Scilab Manual for IMAGE AND VIDEO PROCESSING by Dr Sujata Kulkarni Electronics and Telecommunication Engineering Sardar Patel Institute Of Technology[1]

Solutions provided by
Dr Kulk3699
Electronics and Telecommunication Engineering
Sardar Patel Institute Of Technology

May 10, 2025

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

Contents

List of Experiments

List of Figures

Experiment: 1

To study and implement basic operations on image, different types of conversions.

Scilab code Solution 1.1 To study and implement basic operations on image and different types of conversions

```
1 //Program Title: To study and implement basic
    operations on image , different types of
    conversions.
2 //Program Description: This scilab code is used to
    perform basic operations like Quantisation, Down-
    sampling, Thresholding, Conversion to grayscale,
    HSV, YCbCr, negation, complement, etc.
3
4 //Note: Details of scilab software version and OS
    version used:
5 //Tested on OS: Windows 7 SP1, 64 bit
6 //Scilab version: 6.0.1 (Tested on 64 bit version)
7 //Toolbox used: Image Processing and Computer Vision
     Toolbox (version 2.0)
8 //Reference book name : Digital Image Processing
    book (author : Rafael C. Gonzalez and Richard E.
```

```
    Woods)
9
10 clear;
11 clc;
12 clear all;
13 close;
14
15 img = imread('lena.jpg'); // Reading Image
16 figure(); xname("Original image");
17 imshow(img);
18
19 // Convert Image to Grayscale
20 img_gray = rgb2gray(img);
21 figure(); xname("Gray image");
22 imshow(img_gray);
23
24 // Image quantization
25 // For image quantization we first perform the
      integer division followed by integral
      multiplication
26 img_128 = img/128;
27 img_128_2 = img_128 * 128;
28 figure(); xname("Half quantized image"); imshow(
      img_128_2);
29
30
31 img_64 = img/64;
32 img_64_2 = img_64 * 64;
33 figure(); xname("Quarter quantized image"); imshow(
      img_64_2);
34
35 // Image sampling
36 // In this we pixelate the image resizing it to
      small size then again to original size,
      downsampling by a factor of 8
37 [m,n]=size(img);
38 // Code for combining 64 pixels into one,
      downsampling by a factor of 8
```

```
39 image_64_combine = imresize(img,1/8);
40 image_64_combine = imresize(image_64_combine, 8);
41 figure(); xname("Sampling image combining 64 pixels"
     ); imshow(image_64_combine);
42
43 //Image thresholding
44 // For binary thresholding we quantize the gray
     image in two levels
45 // this is done by performing integral division on
     gray image by 128 followed by integral
     multiplication
46 img_binthresh = img_gray / 128;
47 img_binthresh = img_binthresh * 255;
48 figure(); xname("Binary thresholded image"); imshow(
     img_binthresh);
49
50 //Image Interpolation
51 [rows, columns] = size(img);
52 scale = [int(2*rows),int(2*columns)];
53 img_resize_NEAREST = imresize(img, scale, 'nearest')
     ; figure(); xname("Interpolation using nearest");
      imshow(img_resize_NEAREST);
54
55 img_resize_LINEAR = imresize(img, scale, 'bilinear')
     ; figure(); xname("Interpolation using linear");
     imshow(img_resize_LINEAR);
56
57 img_resize_BICUBIC = imresize(img, scale, 'bicubic')
     ; figure(); xname("Interpolation using bicubic");
      imshow(img_resize_BICUBIC);
58
59 //Conversion from RGB to Grayscale
60 bw=rgb2gray(img);
61 figure(); xname("Grayscale image");
62 imshow(bw);
63
64 // Converting rgb to hsv
65 img_hsv = rgb2hsv(img);
```

```
66 figure(); xname("HSV format");
67 imshow(img_hsv);
68
69 // Converting rgb to YCBCR
70 img_ycbcr = rgb2ycbcr(img);
71 figure(); xname("YCBCR format");
72 imshow(img_ycbcr);
73
74 // Image negation
75 img_negetion = 255 - img_gray;
76 figure(); xname("Negation");imshow(img_negetion);
77
78 img_complement = imcomplement(img);
79 figure(); xname("imcomplement Negation");imshow(
      img_complement);
80
81 //Data-type conversion
82 img_int8 = im2int8(img);
83 img_int16 = im2int16(img);
84 img_int32 = im2int32(img);
85 img_uint8 = im2uint8(img);
86 img_uint16 = im2uint16(img);
87 img_double = im2double(img);
88
89 //Subplot
90 scf(20);
91 figure(); xname("ALL images");
92 var_rows = 3;
93 var_cols = 3;
94 subplot(var_rows,var_cols,1), imshow(img);
95 title('Original image');
96 subplot(var_rows,var_cols,2), imshow(img_gray);
97 title('Grayscale Image');
98 subplot(var_rows,var_cols,3), imshow(img_complement)
      ;
99 title('Complement Image');
100 subplot(var_rows,var_cols,4), imshow(img_binthresh);
101 title('Binary Image');
```

```
102 subplot(var_rows,var_cols,5), imshow(
      image_64_combine);
103 title('Downsampled image');
104 subplot(var_rows,var_cols,6), imshow(img_negetion);
105 title('Negated image');
106 subplot(var_rows,var_cols,7), imshow(img_hsv);
107 title('HSV Image');
108 subplot(var_rows,var_cols,8), imshow(img_ycbcr);
109 title('YcbCr Image');
110 subplot(var_rows,var_cols,9), imshow(img_64_2);
111 title('Quantized Image');
```

Figure 1.1: To study and implement basic operations on image and different types of conversions

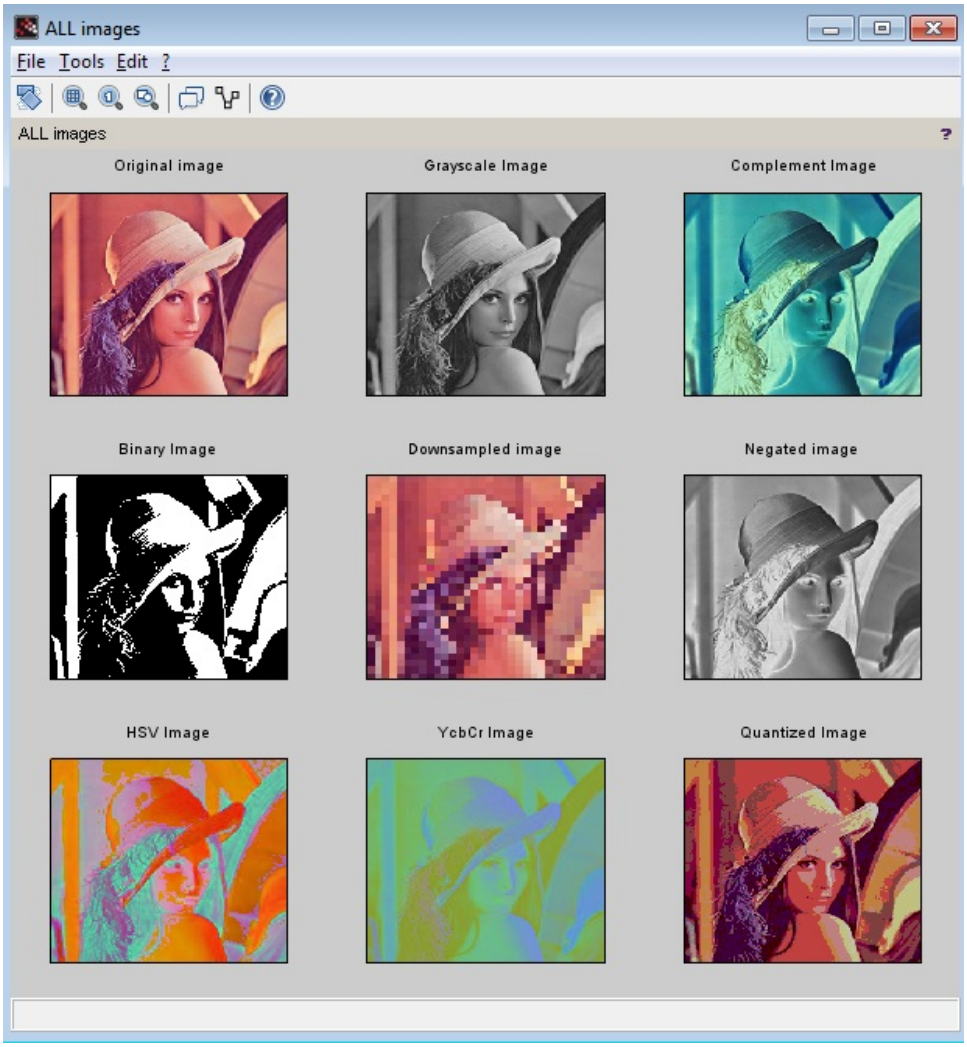

Figure 1.2: To study and implement basic operations on image and different types of conversions

Experiment: 2

To implement different transforms on given image.

Scilab code Solution 2.2 To implement different transforms on given image

```
1 //Program Title: To implement different transforms
    on given image.
2 //Program Description: This scilab code is used to
    implement DCT and DFT transforms on an image and
    also perform reconstruction of original image
    using inverse DCT and inverse DFT.
3
4 //Note: Details of scilab software version and OS
    version used:
5 //Tested on OS: Windows 7 SP1, 64 bit
6 //Scilab version: 6.0.1 (Tested on 64 bit version)
7 //Toolbox used: Image Processing and Computer Vision
     Toolbox (version 2.0)
8 //Reference book name : Digital Image Processing
    book (author : Rafael C. Gonzalez and Richard E.
    Woods)
9
10 clear;
```

```
clc;
clear all;
close;

img = imread("lena.jpg");
figure(); xname("Original image");
imshow(img);

img_gray = rgb2gray(img);
img_double = im2double(img_gray);

// DCT of image using scilab  function
img_dct = dct(img_double);
figure(); xname("DCT of image using inbuilt function
   ");
imshow(img_dct);


// Creating the Twiddle Factor Matrix c
[m,n]=size(img_gray);
for x=1:m
    for y=1:n
        if x==1  // for row number one
            c(1,y)=sqrt(1/m);
        else
            c(x,y) = sqrt(2/m)*cos((%pi*(2*y+1)*x)
               /(2*m));
        end
    end
end

// DCT of image using code
result = c * img_double * c';
figure(); xname("DCT of image using code");
imshow(result);

// Inverse DCT of image using scilab  function
img_idct = idct(img_dct);
```

```
47 figure(); xname("Inverse DCT of image using inbuilt
      function");
48 imshow(img_idct);
49
50
51 // Inverse DCT of image using code
52 result_idct = inv(c) * result* inv(c');
53 figure(); xname("Inverse DCT of image using code");
54 imshow(result_idct);
55
56 //
      **********************************************************************

57 //DFT
58 // DFT of image using code
59 [m,n]=size(img_gray);
60 for x=1:m
61     for y=1:n
62         c(x,y) = exp((-2*%i*%pi*((x-1)*(y-1)))/m);
63
64     end
65 end
66
67 dft = c * img_double * inv(c);
68 res=dft;
69 dft = fftshift(dft);
70 dft = abs(dft);
71 figure(); xname("DFT of image using code");
72 imshow(dft);
73
74 //  INVERSE DFT of image using code
75 idft = inv(c) * res * c ;
76 res_idft = abs(idft);
77 figure(); xname("Inverse DFT of image using code");
78 imshow(res_idft);
```

Figure 2.1: To implement different transforms on given image

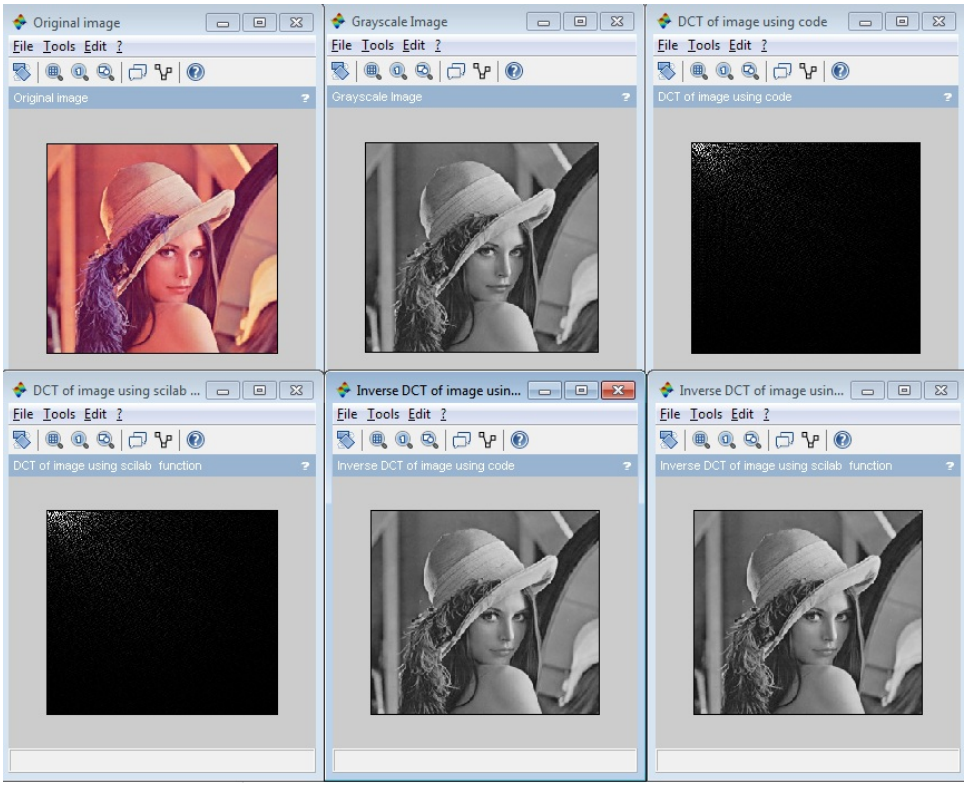

Figure 2.2: To implement different transforms on given image

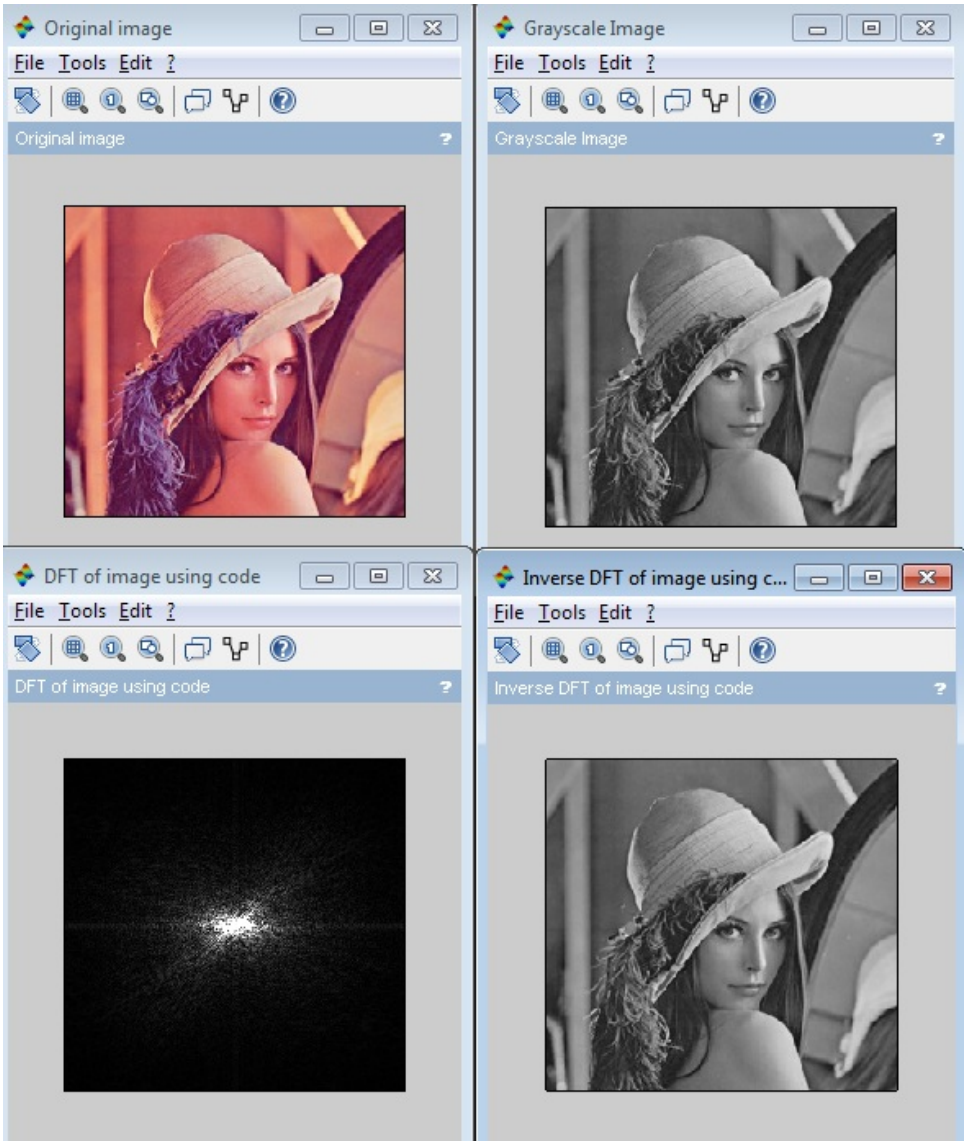

Figure 2.3: To implement different transforms on given image

Experiment: 3

To perform image enhancement by point operation/processing.

Scilab code Solution 3.3 To perform image enhancement by point operation processing

```
1 //Program Title: To perform image enhancement by
    point operation/processing.
2 //Program Description: This scilab code is used to
    perform image enhancement using point processing
    techniques like Contrast Stretching, Log
    transform, Power Law transform, Gray level
    slicing(with and without background), Bit plane
    slicing.
3
4 //Note: Details of scilab software version and OS
    version used:
5 //Tested on OS: Windows 7 SP1, 64 bit
6 //Scilab version: 6.0.1 (Tested on 64 bit version)
7 //Toolbox used: Image Processing and Computer Vision
     Toolbox (version 2.0)
8 //Reference book name : Digital Image Processing
    book (author : Rafael C.Gonzalez and Richard E.
    Woods)
```

```
 9
10 clear;
11 clc;
12 clear all;
13 close;
14
15 img=imread("ip1_texture.jpeg"); // input image 1 -->
       ip1_texture.jpeg
16 figure();xname("Original image");
17 imshow(img);
18
19 img_gray = rgb2gray(img);
20 figure();xname("Grayscale image");
21 imshow(img_gray);
22
23 ////////////////////////////    Contrast
      Stretching
      ////////////////////////////////////////
24 c = min(img_gray);
25 d=  max(img_gray);
26 a=0
27 b=255
28
29 MP = (b-a)/(d-c);
30 img_contrast = (img_gray-c).*MP+a;
31 figure(); xname("Contrast Stretched image");
32 imshow(img_contrast);
33
34 ////////////////////////////    log transform
      ////////////////////////////////////////
35 c=0.5
36 [m,n]=size(img_gray);
37 im_double = im2double(img_gray);
38 for x=1:m
39     for y=1:n
40         img_log1(x,y) = c*log(1+ im_double(x,y))
41     end
42 end
```

```
43
44 figure(); xname("Log transformed image: c= 0.5");
45 imshow(img_log1);
46
47 c=1
48 [m,n]=size(img_gray);
49 im_double = im2double(img_gray);
50 for x=1:m
51     for y=1:n
52         img_log2(x,y) = c*log(1+ im_double(x,y))
53     end
54 end
55 figure(); xname("Log transformed image: c= 1");
56 imshow(img_log2);
57
58
59 c=1.5
60 [m,n]=size(img_gray);
61 im_double = im2double(img_gray);
62 for x=1:m
63     for y=1:n
64         img_log3(x,y) = c*log(1+ im_double(x,y))
65     end
66 end
67 figure(); xname("Log transformed image: c= 1.5");
68 imshow(img_log3);
69
70 ////////////////////////////////    Power Law
      transform
      ////////////////////////////////////////////
71
72 gamma = 0.5;
73 for x=1:m
74     for y=1:n
75         img_pow1(x,y) = c*(im_double(x,y))^gamma;
76     end
77 end
78 figure(); xname("Power Law transformed image: gamma
```

```
    = 0.5");
imshow(img_pow1);

gamma = 1.5;
for x=1:m
    for y=1:n
        img_pow2(x,y) = c*(im_double(x,y))^gamma;
    end
end
figure(); xname("Power Law transformed image: gamma
    = 1.5");
imshow(img_pow2);

gamma = 5;
for x=1:m
    for y=1:n
        img_pow3(x,y) = c*(im_double(x,y))^gamma;
    end
end
figure(); xname("Power Law transformed image: gamma
    = 5");
imshow(img_pow3);

////////////////////////////////    Gray Level
    Slicing (with Background)
    ////////////////////////////////////////

for x=1:m
    for y=1:n
        if(img_gray(x,y)>50 & img_gray(x,y)<200)
            img_gray_with(x,y)=255;
        else
            img_gray_with(x,y)= im_double(x,y);
        end
    end
end
figure(); xname("Gray Level Slicing with background"
    );
```

```
111 imshow(img_gray_with);
112
113 ////////////////////////////////    Gray Level
    Slicing (without Background)
    ////////////////////////////////////////
114
115 for x=1:m
116     for y=1:n
117         if(img_gray(x,y)>50 & img_gray(x,y)<200)
118             img_gray_without(x,y)=255;
119         else
120             img_gray_without(x,y)= 0;
121         end
122     end
123 end
124 figure(); xname("Gray Level Slicing without
    background");
125 imshow(img_gray_without);
126
127 ////////////////////////////////    Bit plane
    slicing
    ////////////////////////////////////////
128 // here we use 'ip2_lena.jpg' as the input image to
    demonstrate the bit plane slicing operation in
    full effect
129
130 img=imread("ip2_lena.jpg"); // second input image
    --> 'ip2_lena.jpg'
131 img_gray = rgb2gray(img);
132
133 [m,n]=size(img_gray);
134 img_8bit = im2uint8(img_gray)
135 for x=1:m
136     for y=1:n
137         bit1(x,y) = bitget(img_8bit(x,y),1)*255;
138         bit2(x,y) = bitget(img_8bit(x,y),2)*255;
139         bit3(x,y) = bitget(img_8bit(x,y),3)*255;
140         bit4(x,y) = bitget(img_8bit(x,y),4)*255;
```

```
141             bit5(x,y) = bitget(img_8bit(x,y),5)*255;
142             bit6(x,y) = bitget(img_8bit(x,y),6)*255;
143             bit7(x,y) = bitget(img_8bit(x,y),7)*255;
144             bit8(x,y) = bitget(img_8bit(x,y),8)*255;
145         end
146 end
147
148 scf(20);
149 figure(); xname("ALL images");
150 var_rows = 2;
151 var_cols = 4;
152 subplot(var_rows,var_cols,1), imshow(bit8);
153 title('Bit Plane 7');
154 subplot(var_rows,var_cols,2), imshow(bit7);
155 title('Bit Plane 6');
156 subplot(var_rows,var_cols,3), imshow(bit6);
157 title('Bit Plane 5');
158 subplot(var_rows,var_cols,4), imshow(bit5);
159 title('Bit Plane 4');
160 subplot(var_rows,var_cols,5), imshow(bit4);
161 title('Bit Plane 3');
162 subplot(var_rows,var_cols,6), imshow(bit3);
163 title('Bit Plane 2');
164 subplot(var_rows,var_cols,7), imshow(bit2);
165 title('Bit Plane 1');
166 subplot(var_rows,var_cols,8), imshow(bit1);
167 title('Bit Plane 0');
```

Figure 3.1: To perform image enhancement by point operation processing

Figure 3.2: To perform image enhancement by point operation processing

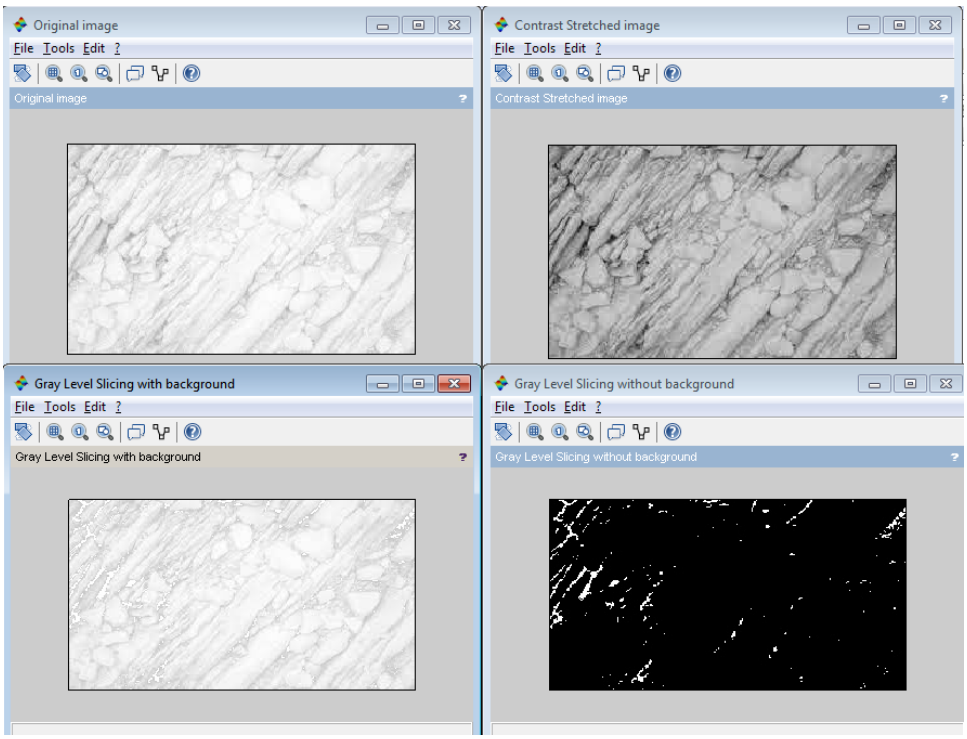

Figure 3.3: To perform image enhancement by point operation processing

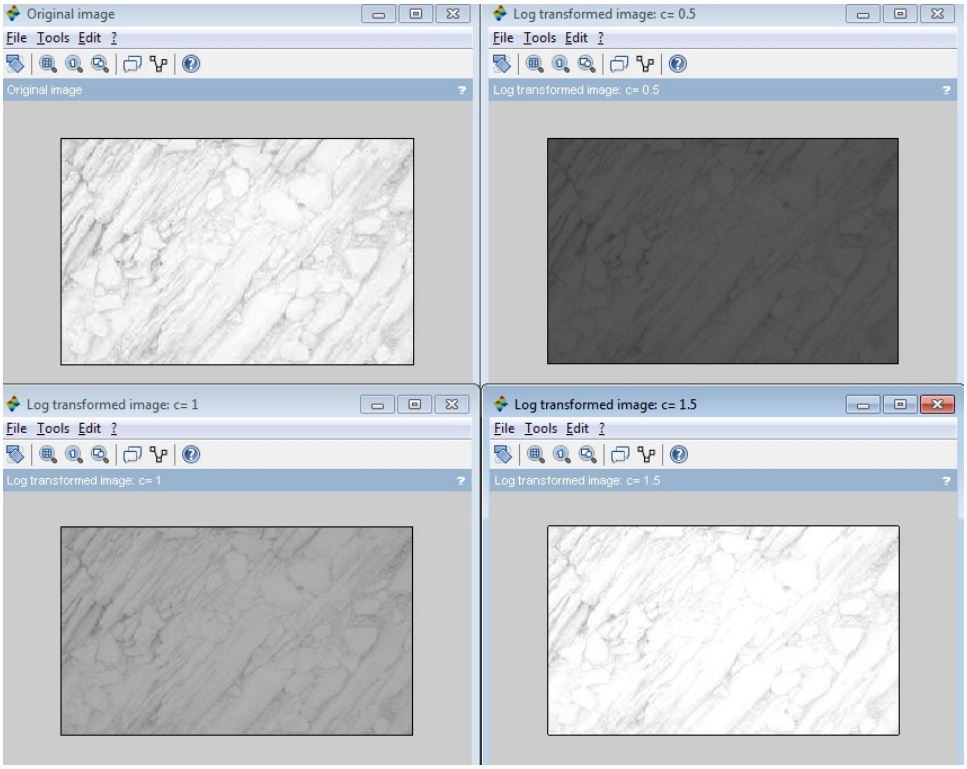

Figure 3.4: To perform image enhancement by point operation processing

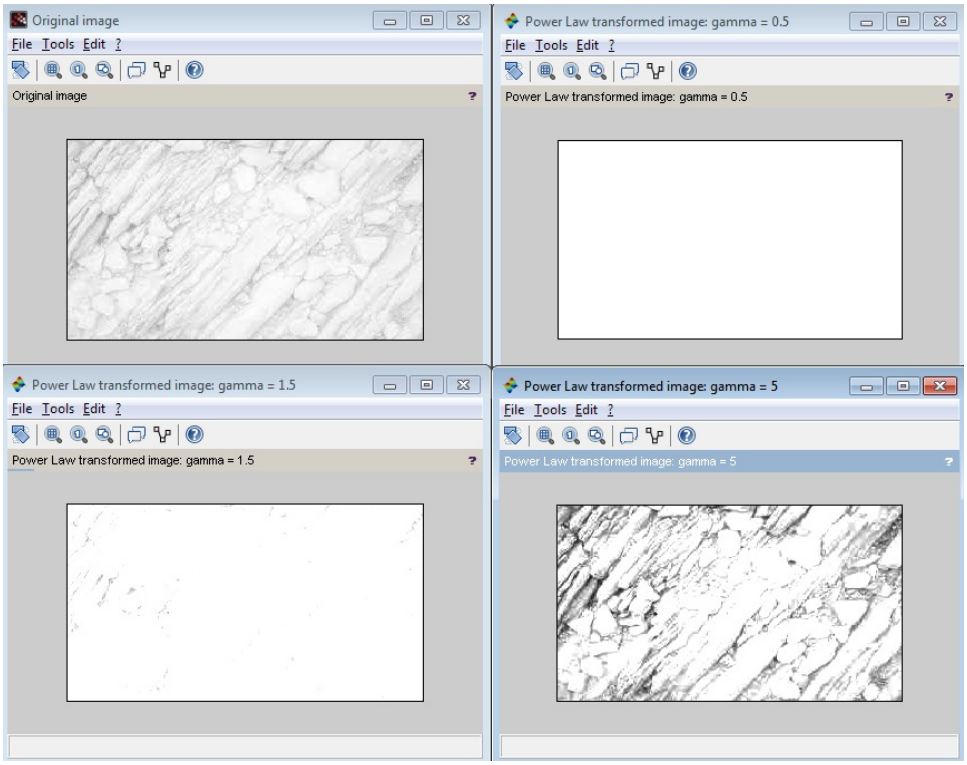

Figure 3.5: To perform image enhancement by point operation processing

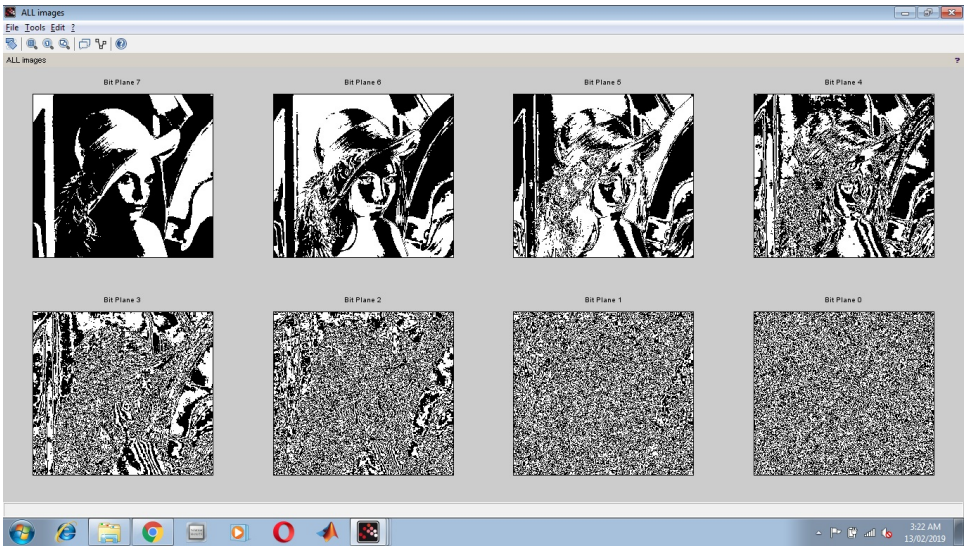

Figure 3.6: To perform image enhancement by point operation processing

Experiment: 4

To study and perform spatial and frequency domain image enhancement techniques.

Scilab code Solution 4.4 To study and perform spatial and frequency domain image enhancement techniques

```
1 //Program Title: To study and perform spatial and
    frequency domain image enhancement techniques.
2 //Program Description: This scilab code is used to
    perform image enhancement using Low pass filter,
    High pass filter, High boost filter, Gaussian
    filter and Histogram Equalization.
3
4 //Note: Details of scilab software version and OS
    version used:
5 //Tested on OS: Windows 7 SP1, 64 bit
6 //Scilab version: 6.0.1 (Tested on 64 bit version)
7 //Toolbox used: Image Processing and Computer Vision
     Toolbox (version 2.0)
8 //Reference book name : Digital Image Processing
    book (author : Rafael C. Gonzalez and Richard E.
    Woods)
```

```
9
10 clear;
11 clc;
12 clear all;
13 close;
14
15 img = imread("lena.jpg"); // input image --> lena.
     jpg
16 img_gray = rgb2gray(img);
17 img_gray = imresize(img_gray, [256, 256]);
18 figure(); xname("Gray image");
19 imshow(img_gray);
20
21 // Creating LPF mask_LPF
22 mask_LPF = ones(3,3)/9;
23 disp(mask_LPF);
24
25 img_LPF = conv2(double(img_gray), mask_LPF);
26 img_LPF = uint8(img_LPF)
27 figure(); xname("Image after LPF");
28 imshow(uint8(img_LPF));
29
30 // Creating HPF mask_HPF
31 mask_HPF = ones(3,3);
32 mask_HPF = mask_HPF*-1;
33 mask_HPF(2,2) = mask_HPF(2,2) + 9
34 disp(mask_HPF);
35 mask_HPF  = mask_HPF/9
36 disp(mask_HPF);
37
38 img_HPF = conv2(double(img_gray), mask_HPF);
39 figure(); xname("Image after HPF");
40
41 // To make negative numbers zeros
42 img_HPF = (abs(img_HPF) + img_HPF)/2;
43 img_HPF = uint8(img_HPF)
44 imshow(uint8(img_HPF));
45
```

```
46 //High Boost Filter
47 //Create HBF mask
48 mask_HBF = ones(3,3);
49 mask_HBF = mask_HBF*-1;
50 A = 5;
51 mask_HBF(2,2) = 8 + A
52 disp(mask_HBF);
53 mask_HBF  = mask_HBF/9
54 disp(mask_HBF);
55 [m,n] = size(img_gray)
56 padded_img = zeros(m+2,n+2);
57
58 //create image with zeros padded at the boundaries
59 u=2;
60 v=2;
61 for x=1:m
62     for y=1:n
63         padded_img(u,v) = img_gray(x,y);
64         v = v+1;
65     end
66     u = u+1;
67     v = 2;
68
69 end
70
71 hbf = zeros(m+2,n+2);
72
73 //applying the HBF mask on the image
74 u=1;v=1;
75 for x=2:m+1
76     for y=2:n+1
77         hbf(x,y) = padded_img(x-1,y-1)*mask_HBF(1,1)
              + padded_img(x-1,y)*mask_HBF(1,2) +
            padded_img(x-1,y+1)*mask_HBF(1,3) +
            padded_img(x,y-1)*mask_HBF(2,1) +
            padded_img(x,y)*mask_HBF(2,2) +padded_img
            (x,y+1)*mask_HBF(2,3) +padded_img(x+1,y
            -1)*mask_HBF(3,1) +padded_img(x+1,y)*
```

```
                mask_HBF(3,2) +padded_img(x+1,y+1)*
                mask_HBF(3,3) ;
78          v=v+1;
79      end
80      u=u+1;
81  end
82
83  //remove padded zeros
84  for x=2:m+1
85      for y=2:n+1
86          hbf_img(x-1,y-1) = hbf(x,y);
87      end
88  end
89
90  //convert all negative values to zeros
91  hbf_img = (abs(hbf_img)+hbf_img)/2;
92
93  //Display HBF image
94  figure();
95  xname("HBF image");
96  imshow(uint8(hbf_img));
97
98
99  //Gaussian Filtering
100 N = 3
101 sigma = 1
102
103 ind = -floor(N/2) : floor(N/2);
104 disp(ind)
105 [X Y] = meshgrid(ind, ind)
106
107 //create gaussian Mask
108 mask_gaussian = (1/(2*%pi*sigma))*exp(-(X.^2 + Y.^2)
        / (2*sigma*sigma));
109 mask_gaussian = [[1, 2 , 1];[2,4,2];[1,2,1]];
110 disp(mask_gaussian)
111 // Normalize so that total area (sum of all weights)
        is 1
```

```
112 mask_gaussian = mask_gaussian / sum(mask_gaussian(:)
    );
113 disp(mask_gaussian)
114
115 img_gaussian = conv2(double(img_gray), mask_gaussian
    );
116 figure();
117 xname("Image after Gaussian Filter");
118 imshow(uint8(img_gaussian));
119 imwrite(uint8(img_gaussian),'
    noise_filtered_img_gaussian.jpg');
120
121
122
123 // HISTOGRAM EQUALIZATION
124 [count, cells]=imhist(img_gray);
125 k=256
126 count = count/(k*k);
127 x= [0:1:(k-1)]';
128 figure();
129 title('original Histogram');
130 plot2d3(x,[count]);
131
132 cdf = zeros(k,1);
133 sum1=0;
134 for m= 1:k
135     sum1 = sum1 + count(m,1);
136     cdf(m,1)= sum1;
137 end
138
139 cdf_multiplied = cdf*(k-1);
140 for m= 1:k
141     new_gray_levels(m,1) = round(cdf_multiplied(m,1)
        );
142 end
143
144 j= new_gray_levels(1,1);
145 for m = 1:k
```

```
146     if m==1 then
147         pix(m,1) = count(m,1)*k*k;
148     end
149     if m ~= 1
150         if j == new_gray_levels(m,1)
151             pix(m,1) = 0;
152         else
153             pix(m,1) = count(m,1)*k*k;
154             j= new_gray_levels(m,1);
155         end
156     end
157 end
158
159 for m=1:k
160     if pix(m,1)==0
161         var = m
162         while pix(var,1)==0
163             if var>1
164                 var = var -1
165             else
166                 break
167             end
168         end
169         pix(var,1)= pix(var,1)+count(m,1)*k*k;
170     end
171 end
172
173 res = zeros(k,k)
174 for m = 1:k
175     for n = 1:k
176         old = img_gray(m,n)
177         for j = 1:k
178             if old == j-1
179                 res(m,n) = new_gray_levels(j,1)
180             end
181         end
182     end
183 end
```

```
184 figure();
185 xname('Equalised_Histogram image');
186 imshow(uint8(res));
187 imwrite(uint8(res),'equal_hist_img.jpg');
188 [count, cells]=imhist(uint8(res));
189
190 count = count/(k*k);
191 x= [0:1:(k-1)]';
192 figure();
193 title('Equalised_Histogram');
194 plot2d3(x,[count]);
```

Figure 4.1: To study and perform spatial and frequency domain image enhancement techniques

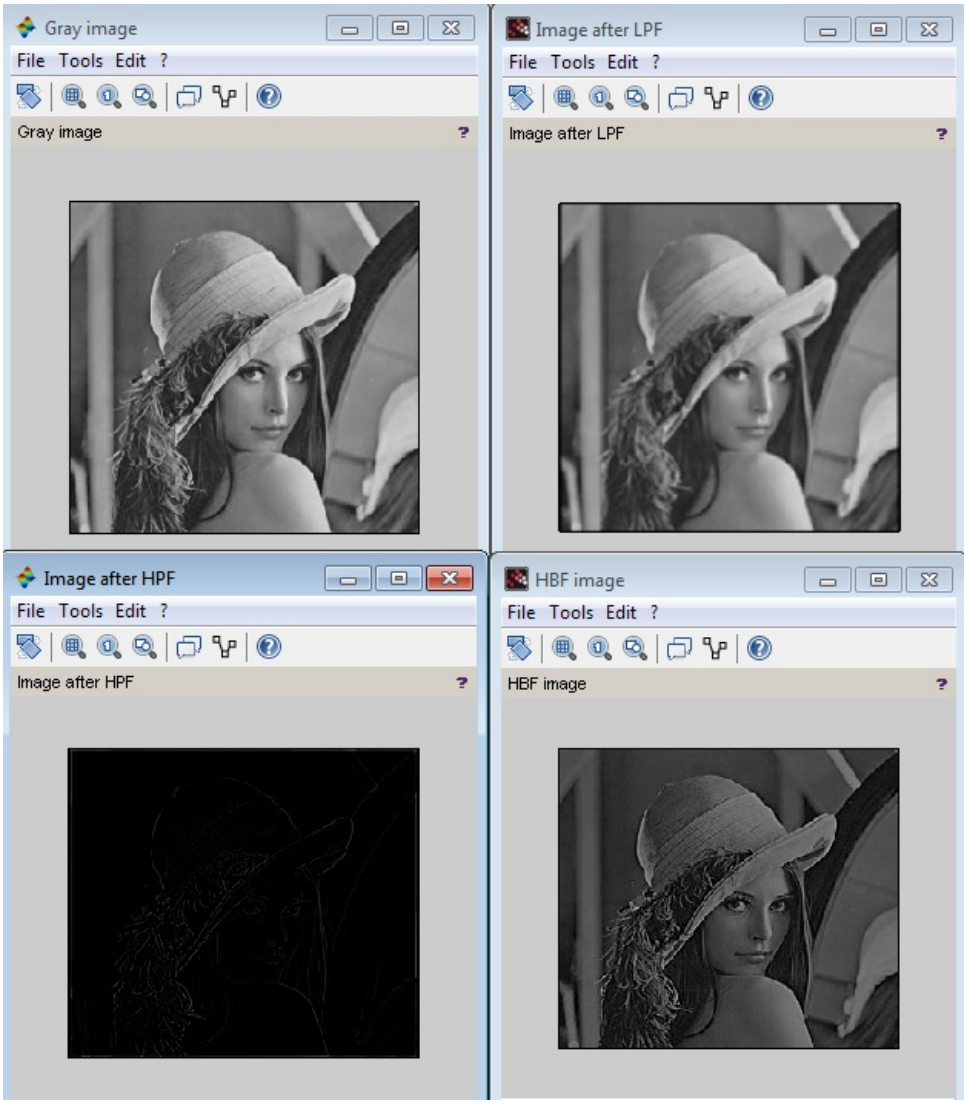

Figure 4.2: To study and perform spatial and frequency domain image enhancement techniques

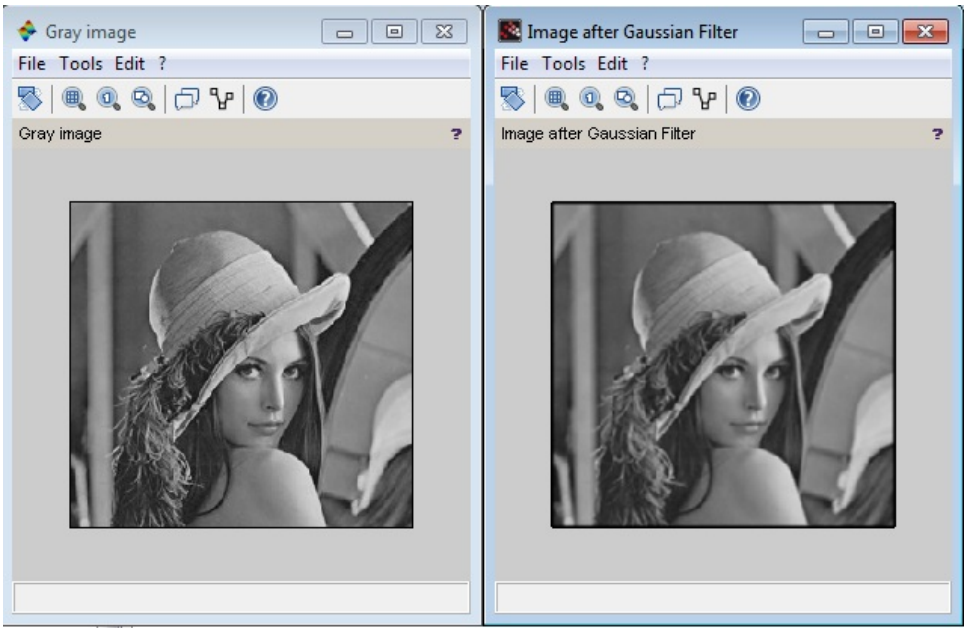

Figure 4.3: To study and perform spatial and frequency domain image enhancement techniques

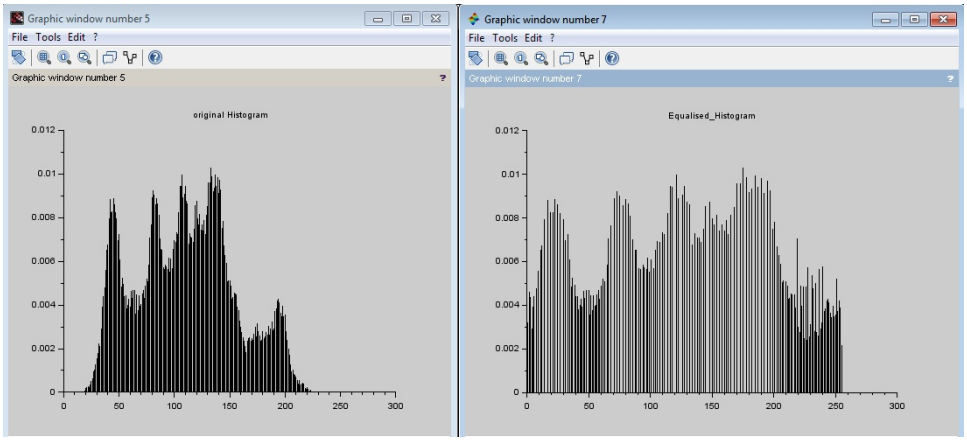

Figure 4.4: To study and perform spatial and frequency domain image enhancement techniques

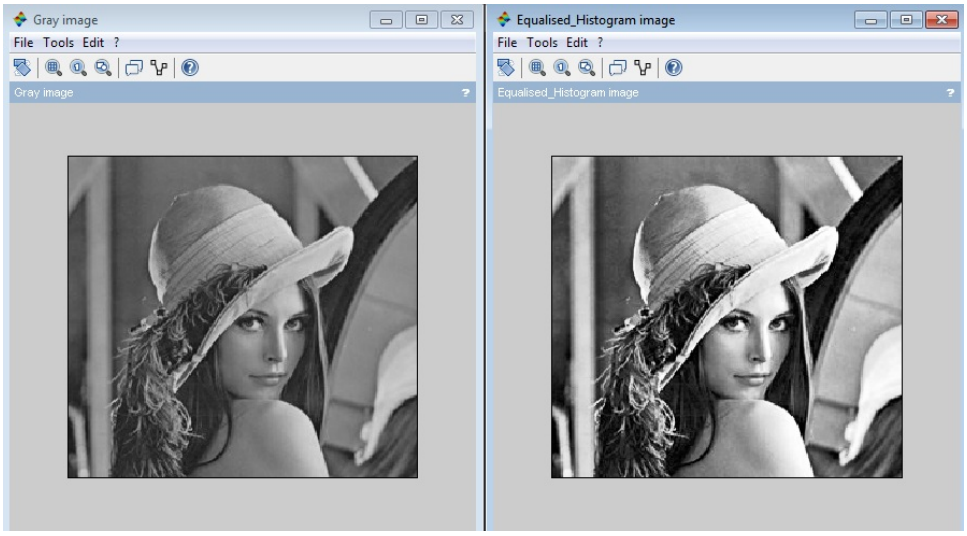

Figure 4.5: To study and perform spatial and frequency domain image enhancement techniques

Figure 4.6: To study and perform spatial and frequency domain image enhancement techniques

Experiment: 5

To study and perform various image segmentation techniques.

Scilab code Solution 5.5 To study and perform various segmentation techniques

```
1 //Program Title: To study and perform various image
    segmentation techniques.
2 //Program Description: This scilab code is used to
    perform segmentation operations like edge
    detection using sobel, canny , prewitt operators,
     thresholding and Morphoogical operations like
    dilation, erosion, opening, closing on an image.
3
4 //Note: Details of scilab software version and OS
    version used:
5 //Tested on OS: Windows 8.1 Pro, 64 bit
6 //Scilab version: 5.5.2 (Tested on 64 bit version)
7 //Toolbox used: SIVP - Scilab Image and Video
    Processing Toolbox (Version 0.5.3.2)
8 //Reference book name : Digital Image Processing (
    author : Rafael C.Gonzalez and Richard E.Woods)
9
10 clear;
```

```
clc;
clear all;
close;

img = imread("ip1_lena.jpg");      // input image 1
  --> ip1_lena.jpg
img_gray = rgb2gray(img);
figure();
xname("Input image 1");
imshow(img_gray);

// EDGE DETECTION
[v,h] = size(img_gray);

v_sobel = [-1, 0, 1; -2,0,2; -1,0,1];
disp(v_sobel);

img_gray_v = conv2(double(img_gray), v_sobel);
figure();
xname("Vertical Edge Detection image");
imshow(img_gray_v);
imwrite(img_gray_v,'ver.jpg')

h_sobel = [-1, -2, -1; 0,0,0; 1,2,1];
disp(h_sobel);
img_gray_h = conv2(double(img_gray), h_sobel);
figure();
xname("Horizontal Edge Detection image");
imshow(img_gray_h);
imwrite(img_gray_h,'hori.jpg')

img_res = img_gray_h + img_gray_v;
figure();
xname("Sum of Edge Detection image");
imshow(img_res);
imwrite(img_res,'sum.jpg')

// Edge Detection using in-built functions
```

```
48 E = edge(img_gray, 'sobel');
49 figure();
50 xname("Sobel edge detection");
51 imshow(E);
52 imwrite(E,'sobel.jpg')
53
54 E2 = edge(img_gray, 'canny', [0.06, 0.2]);
55 figure();
56 xname("Canny edge detection");
57 imshow(E2);
58 imwrite(E2,'canny.jpg')
59
60 E3 = edge(img_gray, 'prewitt');
61 figure();
62 xname("Prewitt edge detection");
63 imshow(E3);
64 imwrite(E3,'prewitt.jpg')
65
66 //THRESHOLDING
67 img_thresh = int(img_gray/128)*255;
68 figure();
69 xname("Global Thresholding");
70 imshow(img_thresh);
71 imwrite(img_thresh,'threshold.jpg');
72
73 //*********************************
     MORPHOLOGICAL PROCESSING
     *************************
74 img = imread("ip2_thanks.jpg");      // input image 2
     --> ip2_thanks.jpg
75 img_gray = rgb2gray(img);
76 figure();
77 xname("Input image 2");
78 imshow(img_gray);
79 // Dilation, Erosion, Opening, Closing
80 [m,n] = size(img_gray)
81 padded_img = zeros(m+2,n+2);
82
```

```
83  //create image with zeros padded at the boundaries
84  u=2;
85  v=2;
86  for x=1:m
87      for y=1:n
88          padded_img(u,v) = img_gray(x,y);
89          v = v+1;
90      end
91      u = u+1;
92      v = 2;
93
94  end
95
96  arr = zeros(1,9);
97
98  u=1;v=1;
99  for x=2:m+1
100     for y=2:n+1
101         arr(1,1) = padded_img(x-1,y-1);
102         arr(1,2) = padded_img(x-1,y);
103         arr(1,3) = padded_img(x-1,y+1);
104         arr(1,4) = padded_img(x,y-1);
105         arr(1,5) = padded_img(x,y);
106         arr(1,6) = padded_img(x,y+1);
107         arr(1,7) = padded_img(x+1,y-1);
108         arr(1,8) = padded_img(x+1,y);
109         arr(1,9) = padded_img(x+1,y+1);
110         img_max(x,y) = max(arr);
111         img_min(x,y) = min(arr);
112         v=v+1;
113     end
114     u=u+1;
115 end
116
117 //remove padded zeros
118 for x=2:m+1
119     for y=2:n+1
120         dilated_img(x-1,y-1) = img_max(x,y);
```

```
121     end
122 end
123
124 //remove padded zeros
125 for x=2:m+1
126     for y=2:n+1
127         eroded_img(x-1,y-1) = img_min(x,y);
128     end
129 end
130
131
132
133
134
135 //Display dilated image
136 figure();
137 xname("Dilated image");
138 imshow(uint8(dilated_img));
139 imwrite(uint8(dilated_img),'dilate.jpg')
140
141 //Display eroded image
142 figure();
143 xname("Eroded image");
144 imshow(uint8(eroded_img));
145 imwrite(uint8(eroded_img),'erode.jpg')
146
147 ////////////////////////////////////////////////
148 //Opening:  erosion followed dilation
149 padded_img = zeros(m+2,n+2);
150
151 //create image with zeros padded at the boundaries
152 u=2;
153 v=2;
154 for x=1:m
155     for y=1:n
156         padded_img(u,v) = eroded_img(x,y);
157         v = v+1;
158     end
```

```
159      u = u+1;
160      v = 2;
161
162  end
163
164  arr = zeros(1,9);
165
166  u=1;v=1;
167  for x=2:m+1
168      for y=2:n+1
169          arr(1,1) = padded_img(x-1,y-1);
170          arr(1,2) = padded_img(x-1,y);
171          arr(1,3) = padded_img(x-1,y+1);
172          arr(1,4) = padded_img(x,y-1);
173          arr(1,5) = padded_img(x,y);
174          arr(1,6) = padded_img(x,y+1);
175          arr(1,7) = padded_img(x+1,y-1);
176          arr(1,8) = padded_img(x+1,y);
177          arr(1,9) = padded_img(x+1,y+1);
178          img_max(x,y) = max(arr);
179          v=v+1;
180      end
181      u=u+1;
182  end
183
184  //remove padded zeros
185  for x=2:m+1
186      for y=2:n+1
187          opening_img(x-1,y-1) = img_max(x,y);
188      end
189  end
190
191  //Display Closing image
192  figure();
193  xname("Opening image");
194  imshow(uint8(opening_img));
195  imwrite(uint8(opening_img),'opening.jpg')
196
```

```
197 ////////////////////////////////////////////////
198 //Closing:  dilation followed erosion
199 padded_img = zeros(m+2,n+2);
200
201 //create image with zeros padded at the boundaries
202 u=2;
203 v=2;
204 for x=1:m
205     for y=1:n
206         padded_img(u,v) = dilated_img(x,y);
207         v = v+1;
208     end
209     u = u+1;
210     v = 2;
211
212 end
213
214 arr = zeros(1,9);
215
216 u=1;v=1;
217 for x=2:m+1
218     for y=2:n+1
219         arr(1,1) = padded_img(x-1,y-1);
220         arr(1,2) = padded_img(x-1,y);
221         arr(1,3) = padded_img(x-1,y+1);
222         arr(1,4) = padded_img(x,y-1);
223         arr(1,5) = padded_img(x,y);
224         arr(1,6) = padded_img(x,y+1);
225         arr(1,7) = padded_img(x+1,y-1);
226         arr(1,8) = padded_img(x+1,y);
227         arr(1,9) = padded_img(x+1,y+1);
228         img_min(x,y) = min(arr);
229         v=v+1;
230     end
231     u=u+1;
232 end
233
234 //remove padded zeros
```

```
235 for x=2:m+1
236     for y=2:n+1
237         closing_img(x-1,y-1) = img_min(x,y);
238     end
239 end
240
241 //Display Closing image
242 figure();
243 xname("Closing image");
244 imshow(uint8(closing_img));
245 imwrite(uint8(closing_img),'closing.jpg')
```

Figure 5.1: To study and perform various segmentation techniques

Figure 5.2: To study and perform various segmentation techniques

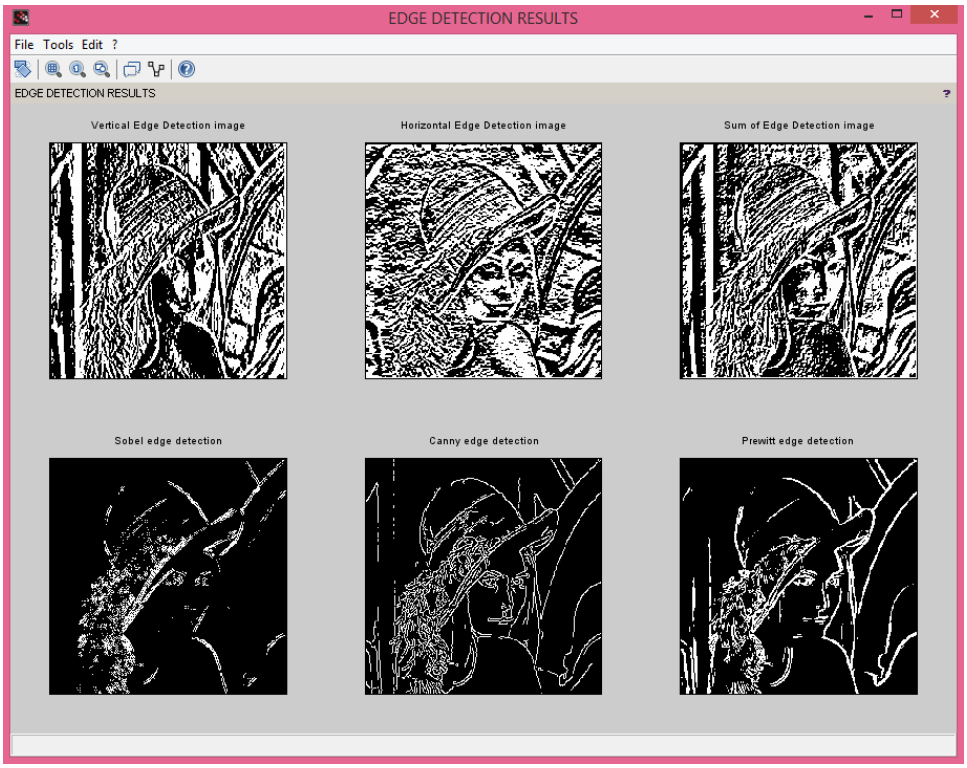

Figure 5.3: To study and perform various segmentation techniques

Figure 5.4: To study and perform various segmentation techniques

Figure 5.5: To study and perform various segmentation techniques