;
5 clc;
6 x = imread('E:\DIP_JAYARAMAN\Chapter6\flower2.jpg');
  //SIVP toolbox
7 x=double(rgb2gray(x));
8 sigma = 50;
9 Gamma = 1;
10 alpha = 1; // It indicates Wiener filter
11 [M N]=size(x);
12 h = zeros(M,N);
13 for i = 1:5
14     for j = 1:5
15         h(i,j) = 1/25;
16     end
17 end
18 Freqa = fft2d(x);
19 Freqh = fft2d(h);
20 y = real(ifft2d(Freqh.*Freqa)) //image degradation
21 y = y+25*rand(M,N,"normal"); //Adding random noise
  with normal distribution
22 Freqy = fft2d(y);
23 Powy = abs(Freqy).^2/(M*N);
24 sFreqh = Freqh.*(abs(Freqh)>0)+1/Gamma*(abs(Freqh)
  ==0);
```

```

25 iFreqh = 1/sFreqh;
26 iFreqh = iFreqh'.*(abs(Freqh)*Gamma>1)+Gamma*abs(
    sFreqh)*iFreqh*(abs(sFreqh)*Gamma<=1);
27 iFreqh = iFreqh/(max(max(abs(iFreqh))));
28 Powy = Powy.*(Powy>sigma^2)+sigma^2*(Powy<=sigma^2);
29 Freqg = iFreqh.*(Powy-sigma^2)./(Powy-(1-alpha)*
    sigma^2);
30 ResFreqa = Freqg.*Freqy;
31 Resa = real(ifft2d(ResFreqa));
32 x = uint8(x);
33 y = uint8(y);
34 Resa = uint8(Resa);
35 ShowImage(x, 'Original Image')
36 title('Original Image')
37 figure
38 ShowImage(y, 'Degraded Image')
39 title('Degraded Image')
40 figure
41 ShowImage(Resa, 'Restored Image')
42 title('Restored Image')

```

---

**Scilab code Exa 6.18** Scilab code to Perform Average Filtering operation

```

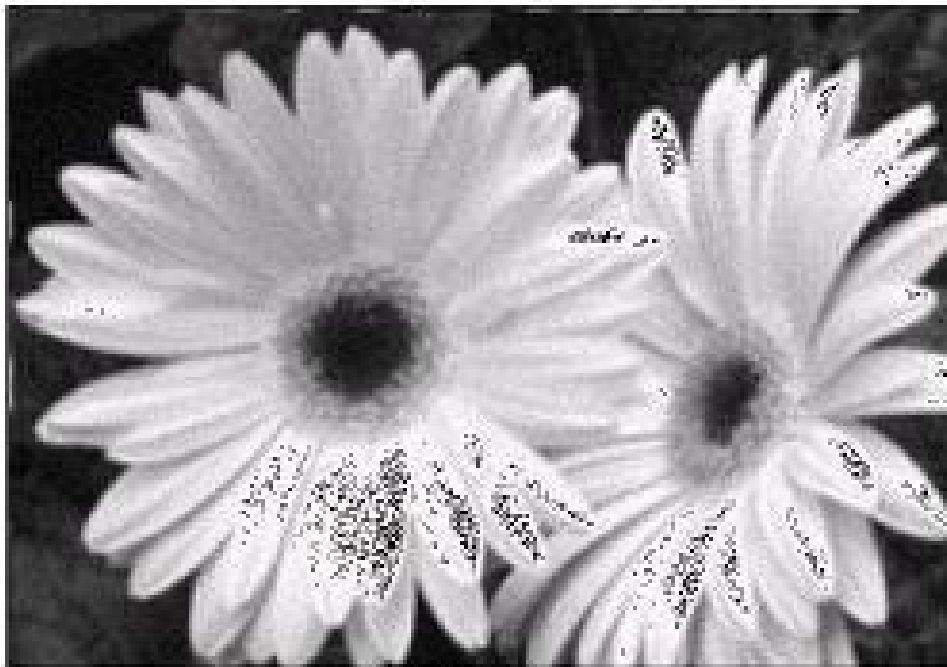
1 //Caption:Scilab code to Perform Average Filtering
    operation
2 //Fig6.18
3 //page 349
4 clc;
5 close;
6 a= imread('E:\DIP_JAYARAMAN\Chapter6\lenna.jpg');//
    SIVP toolbox
7 a=imnoise(a,'salt & pepper', 0.2); //Add salt&pepper
    noise tothe image

```

Original Image



Degraded Image



Restored Image





```

8 a=double(a);
9 [m n]=size(a);
10 N=input('enter the window size='); //The window size
    can be 3x3,5x5etc
11 Start=(N+1)/2;
12 Out_Imag=a;
13 for i=Start:(m-Start+1)
14 for j=Start:(n-Start+1)
15     limit=(N-1)/2;
16     Sum=0;
17     for k=-limit:limit,
18         for l=-limit:limit,
19             Sum=Sum+a(i+k,j+l);
20         end
21     end
22     Out_Imag(i,j)=Sum/(N*N);
23 end
24 end
25 a = uint8(a);
26 Out_Imag = uint8(Out_Imag);
27 ShowImage(a,'original Image')
28 title('Noisy Image')
29 figure
30 ShowImage(Out_Imag,'average filtered Image')
31 title('5x5 average filtered Image');

```

---

**Scilab code Exa 6.21** Scilab code to Perform median filtering

```

1 //Caption:Scilab code to Perform median filtering
2 //Fig6.21
3 //page 352
4 clc;
5 close;

```



Figure 6.7: Scilab code to Perform Average Filtering operation

```

6 c = imread('E:\DIP_JAYARAMAN\Chapter6\cameraman.jpg')
   );//SIVP toolbox
7 N = input('Enter the window size');
8 a = double(imnoise(c,'salt & pepper',0.2));
9 [m,n] = size(a);
10 b = a;
11 if(modulo(N,2)==1)
12     Start = (N+1)/2;
13     End = Start;
14     limit1 = (N-1)/2;
15     limit2 = limit1;
16 else
17     Start = N/2;
18     End = Start+1;
19     limit1 = (N/2)-1;
20     limit2 = limit1+1;
21 end
22 for i = Start:(m-End+1)
23     for j = Start:(n-End+1)
24         I = 1;
25         for k = -limit1:limit2
26             for l = -limit1:limit2
27                 mat(I) = a(i+k,j+1)
28                 I = I+1;
29             end

```

```

30         end
31         mat = gsort(mat);
32         if(modulo(N,2)==1)
33             b(i,j) = (mat(((N^2)+1)/2));
34         else
35             b(i,j) = (mat((N^2)/2)+mat(((N^2)/2)+1))/2;
36         end
37     end
38 end
39 a = uint8(a);
40 b = uint8(b);
41 figure
42 ShowImage(c, 'Original Image')
43 title('Original Image')
44 figure
45 ShowImage(a, 'noisy image')
46 title('noisy image')
47 figure
48 ShowImage(b, 'Median Filtered Image')
49 title('5x5 Median Filtered Image')

```

---

check Appendix [AP 3](#) for dependency:

Func\_medianall.sci

**Scilab code Exa 6.23** Scilab code to Perform median filtering of colour image

```

1 //Caption: Scilab code to Perform median filtering of
   colour image
2 //Fig6.23(a)
3 //page 353
4 clc;
5 close;

```

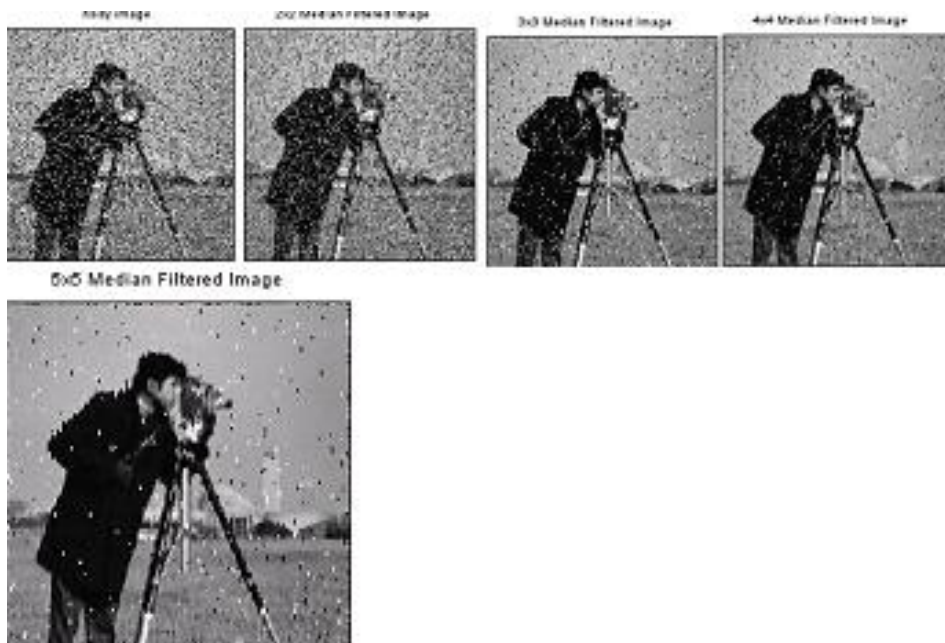


Figure 6.8: Scilab code to Perform median filtering

```

6 a=imread('E:\DIP_JAYARAMAN\Chapter6\peppers.png');
  //SIVP toolbox
7 N=input('enter the window size');
8 b=imresize(a,[256,256]);
9 b=imnoise(b,'salt & pepper',.1);
10 [m n]=size(b);
11 R=b(:,:,1);
12 G=b(:,:,2);
13 B=b(:,:,3);
14 Out_R=Func_medianall(R,N); //Applying Median filter
  to R plane
15 Out_G=Func_medianall(G,N); //Applying Median filter
  to G plane
16 Out_B=Func_medianall(B,N); //Applying Median filter
  to B plane
17 Out_Image(:,:,1)=Out_R;
18 Out_Image(:,:,2)=Out_G;
19 Out_Image(:,:,3)=Out_B;

```

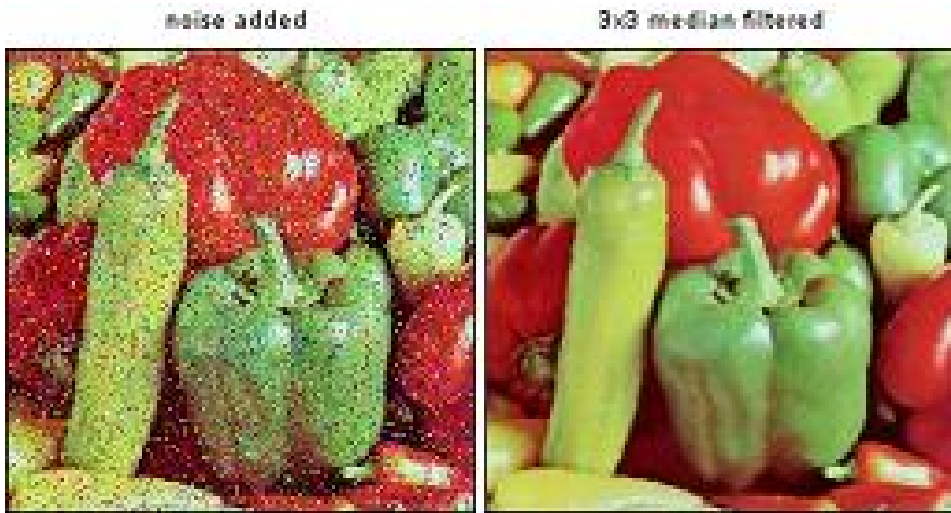


Figure 6.9: Scilab code to Perform median filtering of colour image

```

20 b = uint8(b);
21 Out_Image = uint8(Out_Image);
22 //ShowColorImage(b,'noise added')
23 //title('noise added')
24 figure
25 ShowColorImage(Out_Image,'3x3 median filtered')
26 title('3x3 median filtered')

```

---

#### Scilab code Exa 6.24 Scilab code to Perform Trimmed Average Filter

```

1 //Caption:Scilab code to Perform Trimmed Average
  Filter
2 //Alpha trimmed average filter
3 //Fig6.24
4 //page 355
5 clc;
6 close;

```

```

7 c = imread('E:\DIP_JAYARAMAN\Chapter6\lenna.jpg'); //
  SIVP toolbox
8 s = 1; //s denotes the number of values to be left
  in the end
9 r = 1;
10 N = 9; //3x3 window
11 a = double(imnoise(c, 'gaussian'));
12 [m,n] = size(a);
13 b = zeros(m,n);
14 for i= 2:m-1
15     for j = 2:n-1
16         mat = [a(i,j),a(i,j-1),a(i,j+1),a(i-1,j),a(i
                +1,j),a(i-1,j-1),...
                a(i-1,j+1),a(i-1,j+1),a(i+1,j+1)];
17         sorted_mat = gsort(mat);
18         Sum=0;
19         for k=r+s:(N-s)
20             Sum = Sum+mat(k);
21         end
22         b(i,j)= Sum/(N-r-s);
23     end
24 end
25 end
26 a = uint8(a);
27 b = uint8(b);
28 //figure
29 //imshow(c)
30 //title('Original Image')
31 figure
32 ShowImage(a, 'noisy image')
33 title('noisy image')
34 figure
35 ShowImage(b, 'Trimmed Average Filtered Image')
36 title('Trimmed Average Filtered Image')

```

---

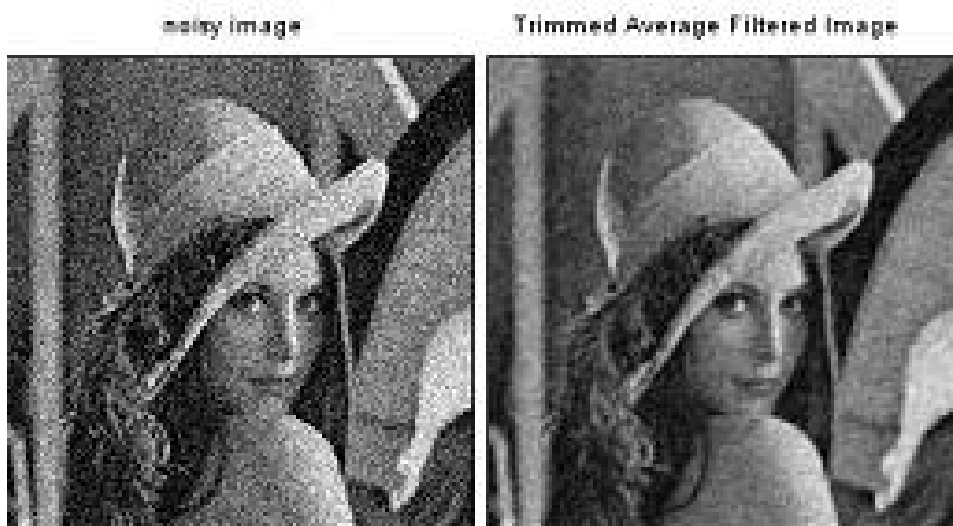


Figure 6.10: Scilab code to Perform Trimmed Average Filter

# Chapter 7

## Image Segmentation

Scilab code Exa 7.23 Scilab code for Differentiation of Gaussian function

```
1 //Caption: Scilab code for Differentiation of
   Gaussian function
2 //Fig7.23
3 //page388
4 clc;
5 close;
6 sigma=input('Enter the value of sigma:')
7 i=-10:.1:10;
8 j=-10:.1:10;
9 r=sqrt(i.*i+j.*j);
10 y=(1/(sigma^2))*(((r.*r)/sigma^2)-1).*exp(-r.*r/2*
   sigma^2);
11 plot(i,y)
12 legend(sprintf('The sigma value is %g',sigma))
13 xtitle('Differentiation of Gaussian function')
```

---



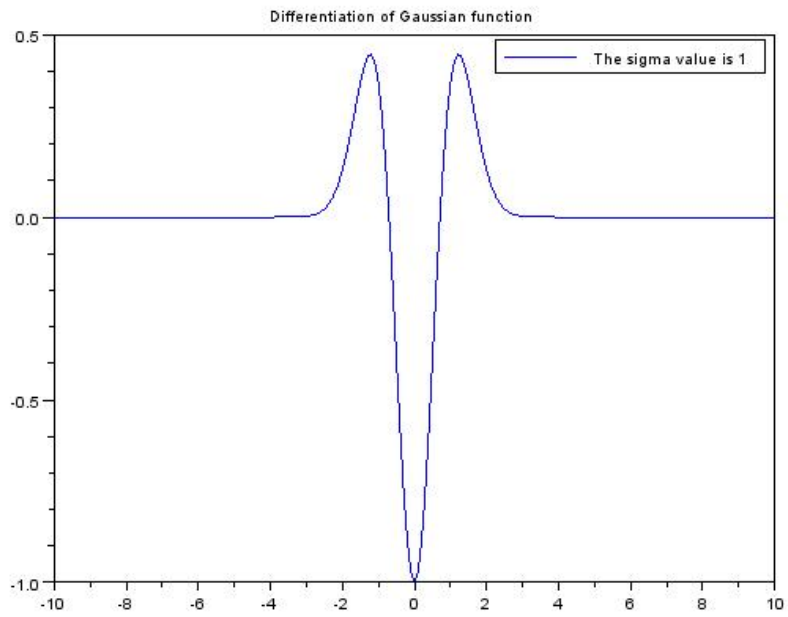


Figure 7.1: Scilab code for Differentiation of Gaussian function



Figure 7.2: Scilab code for Differentiation of Gaussian function

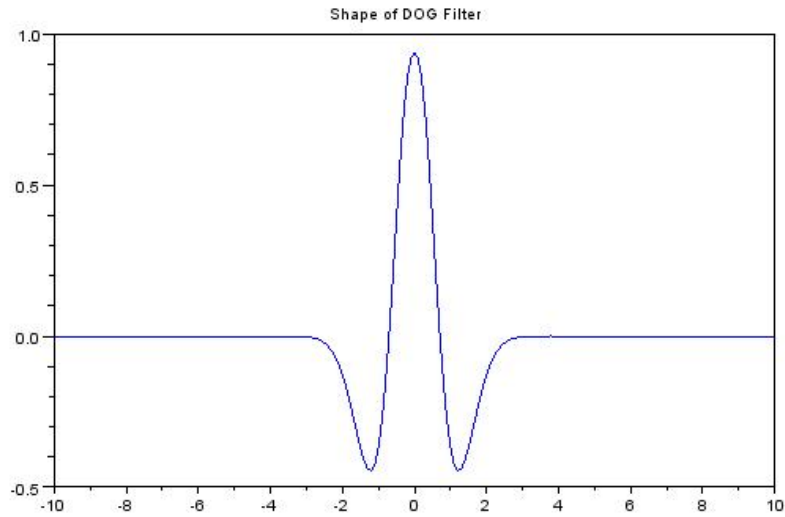


Figure 7.3: Scilab code for Differentiation of Gaussian Filter function

**Scilab code Exa 7.25** Scilab code for Differentiation of Gaussian Filter function

```

1 //Caption: Scilab code for Differentiation of
  Gaussian Filter function
2 //Fig7.25
3 //page389
4 clc;
5 close;
6 sigma1 = input('Enter the value of sigma1:')
7 sigma2 = input('Enter the value of sigma2:')
8 i=-10:.1:10;
9 j=-10:.1:10;
10 r=sqrt(i.*i+j.*j);

```



Figure 7.4: Scilab code for Differentiation of Gaussian Filter function

```
11 y1 = (1/(sigma1^2))*(((r.*r)/sigma1^2)-1).*exp(-r.*r
    /2*sigma1^2);
12 y2 = (1/(sigma2^2))*(((r.*r)/sigma2^2)-1).*exp(-r.*r
    /2*sigma2^2);
13 y = y1-y2;
14 plot(i,y)
15 xtitle('Shape of DOG Filter')
16 //Result
17 //Enter the value of sigma1: 4
18 //Enter the value of sigma2: 1
19 //
```

---

**Scilab code Exa 7.27** Scilab code for Edge Detection using Different Edge detectors

```
1 //Caption: Scilab code for Edge Detection using
   Different Edge detectors
2 //[1]. Sobel [2].Prewitt [3].Log [4].Canny
3 //Fig7.27
4 //page389
5 close;
6 clc;
7 a = imread('E:\DIP_JAYARAMAN\Chapter7\sailing.jpg');
8 a = rgb2gray(a);
9 c = edge(a, 'sobel');
10 d = edge(a, 'prewitt');
11 e = edge(a, 'log');
12 f = edge(a, 'canny');
13 ShowImage(a, 'Original Image')
14 title('Original Image')
15 figure
16 ShowImage(c, 'Sobel')
17 title('Sobel')
18 figure
19 ShowImage(d, 'Prewitt')
20 title('Prewitt')
21 figure
22 ShowImage(e, 'Log')
23 title('Log')
24 figure
25 ShowImage(f, 'Canny')
26 title('Canny')
```

---

**Scilab code Exa 7.30** Scilab code to perform watershed transform

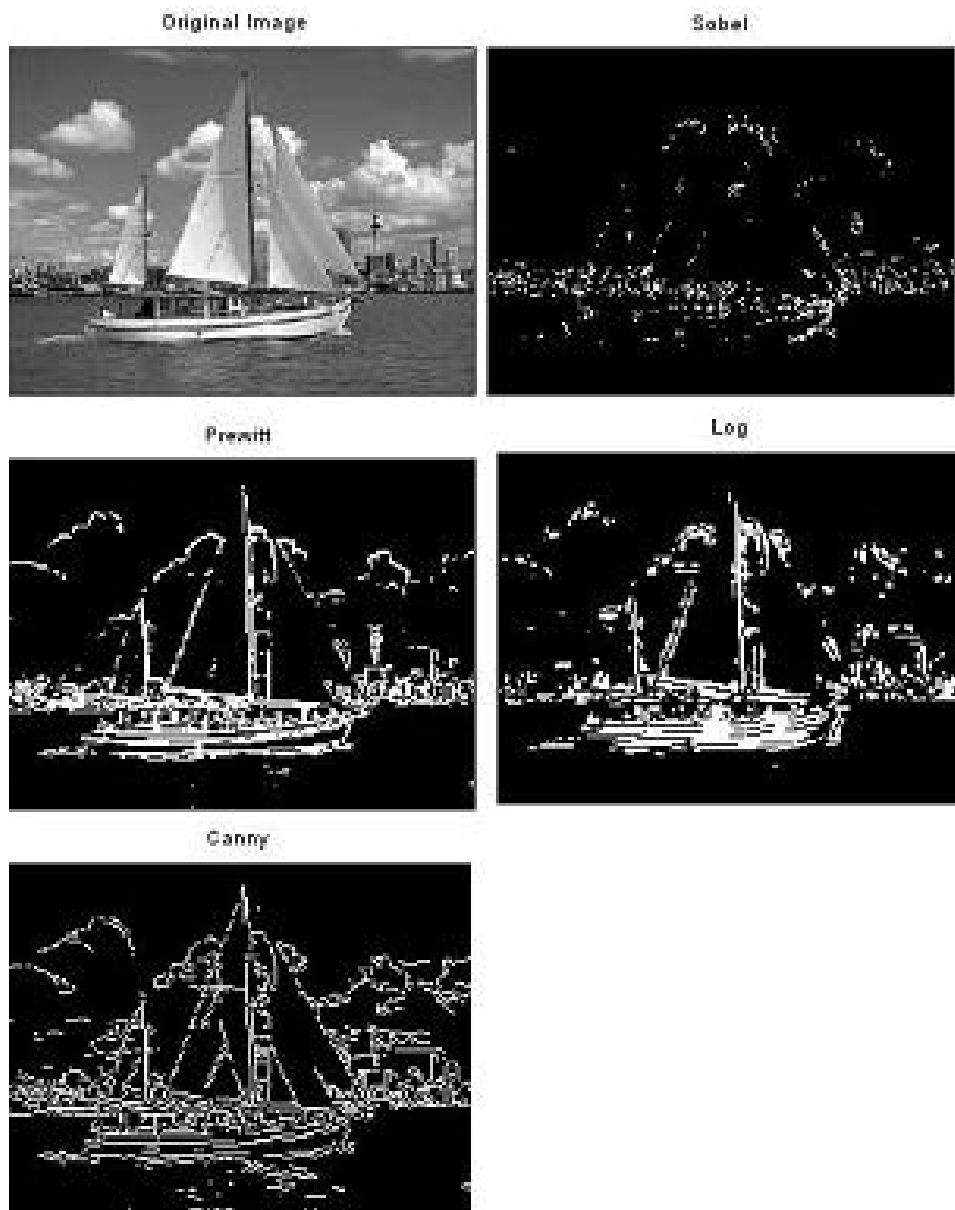


Figure 7.5: Scilab code for Edge Detection using Different Edge detectors

```

1 //Caption: Scilab code to perform watershed
  transform
2 //Fig7.30
3 //Page396
4 clc;
5 close;
6 b = imread('E:\DIP_JAYARAMAN\Chapter7\teaset.png');
7 a = rgb2gray(b);
8 global EDGE_SOBEL;
9 Gradient = EdgeFilter(a, EDGE_SOBEL);
10 Threshold1 = CalculateOtsuThreshold(Gradient); //
   determine a threshold
11 EdgeImage = ~SegmentByThreshold(Gradient,Threshold1)
   ;
12 DistanceImage = DistanceTransform(EdgeImage);
13 Threshold2 = CalculateOtsuThreshold(DistanceImage)
   // determine a threshold
14 ThresholdImage = SegmentByThreshold(DistanceImage ,
   Threshold2);
15 MarkerImage = SearchBlobs(ThresholdImage);
16 SegmentedImage = Watershed(Gradient,MarkerImage);
17 figure
18 ShowColorImage(b, 'teaset ')
19 title('teaset.png')
20 figure
21 ColorMapLength = length(unique(SegmentedImage));
22 ShowImage(SegmentedImage, 'Result of Watershed
   Transform ',jetcolormap(ColorMapLength));

```

---

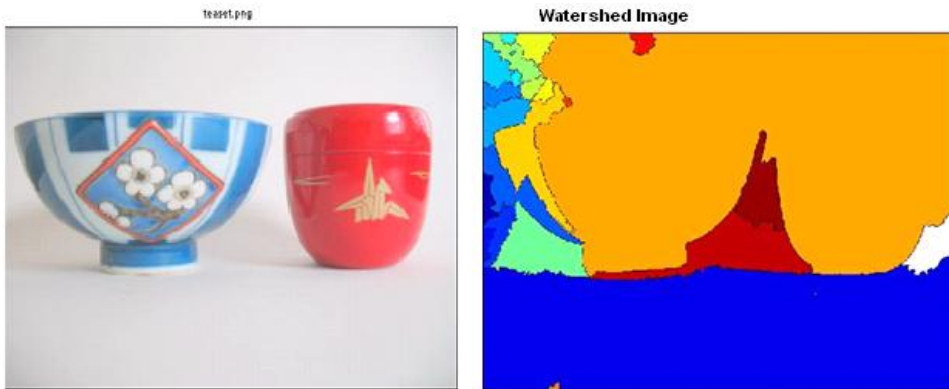


Figure 7.6: Scilab code to perform watershed transform



# Chapter 8

## Object Recognition

Scilab code Exa 8.4 To verify the given matrix is a covariance matrix

```
1 //Caption: To verify the given matrix is a
   covariance matrix
2 //Problem 4
3 //page438
4 close;
5 clear;
6 clc;
7 K = [37,-15;-15,37];
8 evals = spec(K);
9 evals = gsort(evals);
10 disp(evals, 'Eigen Values are =')
11 if (evals==abs(evals)) then
12     disp('Both the eigen values are non-negative and
           the given matrix is a covariance matrix');
13 else
14     disp('non-covariance matrix')
15 end
```

---

Scilab code Exa 8.5 To compute the covariance of the given 2D data

```

1 //Caption: To compute the covariance of the given 2D
  data
2 //Problem 5
3 //page439
4 close;
5 clear;
6 clc;
7 X1 = [2,1]';
8 X2 = [3,2]';
9 X3 = [2,3]';
10 X4 = [1,2]';
11 X = [X1,X2,X3,X4];
12 disp(X, 'X=');
13 [M,N] = size(X); //M=rows, N = columns
14 for i =1:N
15     m(i) = mean(X(:,i));
16     A(:,i) = X(:,i)-m(i);
17 end
18 m = m';
19 disp(m, 'mean =');
20 K = A'*A;
21 K = K/(M-1);
22 disp(K, 'The Covaraince matix is K =')
23 //Result
24 //X=
25 //      2.      3.      2.      1.
26 //      1.      2.      3.      2.
27 //mean =
28 //      1.5      2.5      2.5      1.5
29 //
30 //The Covaraince matix is K =
31 //      0.5      0.5     - 0.5     - 0.5
32 //      0.5      0.5     - 0.5     - 0.5
33 //     - 0.5     - 0.5      0.5      0.5
34 //     - 0.5     - 0.5      0.5      0.5

```

---

Scilab code Exa 8.9 Develop a perceptron AND function with bipolar inputs and targets

```
1 //Caption: Develop a perceptron AND function with
   bipolar inputs and targets
2 //Problem 9
3 //page441
4 close;
5 clear;
6 clc;
7 X1 = [1,-1,1,-1]; //X1 and X2 are input vectors to
   AND function
8 X2 = [1,1,-1,-1];
9 //b = [1,1,1,1]; //Biasing vector
10 T = [1,-1,-1,-1]; //Target vector for AND function
11 W1 = 0; //Weights are initialized
12 W2 = 0;
13 b = 0; //bias initialized
14 alpha = 1; //learning rate
15 for i = 1:length(X1)
16     Yin(i) = b+X1(i)*W1+X2(i)*W2;
17     if (Yin(i)>=1)
18         Y(i)=1;
19     elseif((Yin(i)<1)&(Yin(i)>=-1))
20         Y(i)=0;
21     elseif(Yin(i)<-1)
22         Y(i)=-1;
23     end
24     disp(Yin(i), 'Yin=')
25     disp(Y(i), 'Y=')
26     if(Y(i)~=T(i))
27         b = b+alpha*T(i);
28         W1 = W1+alpha*T(i)*X1(i);
29         W2 = W2+alpha*T(i)*X2(i);
30         disp(b, 'b=')
```

```
31         disp(W1, 'W1=')
32         disp(W2, 'W2=')
33     end
34 end
35 disp('Final Weights after one iteration are')
36 disp(b, 'Bias Weigth b=')
37 disp(W1, 'W1=')
38 disp(W2, 'W2=')
```

---

# Chapter 9

## Image Compression

Scilab code Exa 9.9 Program performs Block Truncation Coding BTC

```
1 //Caption: Program performs Block Truncation Coding(
    BTC)
2 //Example 9.9
3 //page512
4 close;
5 clear;
6 clc;
7 x =
    [65,75,80,70;72,75,82,68;84,72,62,65;66,68,72,80];

8 disp(x,'Original Block is x =')
9 [m1 n1]=size(x);
10 blk=input('Enter the block size:');
11 for i = 1 : blk : m1
12     for j = 1 : blk : n1
13         y = x(i:i+(blk-1),j:j+(blk-1)) ;
14         m = mean(mean(y));
15         disp(m,'mean value is m =')
16         sig=std2(y);
17         disp(sig,'Standard deviation of the block is
                =')
```

```

18         b = y > m ; //the binary block
19         disp(b,'Binary allocation matrix is B=')
20         K = sum(sum(b));
21         disp(K,'number of ones =')
22         if (K ~= blk^2 ) & ( K ~= 0)
23             ml = m-sig*sqrt(K/((blk^2)-K));
24             disp(ml,'The value of a =')
25             mu = m+sig*sqrt(((blk^2)-K)/K);
26             disp(mu,'The value of b =')
27             x(i:i+(blk-1), j:j+(blk-1)) = b*mu
                +(1- b)*ml;
28         end
29     end
30 end
31 disp(round(x),'Reconstructed Block is x =')
32 //Result
33 //Original Block is x =
34 //
35 //      65.      75.      80.      70.
36 //      72.      75.      82.      68.
37 //      84.      72.      62.      65.
38 //      66.      68.      72.      80.
39 //
40 //Enter the block size:4
41 //mean value is m = 72.25
42 //Standard deviation of the block is = 6.6282225
43 //Binary allocation matrix is B=
44 //
45 //   F T T F
46 //   F T T F
47 //   T F F F
48 //   F F F T
49 //
50 //number of ones = 6
51 //The value of a = 67.115801
52 //The value of b = 80.806998
53 //Reconstructed Block is x =
54 //

```

```

55 //      67.      81.      81.      67.
56 //      67.      81.      81.      67.
57 //      81.      67.      67.      67.
58 //      67.      67.      67.      81.

```

---

**Scilab code Exa 9.59** Program performs Block Truncation Coding

```

1 //Caption: Program performs Block Truncation Coding(
    BTC) by choosing different
2 //block sizes
3 //Fig.9.59: MATLAB Example1
4 //page514
5 close;
6 clc;
7 x =imread('E:\Digital_Image_Processing-Jayaraman\
    Chapter9\lenna.jpg'); //SIVP toolbox
8 //x=imresize(x,[256 256]);
9 x1=x;
10 x=double(x);
11 [m1 n1]=size(x);
12 blk=input('Enter the block size:');
13 for i = 1 : blk : m1
14 for j = 1 : blk : n1
15     y = x(i:i+(blk-1),j:j+(blk-1)) ;
16     m = mean(mean(y));
17     sig=std2(y);
18     b = y > m ; //the binary block
19     K = sum(sum(b));
20         if (K ~= blk^2 ) & ( K ~= 0)
21             m1 = m-sig*sqrt(K/((blk^2)-K));
22             mu = m+sig*sqrt(((blk^2)-K)/K);
23             x(i:i+(blk-1), j:j+(blk-1)) = b*mu
                +(1- b)*m1;
24         end
25 end

```



Figure 9.1: Program performs Block Truncation Coding

```

26 end
27 //imshow(uint8(x))
28 //title('Reconstructed Image')
29 x = uint8(x);
30 figure(1)
31 imshow(x1)
32 title('Original Image'); //IPD toolbox
33 figure(2)
34 ShowImage(x, 'Reconstructed Image'); //IPD toolbox
35 title('Block Size = 8')

```

---



# Chapter 10

## Binary Image Processing

Scilab code Exa 10.17 Scilab Code for dilation and erosion process

```
1 //Caption: Scilab Code for dilation and erosion
  process
2 //Fig.10.17
3 //Page553
4 close;
5 clear;
6 clc;
7 a = imread('E:\DIP_JAYARAMAN\Chapter10\morph1.bmp');
  //SIVP toolbox
8 //b =[1,1,1;1,1,1;1,1,1];
9 StructureElement = CreateStructureElement('square',
  3) ;
10 a1 = DilateImage(a,StructureElement);
11 a2 = ErodeImage(a,StructureElement);
12 //Displaying original Image
13 //imshow(a)
14 figure(1)
15 ShowImage(a,'Original Image');
16 //Displaying Dilated Image
17 //imshow(a1)
18 figure(2)
```

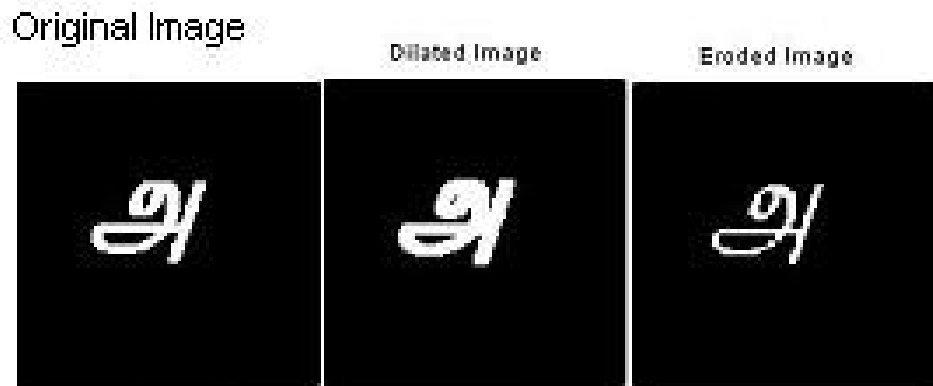


Figure 10.1: Scilab Code for dilation and erosion process

```

19 ShowImage(a1, 'Dilated Image');
20 xtitle('Dilated Image')
21 //Displaying Eroded Image
22 //imshow(a2)
23 figure(3)
24 ShowImage(a2, 'Eroded Image');
25 xtitle('Eroded Image')

```

---

**Scilab code Exa 10.19** Scilab Code to perform an opening and closing operation on t

```

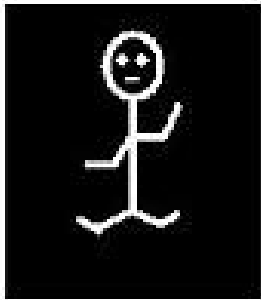
1 //Caption: Scilab Code to perform an opening and
   closing operation on the image
2 //Fig.10.19
3 //Page555
4 close;
5 clear;
6 clc;
7 a = imread('E:\DIP_JAYARAMAN\Chapter10\morph2.bmp');
   //SIVP toolbox

```

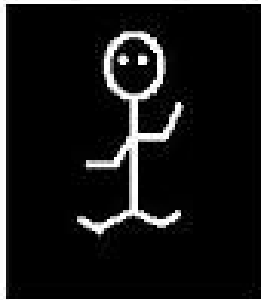
```
8 //b =[1,1,1;1,1,1;1,1,1];
9 StructureElement = CreateStructureElement('square',
    3) ;
10 //Opening is done by first applying erosion and then
    dilation operations on image
11 b1 = ErodeImage(a,StructureElement);
12 b2 = DilateImage(b1,StructureElement);
13 //Closing is done by first applying dilation and
    then erosion operation on image
14 a1 = DilateImage(a,StructureElement);
15 a2 = ErodeImage(a1,StructureElement);
16 //Displaying original Image
17 figure(1)
18 ShowImage(a,'Original Image');
19 //Displaying Opened Image
20 figure(2)
21 ShowImage(b2,'Opened Image');
22 xtitle('Opened Image')
23 //Displaying Closed Image
24 figure(3)
25 ShowImage(a2,'Closed Image');
26 xtitle('Closed Image')
```

---

Original Image



Opened image



Closed image

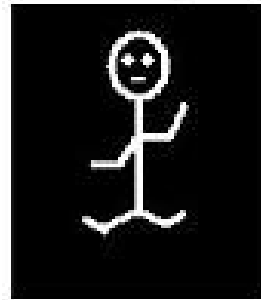


Figure 10.2: Scilab Code to perform an opening and closing operation on the image

# Chapter 11

## Colour Image Processing

Scilab code Exa 11.4 Read an RGB image and extract the three colour components red

```
1 //Caption:Read an RGB image and extract the three
   colour components: red,green
2 //and blue
3 //Fig.11.4: MATLAB Example1
4 //page588
5 clc;
6 close;
7 RGB = imread('E:\DIP_JAYARAMAN\Chapter11\peppers.png
   '); //SIVP toolbox
8 R = RGB;
9 G = RGB;
10 B = RGB;
11 R(:, :, 2)=0;
12 R(:, :, 3)=0;
13 G(:, :, 1)=0;
14 G(:, :, 3)=0;
15 B(:, :, 1)=0;
16 B(:, :, 2)=0;
17 figure(1)
18 ShowColorImage(RGB, 'Original Color Image'); //IPD
   toolbox
```

```

19 title('Original Color Image');
20 figure(2)
21 ShowColorImage(R, 'Red Component');
22 figure(3)
23 ShowColorImage(G, 'Green Component');
24 figure(4)
25 ShowColorImage(B, 'Blue Component');

```

---

Scilab code Exa 11.12 Read a Colour image and separate the colour image into red g

```

1 //Caption:Read a Colour image and separate the
   colour image into: red,green
2 //and blue planes
3 //Fig.11.12: MATLAB Example2
4 //page592
5 clc;
6 close;
7 RGB = imread('E:\DIP_JAYARAMAN\Chapter11\peppers.png
   '); //SIVP toolbox
8 a1 = RGB;
9 b1 = RGB;
10 c1 = RGB;
11 a1(:, :, 1)=0;
12 b1(:, :, 2)=0;
13 c1(:, :, 3)=0;
14 figure(1)
15 ShowColorImage(RGB, 'Original Color Image'); //IPD
   toolbox
16 figure(2)
17 ShowColorImage(a1, 'Red Missing');
18 figure(3)
19 ShowColorImage(b1, 'Green Missing');
20 figure(4)

```



Figure 11.1: Read an RGB image and extract the three colour components red green blue

```
21 ShowColorImage(c1, 'Blue Missing');
```

---

Scilab code Exa 11.16 Compute the histogram of the colour image

```
1 //Caption:Compute the histogram of the colour image
2 //Fig.11.16: MATLAB Example3
3 //page595
4 clc;
5 close;
6 I = imread('E:\DIP_JAYARAMAN\Chapter11\lavender.jpg'
7           ); //SIVP toolbox
8 figure(1)
9 ShowColorImage(I, 'Original Color Image'); //IPD
10 toolbox
11 J = im2double(I);
12 [index,map] = RGB2Ind(I); //IPD toolbox
13 pixels = prod(size(index));
14 hsv = rgb2hsv(J);
15 h = hsv(:,1);
16 s = hsv(:,2);
17 v = hsv(:,3);
18 //Finds location of black and white pixels
19 darks = find(v<0.2);
20 lights = find(s<0.05 & v>0.85);
21 h([darks lights])=-1;
22 //Gets the number of all pixels for each colour bin
23 black_pixels = length(darks)/pixels;
24 white_pixels = length(lights)/pixels;
25 red = length(find((h > .9167 | h <= .083) & h ~= -1)
26           )/pixels;
27 yellow = length(find(h > .083 & h <= .25))/pixels;
28 green = length(find(h > .25 & h <= .4167))/pixels;
29 cyan = length(find(h > .4167 & h <= .5833))/pixels;
```





Figure 11.2: Read a Colour image and separate the colour image into red green and blue planes

```

27 blue = length(find(h > .5833 & h <= .75))/pixels;
28 magenta = length(find(h > .75 & h <= .9167))/pixels;
29 //Plots histogram
30 figure(2)
31 a=gca();
32 a.data_bounds=[0,0;8,1]
33 n = 0:0.1:1;
34 plot2d2(n,red*ones(1,length(n)),5)
35 n1 = 1:0.1:2;
36 plot2d2(n1,yellow*ones(1,length(n)),7)
37 n2 = 2:0.1:3;
38 plot2d2(n2,green*ones(1,length(n)),8)
39 n3 = 3:0.1:4;
40 plot2d2(n3,cyan*ones(1,length(n)),9)
41 n4 = 4:0.1:5;
42 plot2d2(n4,blue*ones(1,length(n)),2)
43 n5 = 5:0.1:6;
44 plot2d2(n5,magenta*ones(1,length(n)),3)
45 n6 = 6:0.1:7;
46 plot2d2(n6,white_pixels*ones(1,length(n)),0)
47 n7 = 7:0.1:8
48 plot2d2(n7,black_pixels*ones(1,length(n)),5)

```

---

**Scilab code Exa 11.18** Perform histogram equalisation of the given RGB image

```

1 //Caption:Perform histogram equalisation of the
   given RGB image
2 //Fig.11.18: MATLAB Example4
3 //page596
4 clc;
5 close;
6 a = imread('E:\DIP_JAYARAMAN\Chapter11\peppers.png')
   ; //SIVP toolbox
7 //conversion of RGB to YIQ format
8 b = rgb2ntsc(a);

```

```

9 //Histogram equalisation of Y component alone
10 b(:,:,1) =
11 //conversion of YIQ to RGB format
12 c = ntsc2rgb(b);
13 figure(1)
14 ShowColorImage(a, 'Original Image'); //IPD toolbox
15 figure(2)
16 ShowColorImage(c, 'Histogram equalized Image');
    //IPD toolbox

```

---

**Scilab code Exa 11.21** This program performs median filtering of the colour image

```

1 //Caption:This program performs median filtering of
    the colour image
2 //Fig.11.21: MATLAB Example5
3 //page598
4 clc;
5 close;
6 a = imread('E:\DIP_JAYARAMAN\Chapter11\peppers.png')
    ; //SIVP toolbox
7 b = imnoise(a, 'salt & pepper', 0.2);
8 c(:,:,1)= MedianFilter(b(:,:,1), [3 3]);
9 c(:,:,2)= MedianFilter(b(:,:,2), [3 3]);
10 c(:,:,3)= MedianFilter(b(:,:,3), [3 3]);
11 figure(1)
12 ShowColorImage(a, 'Original Image'); //IPD toolbox
13 figure(2)
14 ShowColorImage(b, 'corrupted Image'); //IPD
    toolbox
15 figure(3)
16 ShowColorImage(c, 'Median Filtered Image'); //IPD
    toolbox

```

---

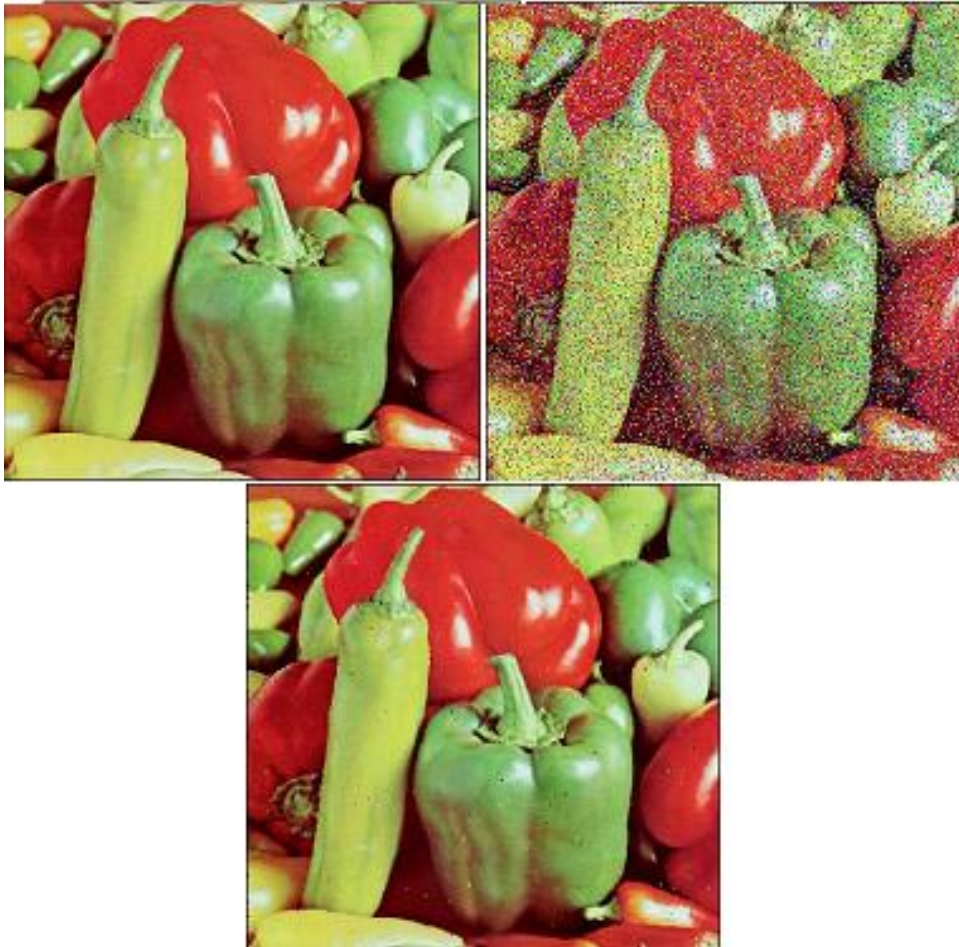


Figure 11.3: This program performs median filtering of the colour image

Scilab code Exa 11.24 Filtering only the luminance component

```
1 //Caption: Filtering only the luminance component
2 //Fig.11.24: MATLAB Example6
3 //page599
4 clc;
5 close;
6 a = imread('E:\DIP_JAYARAMAN\Chapter11\peppers.png');
    ; //SIVP toolbox
7 //conversion of RGB to YIQ format
8 yiq = rgb2ntsc(a);
9 //Extract the Y component alone
10 b = yiq(:,:,1);
11 h = [-1,-1,-1;-1,8,-1;-1,-1,-1];
12 //Perform high pass filtering only on Y component
13 c1 = conv2d2(b,h);
14 [m,n]= size(b);
15 for i =1:m
16     for j=1:n
17         D(i,j)= c1(i,j);
18     end
19 end
20 yiq(:,:,1)=D;
21 //convert YIQ to RGB format
22 a1 = ntsc2rgb(yiq);
23 figure(1)
24 ShowColorImage(a, 'Original Image'); //IPD toolbox
25 figure(2)
26 ShowColorImage(a1, 'High Pass filtered Image'); //
    IPD toolbox
```

---

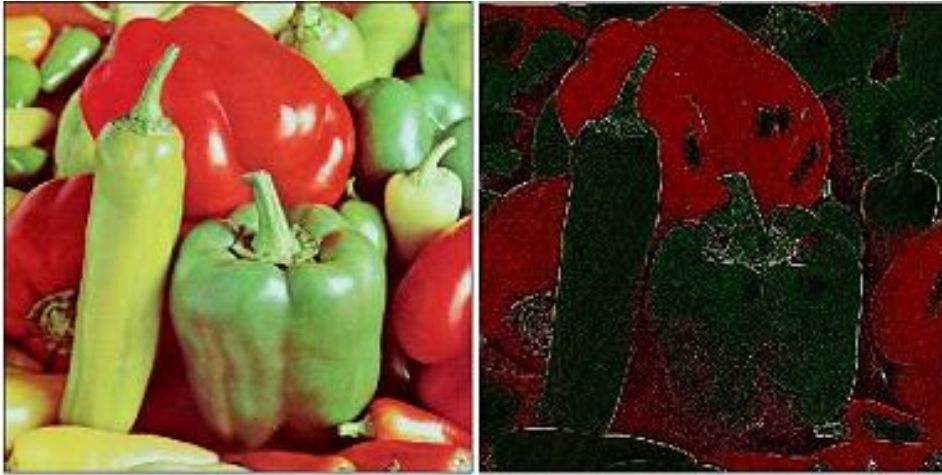


Figure 11.4: Filtering only the luminance component

**Scilab code Exa 11.28** Perform gamma correction for the given colour image

```

1 //Caption:Perform gamma correction for the given
  colour image
2 //Fig.11.28: MATLAB Example7
3 //page603
4 close;
5 clear;
6 clc;
7 I = imread('E:\DIP_JAYARAMAN\Chapter11\ararauna.png'
  ); //SIVP toolbox
8 gamma_Value = 0.5;
9 max_intensity = 255; //for uint8 image
10 //Look up table creation
11 LUT = max_intensity.*(([0:max_intensity]./
  max_intensity).^gamma_Value);
12 LUT = floor(LUT);
13 //Mapping of input pixels into lookup table values
14 K = double(I)+1;

```

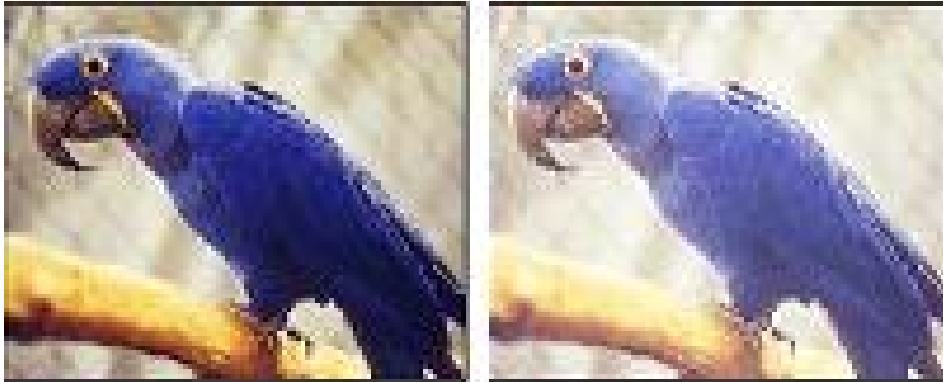


Figure 11.5: Perform gamma correction for the given colour image

```

15 J = zeros(I);
16 [m,n,p]= size(K);
17 for i = 1:m
18     for j =1:n
19         for k = 1:p
20             J(i,j,k)= LUT(K(i,j,k));
21         end
22     end
23 end
24 figure(1)
25 ShowColorImage(I, 'Original Image'); //IPD toolbox
26 figure(2)
27 ShowColorImage(uint8(J), 'Gamma Corrected Image');
    //IPD toolbox

```

---

### Scilab code Exa 11.30 Perform Pseudo Colouring Operation

```

1 //Caption:Perform Pseudo–Colouring Operation
2 //Fig.11.30
3 //page604

```

```

4  close;
5  clear;
6  clc;
7  K = imread('E:\DIP_JAYARAMAN\Chapter11\lenna.jpg');
    //SIVP toolbox
8  [m,n]= size(K);
9  I = uint8(K);
10 for i = 1:m
11     for j =1:n
12         if (I(i,j)>=0 & I(i,j)<50)
13             J(i,j,1)=I(i,j)+50;
14             J(i,j,2)=I(i,j)+100;
15             J(i,j,3)=I(i,j)+10;
16         elseif (I(i,j)>=50 & I(i,j)<100)
17             J(i,j,1)=I(i,j)+35;
18             J(i,j,2)=I(i,j)+128;
19             J(i,j,3)=I(i,j)+10;
20         elseif (I(i,j)>=100 & I(i,j)<150)
21             J(i,j,1)=I(i,j)+152;
22             J(i,j,2)=I(i,j)+130;
23             J(i,j,3)=I(i,j)+15;
24         elseif (I(i,j)>=150 & I(i,j)<200)
25             J(i,j,1)=I(i,j)+50;
26             J(i,j,2)=I(i,j)+140;
27             J(i,j,3)=I(i,j)+25;
28         elseif (I(i,j)>=200 & I(i,j)<=256)
29             J(i,j,1)=I(i,j)+120;
30             J(i,j,2)=I(i,j)+160;
31             J(i,j,3)=I(i,j)+45;
32         end
33     end
34 end
35 figure(1)
36 ShowImage(K, 'Original Image'); //IPD toolbox
37 figure(2)
38 ShowColorImage(J, 'Pseudo Coloured Image'); //IPD
    toolbox

```

---



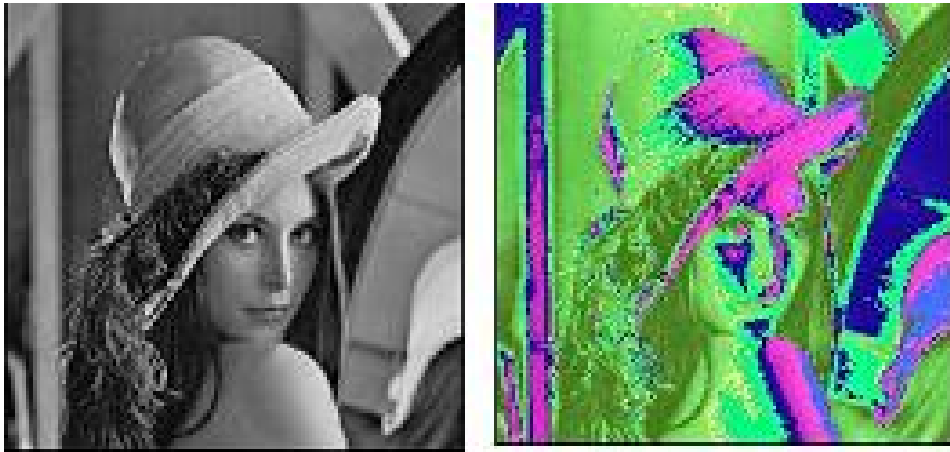


Figure 11.6: Perform Pseudo Colouring Operation

**Scilab code Exa 11.32** Read an RGB image and segment it using the threshold method

```

1 //Caption:Read an RGB image and segment it using the
  threshold method
2 //Fig11.32
3 //Page605
4 close;
5 clc;
6 I = imread('E:\DIP_JAYARAMAN\Chapter11\ararauna.png'
  ); //SIVP toolbox
7 //Conversion of RGB to YCbCr
8 b = rgb2ycbcr_1(I); //SIVP toolbox
9 [m,n,p]=size(b);
10 b = uint8(b);
11 //Threshold is applied only to Cb component
12 mask = b(:,:,2)>120;
13 figure(1)

```

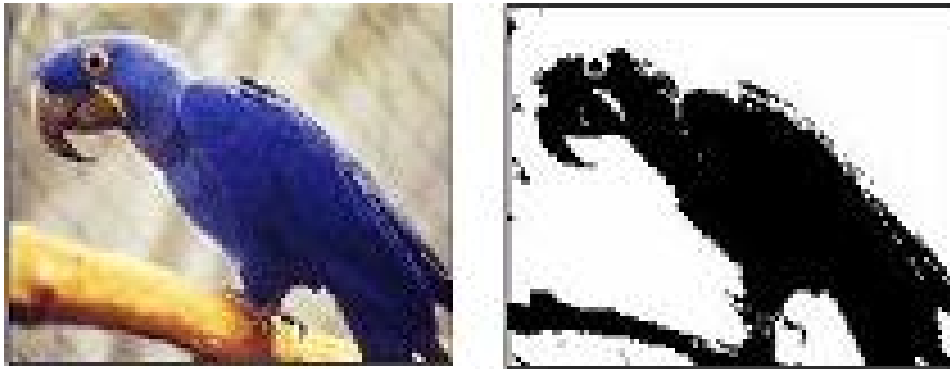


Figure 11.7: Read an RGB image and segment it using the threshold method

```
14 ShowColorImage(I, 'Original Image'); //IPD toolbox  
15 figure(2)  
16 ShowImage(mask, 'Segmented Image'); //IPD toolbox
```

---

# Chapter 12

## Wavelet based Image Processing

Scilab code Exa 12.9 Scilab code to perform wavelet decomposition

```
1 //Caption: Scilab code to perform wavelet
   decomposition
2 //Fig12.10
3 //Page624
4 clc;
5 close;
6 x = ReadImage('E:\DIP_JAYARAMAN\Chapter12\lenna.jpg'
   );
7 //The image in unsigned integer or double has to be
   converted into normalized
8 //double format
9 x = im2double(x);
10 //First Level decomposition
11 [CA,CH,CV,CD]=dwt2(x, 'db1');
12 //Second level decomposition
13 [CA1,CH1,CV1,CD1]=dwt2(CA, 'db1');
14 CA = im2int8(CA);
15 CH = im2int8(CH);
16 CV = im2int8(CV);
```

```

17 CD = im2int8(CD);
18 CA1 = im2int8(CA1);
19 CH1 = im2int8(CH1);
20 CV1 = im2int8(CV1);
21 CD1 = im2int8(CD1);
22 A = [CA,CH;CV,CD];
23 B = [CA1,CH1;CV1,CD1];
24 imshow(B)
25 title('Result of Second Level Decomposition')

```

---

**Scilab code Exa 12.42** Scilab code to generate different levels of a Gaussian pyramid

```

1 //Caption: Scilab code to generate different levels
  of a Gaussian pyramid
2 //Fig12.42
3 //Page651
4 clc;
5 close;
6 a = imread('E:\DIP_JAYARAMAN\Chapter12\apple3.bmp');
7 a = rgb2gray(a);
8 b = a;
9 kernelsize = input('Enter the size of the kernel:');
10 sd = input('Enter the standard deviation of the
  Gaussian window:');
11 rf = input('Enter the Reduction Factor:');
12 //Routine to generate Gaussian kernel
13 k = zeros(kernelsize, kernelsize);
14 [m n] = size(b);
15 t = 0;
16 for i = 1:kernelsize
17     for j=1:kernelsize
18         k(i,j) = exp(-((i-kernelsize/2).^2+(j-
  kernelsize/2).^2)/(2*sd.^2))/(2*%pi*sd
  .^2);
19         t = t+k(i,j);

```

```

20     end
21 end
22 for i = 1:kernelsize
23     for j = 1:kernelsize
24         k(i,j) = k(i,j)/t;
25     end
26 end
27 for t = 1:1:rf
28     //convolve it with the picture
29     FilteredImg = b;
30     if t==1
31         FilteredImg = filter2(k,b)/255;
32     else
33         FilteredImg = filter2(k,b);
34     end;
35     //compute the size of the reduced image
36     m = m/2;
37     n = n/2;
38     //create the reduced image through sampling
39     b = zeros(m,n);
40     for i = 1:m
41         for j = 1:n
42             b(i,j) = FilteredImg(i*2,j*2);
43         end;
44     end;
45 end;
46 figure
47 ShowImage(a, 'Original Image')
48 figure
49 ShowImage(b, 'Different Levels of Gausain Pyramid')
50 title('Different Levels of Gausain Pyramid Level 2')

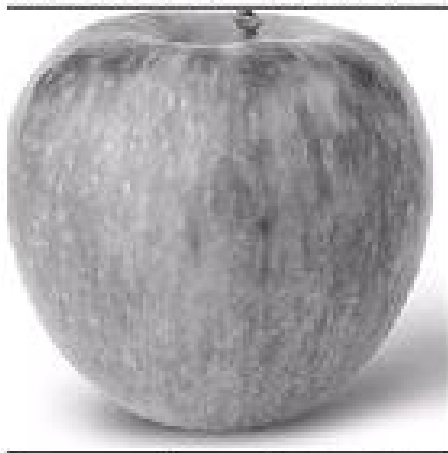
```

---

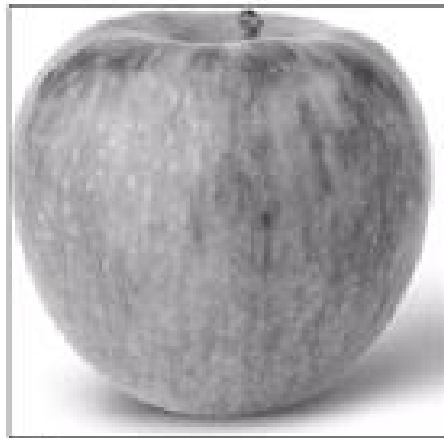
Scilab code Exa 12.57 Scilab code to implement watermarking in spatial domain



Different Levels of Gaussian Pyramid Level 0



Different Levels of Gaussian Pyramid Level 1



Different Levels of Gaussian Pyramid Level 2

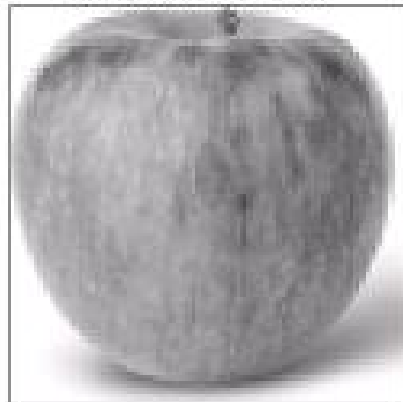


Figure 12.1: Scilab code to generate different levels of a Gaussian pyramid

```

1 //Caption: Scilab code to implement watermarking in
    spatial domain
2 //Fig12.57
3 //Page662
4 clc
5 close
6 a = imread('E:\DIP_JAYARAMAN\Chapter12\cameraman.jpg
    ');
7 figure
8 imshow(a)
9 title('Base Image');
10 b = imread('E:\DIP_JAYARAMAN\Chapter12\keyimage.jpg'
    ');
11 b = rgb2gray(b);
12 b = imresize(b,[32 32],'bicubic');
13 [m1 n1]=size(b);
14 figure
15 imshow(b)
16 title('Mark Image');
17 [m n]=size(a);
18 i1 = 1;
19 j1 = 1;
20 p = 1;
21 c = a;
22 iii = 1;
23 jjj = 1;
24 a = uint8(a);
25 b = uint8(b);
26 for ff = 1:8
27     for i = 1:32
28         jjj = 1;
29         for j = j1:j1+n1-1
30             a(i,j) = bitand(a(i,j),uint8(254)); //
                LSB of base image is set to zero.
31             temp = bitand(b(i,jjj),uint8((2^ff)-1));
                //MSB of the mark is extracted.
32             temp = temp/((2^ff)-1);
33             c(i,j) = bitor(a(i,j),uint8(temp)); //MSB

```



Figure 12.2: Scilab code to implement watermarking in spatial domain

```

                                of mark is inerted into the %LSB of
                                the base
34         jjj = jjj+1;
35         end
36     end
37     j1 = j1+32;
38 end
39 imshow(c)
40 title('Marked Image');
41 imwrite(c, 'E:\DIP_JAYARAMAN\Chapter12\marking.jpg');

```

---

**Scilab code Exa 12.63** Scilab code to implement wavelet based watermarking

```

1 //Caption: Scilab code to implement wavelet-based
  watermarking
2 //Fig12.63
3 //Page666
4 clc;
5 close;

```



```

6 //Original Image
7 img = imread('E:\DIP_JAYARAMAN\Chapter12\cameraman.
    jpg');
8 figure
9 imshow(img)
10 title('Original Image');
11 [p q] = size(img);
12 //Generate the key
13 //key = imread('E:\DIP_JAYARAMAN\Chapter12\keyimg1.
    png');
14 //key = imresize(key,[p q]);
15 key = imread('E:\DIP_JAYARAMAN\Chapter12\keyimage.
    jpg');
16 key = rgb2gray(key);
17 c = 0.001; //Initialise the weight of Watermarking
18 figure
19 imshow(key)
20 title('Key');
21 //Wavelet transform of original image (base image)
22 img = double(img);
23 key = double(key);
24 [ca,ch,cv,cd] = dwt2(img,'db1');//Compute 2D wavelet
    transform
25 //Perform the watermarking
26 y = [ca ch;cv cd];
27 Y = y + c*key;
28 p=p/2;
29 q=q/2;
30 for i=1:p
31     for j=1:q
32         nca(i,j) = Y(i,j);
33         ncv(i,j) = Y(i+p,j);
34         nch(i,j) = Y(i,j+q);
35         ncd(i,j) = Y(i+p,j+q);
36     end
37 end
38 //Display the Watermarked image
39 wimg = idwt2(nca,nch,ncv,ncd,'db1');

```

```
40 wimg1 = uint8(wimg);
41 figure
42 imshow(wimg1)
43 title('Watermarked Image')
44 //Extraction of key from Watermarked image
45 [rca,rch,rcv,rcd] = dwt2(wimg,'db1'); //Compute 2D
    wavelet transform
46 n1=[rca,rch;rcv,rcd];
47 N1=n1-y;
48 N1 = N1*4;
49 N1 = im2int8(N1);
50 figure
51 imshow(N1)
52 title('Extract the key from watermarked image')
```

---

# Appendix

Scilab code AP 1 2D Fast Fourier Transform

```
1 function [a2] = fft2d(a)
2 //a = any real or complex 2D matrix
3 //a2 = 2D-DFT of 2D matrix 'a'
4 m=size(a,1)
5 n=size(a,2)
6 // fourier transform along the rows
7 for i=1:n
8 a1(:,i)=exp(-2*i*pi*(0:m-1)'.*(0:m-1)/m)*a(:,i)
9 end
10 // fourier transform along the columns
11 for j=1:m
12 a2temp=exp(-2*i*pi*(0:n-1)'.*(0:n-1)/n)*(a1(j,:))
13 a2(j,:)=a2temp.'
14 end
15 for i = 1:m
16     for j = 1:n
17         if((abs(real(a2(i,j))))<0.0001)&(abs(imag(a2(
18             i,j))))<0.0001))
19             a2(i,j)=0;
20         elseif(abs(real(a2(i,j))))<0.0001)
21             a2(i,j)= 0+%i*imag(a2(i,j));
22         elseif(abs(imag(a2(i,j))))<0.0001)
23             a2(i,j)= real(a2(i,j))+0;
24     end
25 end
```

25 end

---

### Scilab code AP 2 2D Inverse FFT

```
1 function [a] =ifft2d(a2)
2 //a2 = 2D-DFT of any real or complex 2D matrix
3 //a = 2D-IDFT of a2
4 m=size(a2,1)
5 n=size(a2,2)
6 //Inverse Fourier transform along the rows
7 for i=1:n
8 a1(:,i)=exp(2*i*pi*(0:m-1)'.*(0:m-1)/m)*a2(:,i)
9 end
10 //Inverse fourier transform along the columns
11 for j=1:m
12 atemp=exp(2*i*pi*(0:n-1)'.*(0:n-1)/n)*(a1(j,:)).'
13 a(j,:)=atemp.'
14 end
15 a = a/(m*n)
16 a = real(a)
17 endfunction
```

---

### Scilab code AP 3 Median Filtering function

```
1 //The input to the function are the corrupted image
   a and the dimension
2 function [Out_Imag] = Func_medianall(a,N)
3 a=double(a);
4 [m n]=size(a);
5 Out_Imag=a;
6 if(modulo(N,2)==1)
7 Start=(N+1)/2;
8 End=Start;
9 else
10 Start=N/2;
11 End=Start+1;
12 end
13 if(modulo(N,2)==1)
```

```

14     limit1=(N-1)/2;
15     limit2=limit1;
16 else
17     limit1=(N/2)-1;
18     limit2=limit1+1;
19 end
20 for i=Start:(m-End+1),
21     for j=Start:(n-End+1),
22         I=1;
23         for k=-limit1:limit2,
24             for l=-limit1:limit2,
25                 mat(I)=a(i+k,j+1);
26                 I=I+1;
27             end
28         end
29         mat=gsort(mat);           //Sort the elements to
                                   find the median
30         if(modulo(N,2)==1)
31             Out_Imag(i,j)=(mat(((N^2)+1)/2));
32         else
33             Out_Imag(i,j)=(mat((N^2)/2)+mat(((N^2)/2)
                                   +1))/2;
34         end
35     end
36 end

```

---

**Scilab code AP 4** To caculate gray level

```

1 function [g] = gray(m)
2     g = (0:m-1)'/max(m-1,1)
3     g = [g g g]
4 endfunction

```

---

**Scilab code AP 5** To change the gray level of gray image

```

1 function [bout] = grayslice(I,z)
2

```

```

3 // Output variables initialisation (not found in
  input variables)
4 bout=[];
5
6 // Number of arguments in function call
7 [%nargout,%nargin] = argn(0)
8
9 if %nargin==1 then
10  z = 10;
11 elseif ~type(z)==1 then
12  z = double(z);
13 end;
14 n = z;
15 if typeof(I)=='uint8' then
16  z = (255*(0:(n-1)))/n;
17 elseif isa(I, 'uint16') | isa(I, 'int16') then
18  z = 65535*(0:(n-1))/n;
19 else // I is double or single
20  z = (0:(n-1))/n
21 end;
22 [m,n] = size(I);
23 b = zeros(m,n);
24 // Loop over all intervals , except the last
25 for i = 1:length(z)-1
26  // j is the index value we will output , so it
    depend upon storage class
27    if typeof(b)=='uint8'
28      j = i-1;
29    else
30      j = i;
31    end
32    d = find(I>=z(i) & I<z(i+1));
33    if ~isempty(d),
34      b(d) = j;
35    end
36 end
37
38 // Take care of that last interval

```

```
39 d = find(I >= z($));
40 if ~isempty(d) then
41     // j is the index value we will output, so it
        depend upon storage class
42     if typeof(b)=="uint8" then
43         j = length(z)-1;
44     else
45         j = length(z);
46     end;
47     b(d) = j;
48 end;
49 bout = b;
50 bout = double(bout);
51 endfunction
```

---