# Scilab Manual for Digital Image Processing by Dr B.Chandra Mohan Electronics Engineering Bapatla Engineering College[1]

Solutions provided by
Mr R.Senthilkumar- Assistant Professor
Electronics Engineering
Institute of Road and Transport Technology

November 18, 2026

[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 Scilab Solutions 4

1 Histogram display and histogram equalization 6

2 Kernel processing on images leading to image enhancement 10

3 Display of 2D filters frequency responses and processing the images using these filters 14

4 Implementation of Airthmetic Coding for images 18

5 Basic JPEG algorithm implementation 20

6 DPCM encoding and decoding of images 25

7 Simple image watermarking algorithms using LSB substitution 30

8 Simple content based image retrieval using various distance metrics 36

9 Image segmentation algorithms using Snakes 41

10 Color images manipulations, reading and writing of color images 44

11 Color image enhancements 47

12 Color image histogram manipulation 51

13 LOG Masks implementation for gray and color images 55

14 Special effects implementation on grey and color images 57

15 Simple video reading and writing .avi formats and manipulation of video frames 62

# List of Experiments


| | | |
|---|---|---|
| Solution 1.1 | Exp1 | 6 |
| Solution 2.1 | Exp2 | 10 |
| Solution 3.1 | Exp3 | 14 |
| Solution 4.1 | Exp4 | 18 |
| Solution 5.1 | Exp5 | 20 |
| Solution 6.1 | Exp6 | 25 |
| Solution 7.1 | Exp7 | 30 |
| Solution 8.1 | Exp8 | 36 |
| Solution 9.1 | Exp9 | 41 |
| Solution 10.1 | Exp10 | 44 |
| Solution 11.1 | Exp11 | 47 |
| Solution 12.1 | Exp12 | 51 |
| Solution 13.1 | Exp13 | 55 |
| Solution 14.1 | Exp14 | 57 |
| Solution 15.1 | Exp15a | 62 |
| Solution 15.2 | Exp15b | 63 |
| AP 1 | Rotate Image 90 degree | 66 |
| AP 2 | Histogram of Gray images | 67 |
| AP 3 | Image Enhancement | 67 |
| AP 4 | Inverse Zig Zag scanning of pixels | 69 |
| AP 5 | Zig Zag Scanning of Pixels | 70 |
| AP 6 | 2D Fast Fourier Trasnform | 70 |
| AP 7 | Inverse 2D Fast Fourier Transform | 71 |

# List of Figures


1.1 Exp1 . . . . . 8
1.2 Exp1 . . . . . 9

2.1 Exp2 . . . . . 12
2.2 Exp2 . . . . . 13

3.1 Exp3 . . . . . 16
3.2 Exp3 . . . . . 17

6.1 Exp6 . . . . . 28
6.2 Exp6 . . . . . 29

7.1 Exp7 . . . . . 34
7.2 Exp7 . . . . . 35

8.1 Exp8 . . . . . 39
8.2 Exp8 . . . . . 40

11.1 Exp11 . . . . . 49
11.2 Exp11 . . . . . 50

12.1 Exp12 . . . . . 53
12.2 Exp12 . . . . . 54

14.1 Exp14 . . . . . 60
14.2 Exp14 . . . . . 61

# Experiment: 1

# Histogram display and histogram equalization

**Scilab code Solution 1.1** Exp1

```
1  //Program 1 Histogram display and histogram
       equalization
2  //Software version
3  //OS Windows7
4  //Scilab5.4.1
5  //Image Processing Design Toolbox 8.3.1-1
6  //Scilab Image and Video Proccessing toolbox
       0.5.3.1-2
7  clc;
8  clear;
9  close;
10 //a=imread('C:\Users\senthilkumar\Desktop\
       Chandra_Mohan_LAB\Digital_Image_ProcessingLab\
       tire.tif');//Image Path
11 a=imread('C:\Users\senthilkumar\Desktop\
       Chandra_Mohan_LAB\Digital_Image_ProcessingLab\
       tire.jpeg')
12 [m n]=size(a);
13 for i=1:256
```

```
14      b(i)=length(find(a==(i-1)));
15  end
16  pbb=b/(m*n);
17  pb(1)=pbb(1);
18  for i=2:256
19      pb(i)=pb(i-1)+pbb(i);
20  end
21
22  s=pb*255;
23  sb=uint8(round(s));
24  index =0;
25  for i=1:m
26      for j=1:n
27          index = double(a(i,j))+1;//convert it to
                double
28          //otherwise index = 255+1 =0
29          hea(i,j)= sb(index);//histogram equalization
30      end
31  end
32  figure,
33  ShowImage(a,'Original Image')//IPD toolbox
34  title('Original Image')
35  figure
36  plot2d3('gnn',[1:256],b)
37  title('Histogram of the Image')
38  figure
39  ShowImage(hea,'Image after Histogram equalization')
      //IPD toolbox
40  title('Image after Histogram equalization')
```

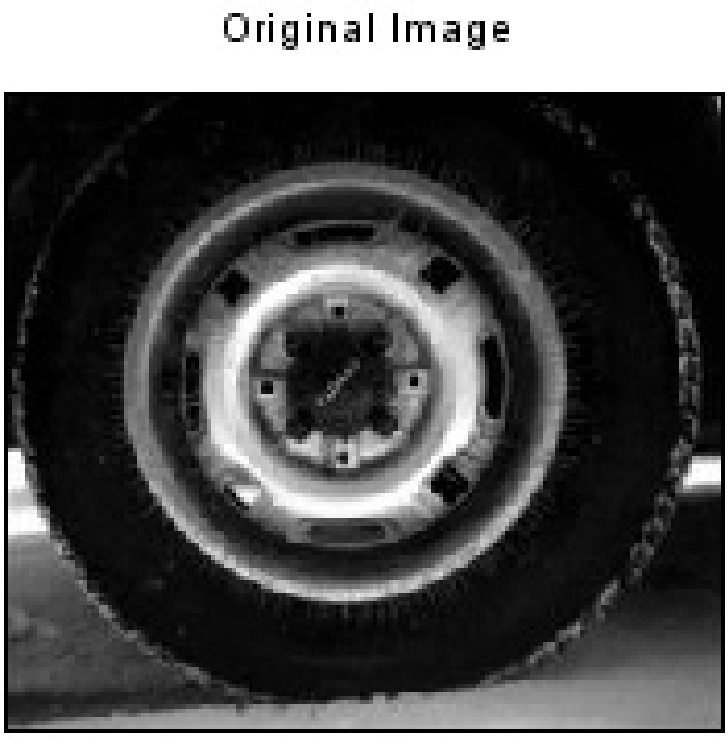



Figure 1.1: Exp1

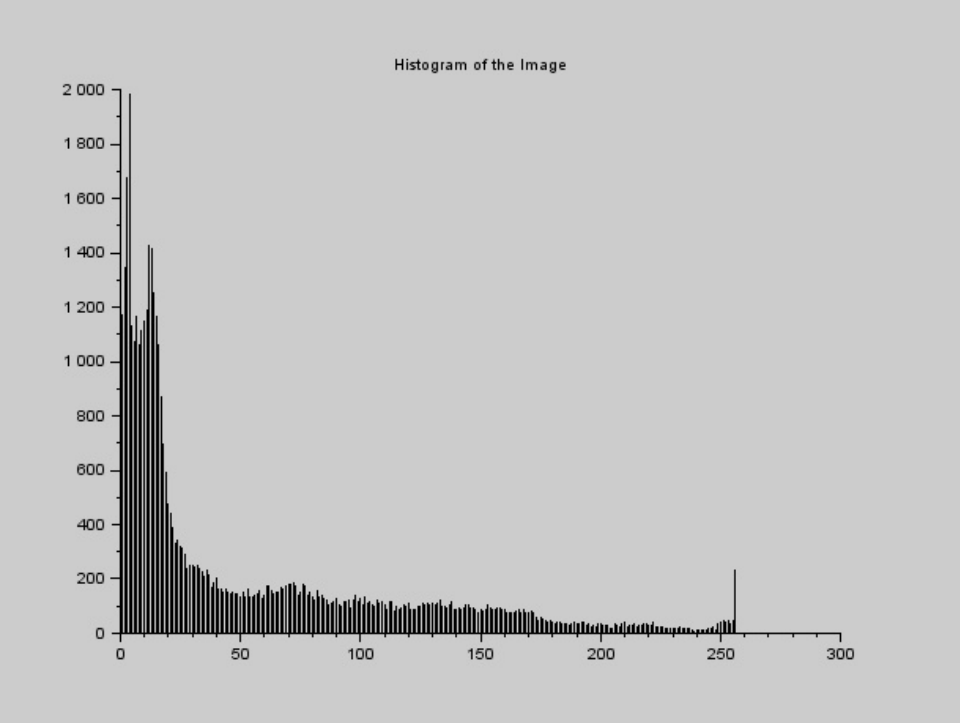



Figure 1.2: Exp1

# Experiment: 2

# Kernel processing on images leading to image enhancement

**Scilab code Solution 2.1** Exp2

```
1  //Program 2.Kernel processing on images leading to
       image enhancement.
2  //Software version
3  //OS Windows7
4  //Scilab5.4.1
5  //Image Processing Design Toolbox 8.3.1-1
6  //Scilab Image and Video Proccessing toolbox
       0.5.3.1-2
7  clc
8  clear
9  close
10 a=imread('C:\Users\senthilkumar\Desktop\
       Chandra_Mohan_LAB\Digital_Image_ProcessingLab\
       cktnoise.jpeg');//SIVP toolbox
11 ks=input('enter the size of the kernel 1 for 1 1  3
       for 3 3  ...');//kernel size 3x3
12 [m n]=size(a);
13
14 a1=zeros(m+ks-1,n+ks-1);
```

```
[m1 n1]=size(a1);
x=floor(ks/2);
a1(1+x:m1-x,1+x:n1-x)=a;
b=[];
c=[];

for i=1+x:m1-x
    for j=1+x:n1-x
        t=a1(i-x:i+x,j-x:j+x);
        men=sum(sum(t))/(ks*ks);
        med=median(t(:));
        b(i-x,j-x)=men;
        c(i-x,j-x)=med;
    end
end

figure
ShowImage(a,'Noised image(before enhancement)');//
   IPD toolbox
title('Noised image(before enhancement)');
figure
ShowImage(uint8(b),'enhancement with mean filtering'
   );//IPD toolbox
title('enhancement with mean filtering');
figure
ShowImage(uint8(c),'enhancement with median
   filtering');//IPD toolbox
title('enhancement with median filtering');
//RESULT
//enter the size of the kernel 1 for 1 1  3 for 3 3
    ...3
```

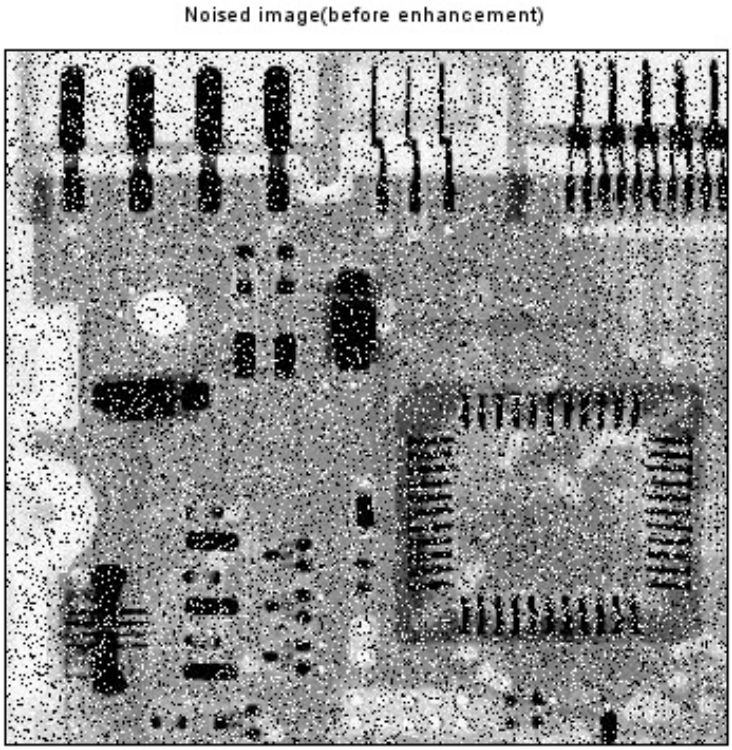



Figure 2.1: Exp2

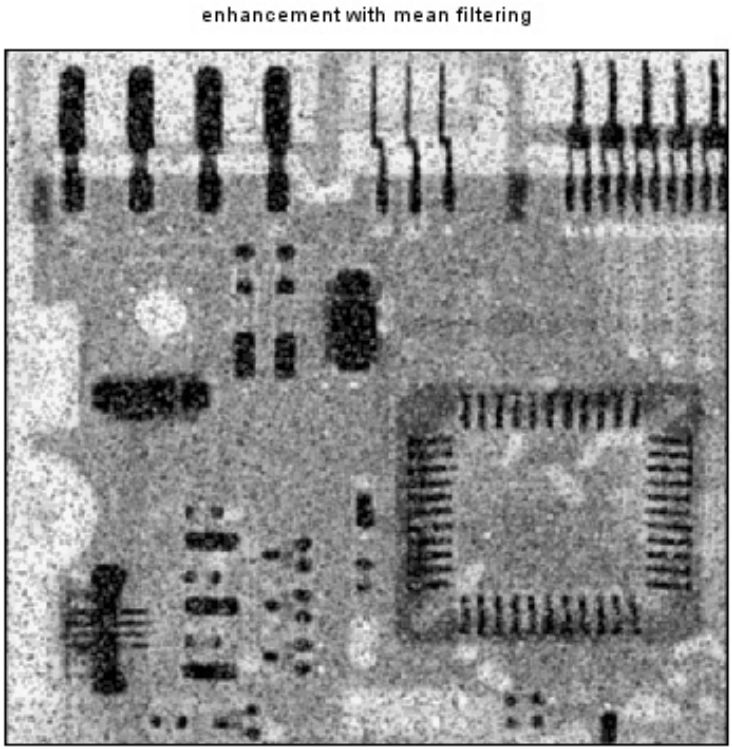



Figure 2.2: Exp2

# Experiment: 3

# Display of 2D filters frequency responses and processing the images using these filters

check Appendix AP 6 for dependency:

`fft2d.sce`

check Appendix AP 7 for dependency:

`ifft2d.sce`

**Scilab code Solution 3.1** Exp3

```
1 //Program 3: Display of 2D filters frequency
    responses and processing the images using these
    filters
2 //Reference: "Digital Image Processing",Dr.S.
    Jayaraman,S.Esakkirajan,T.Veerakumar,TMH,2011
3 //Note: The in-built scilab functions fft2d and
    ifft2d are not working properly
4 //It give wrong results.
5 //Use My functions for 2D-FFT and 2D-IFFT.
6 //Software version
```

```
7  //OS Windows7
8  //Scilab5.4.1
9  //Image Processing Design Toolbox 8.3.1-1
10 //Scilab Image and Video Proccessing toolbox
       0.5.3.1-2
11 clc;
12 close;
13 clear;
14 exec('C:\Users\senthilkumar\Desktop\
       Chandra_Mohan_LAB\Digital_Image_ProcessingLab\
       fft2d.sce')
15 exec('C:\Users\senthilkumar\Desktop\
       Chandra_Mohan_LAB\Digital_Image_ProcessingLab\
       ifft2d.sce')
16 im1 = imread('C:\Users\senthilkumar\Desktop\
       Chandra_Mohan_LAB\Digital_Image_ProcessingLab\
       balloonsnoisy.png');//colour noise image
17 im = rgb2gray(im1);//gray noise image
18 fc = 100; //cut off frequency -more features choose
       high cutoff frequency
19 n = 1;//filter order =1
20 [co,ro]= size(im);
21 cx = round(co/2); //centre of the image
22 cy = round(ro/2);
23 IM = fft2d(double(im));
24 imf = fftshift(IM);
25 H = zeros(co,ro);
26 for i = 1:co
27     for j = 1:ro
28         d = (i-cx).^2+(j-cy).^2;
29         H(i,j) = 1/(1+((d/fc/fc).^(2*n))); //Low
               Pass Butterworth First Order filter
30     end
31 end
32 out_im = imf.*H;
33 out = abs(ifft2d(out_im));
34 out = uint8(out);
35 figure
```



Figure 3.1: Exp3

```
36 ShowColorImage(im1,'Colour Noisy Image')
37 figure
38 ShowImage(im,'Gray Noise Image')
39 figure
40 ShowImage(H,'Low Pass Filter Frequency Response')
41 figure
42 ShowImage(out,'Filtered Image')
```



Figure 3.2: Exp3

# Experiment: 4

# Implementation of Airthmetic Coding for images

**Scilab code Solution 4.1** Exp4

```
1 // Program 4. Implementation of arithmetic coding
    for images
2 //Note 1: In order to run this program download
    Huffman toolbox from
3 //scilab atoms
4 //Note 2: The Huffman atom is used to encode images
    of small size only
5 //Software version
6 //OS Windows7
7 //Scilab5.4.1
8 //Image Processing Design Toolbox 8.3.1-1
9 //Scilab Image and Video Proccessing toolbox
    0.5.3.1-2
10 clear;
11 clc;
12 close;
13 //A=testmatrix('frk',10)+1;
14 a = imread('C:\Users\senthilkumar\Desktop\
    Chandra_Mohan_LAB\Digital_Image_ProcessingLab\
```

```
    cameraman.jpeg');
A = imresize(a,[16 16]); //Only Image of small size
    is possible to call huffcode
B = size(A);
A=A(:).';
A = double(A);
[QT,QM]=huffcode(A); //Huffman Encoding
disp('compressed Bit sequence:');
disp(QT);
disp('Code Table:');
disp(QM);
// Now, the reverse operation
C = huffdeco(QT,QM); //Huffman Decoding
for i=1:B(1)
    E(i,1:B(2))= C((i-1)*B(2)+1:i*B(2));
end
D = E';
E = imresize(D,[32,32]);
figure
ShowImage(a,'Original cameraman Image 256x256')
figure
ShowImage(E,'Reconstructed cameraman Image 256x256')
    ;
```

# Experiment: 5

# Basic JPEG algorithm implementation

check Appendix AP 4 for dependency:

`izigzag5.sci`

check Appendix AP 5 for dependency:

`zigzag5.sci`

**Scilab code Solution 5.1** Exp5

```
1  // Program 5. Basic JPEG algorithm implementation
2  //Software version
3  //OS Windows7
4  //Scilab5.4.1
5  //Image Processing Design Toolbox 8.3.1-1
6  //Scilab Image and Video Proccessing toolbox
       0.5.3.1-2
7  close
8  clear;
9  clc;
10 exec('C:\Users\senthilkumar\Desktop\
       Chandra_Mohan_LAB\Digital_Image_ProcessingLab\
       zigzag_5.sci')
```

```
11 exec('C:\Users\senthilkumar\Desktop\
     Chandra_Mohan_LAB\Digital_Image_ProcessingLab\
     izigzag_5.sci')
12 I = imread('C:\Users\senthilkumar\Desktop\
     Chandra_Mohan_LAB\Digital_Image_ProcessingLab\
     cameraman.jpeg'); //256x256 image
13 I = imresize(I,0.25);//reduced to 64x64 image [in
     order to reduce the computation time]
14 [m,n]=size(I);// Finding the dimensions of the image
      file.
15 I=double(I);
16 q= [16 11 10 16 24 40 51 61;
17      12 12 14 19 26 58 60 55;
18      14 13 16 24 40 57 69 56;
19      14 17 22 29 51 87 80 62;
20      18 22 37 56 68 109 103 77;
21      24 35 55 64 81 104 113 92;
22      49 64 78 87 103 121 120 101;
23      72 92 95 98 112 100 103 99];
24 N=8;                          // Block size for which
     DCT is Computed.
25 M=8;
26 I_Trsfrm.block=zeros(N,M);  // Initialising the DCT
     Coefficients Structure Matrix "I_Trsfrm" with the
      required dimensions.
27 for a=1:m/N
28     for b=1:n/M
29         for k=1:N
30             for l=1:M
31                 Mean_Sum=0;
32                 //2D-Discrete Cosine Transform
                      ///////
33                 for i=1:N
34                     for j=1:M
35                         Mean_Sum = Mean_Sum+double(I
                             (N*(a-1)+i,M*(b-1)+j))*
                             cos(%pi*(k-1)*(2*i-1)/(2*
                             N))*cos(%pi*(l-1)*(2*j-1)
```

```
                                    /(2*M));
36                      end
37                  end
38                  ///////////////
39                  if k==1
40                      Mean_Sum = Mean_Sum*sqrt(1/N);
41                  else
42                      Mean_Sum = Mean_Sum*sqrt(2/N);
43                  end
44                  if l==1
45                      Mean_Sum = Mean_Sum*sqrt(1/M);
46                  else
47                      Mean_Sum = Mean_Sum*sqrt(2/M);
48                  end
49                  I_Trsfrm(a,b).block(k,l)= Mean_Sum;
50              end
51          end
52          // Normalizing the DCT Matrix and Quantizing
                the resulting values.
53          I_Trsfrm(a,b).block=round(I_Trsfrm(a,b).
               block./q);
54      end
55  end
56  I_zigzag.block = zeros(N,M);
57  for a= 1:m/N
58      for b = 1:n/M
59          I_zigzag(a,b).block = zigzag_5(I_Trsfrm(a,b)
               .block);
60      end
61  end
62  I_rec_Trnsfm.block = zeros(N,M);
63  for a= 1:m/N
64      for b = 1:n/M
65          I_rec_Trnsfm(a,b).block = izigzag_5(I_zigzag
               (a,b).block);
66      end
67  end
68  // Denormalizing the Reconstructed Tranform matrix
```

```
    using the same
// Normalization matrix.
for a=1:m/N
    for b=1:n/M
        I_rec_Trnsfm(a,b).block =(I_rec_Trnsfm(a,b).
           block).*q;
    end
end
//Inverse 2D-DCT
for a=1:m/N
    for b=1:n/M
        for i=1:N
            for j=1:M
                Mean_Sum =0;
                for k=1:N
                    for l=1:M
                        if k==1
                           temp =double(sqrt(1/2)*
                              I_rec_Trnsfm(a,b).
                              block(k,l))*cos(%pi*(k
                              -1)*(2*i-1)/(2*N))*cos
                              (%pi*(l-1)*(2*j-1)/(2*
                              M));
                        else
                            temp = double(
                               I_rec_Trnsfm(a,b).
                               block(k,l))*cos(%pi*(
                               k-1)*(2*i-1)/(2*N))*
                               cos(%pi*(l-1)*(2*j-1)
                               /(2*M));
                        end
                        if l==1
                            temp = temp*sqrt(1/2);
                        end
                        Mean_Sum = Mean_Sum+temp;
                    end
                end
                Mean_Sum = Mean_Sum*(2/sqrt(M*N));
```

```
95                      I_rec((a-1)*N+i,(b-1)*M+j)= Mean_Sum
                           ;
96                  end
97              end
98          end
99      end
100     // Displaying the Reconstructed Image.
101     diff_image = im2double(I)*255-I_rec;
102     diff_image = diff_image/max(max(diff_image));
103     diff_image = im2uint8(diff_image);
104     I_rec = I_rec/max(max(I_rec));
105     I_rec = im2uint8(I_rec);
106     figure
107     ShowImage(I_rec,'Recovered Image');
108     figure
109     ShowImage(diff_image,'Difference Image')
110     figure
111     imhist(I_rec);
112     figure
113     imhist(diff);
```

# Experiment: 6

# DPCM encoding and decoding of images

**Scilab code Solution 6.1** Exp6

```
1  // Program 6 DPCM encoding and decoding of images
2  //Software version
3  //OS Windows7
4  //Scilab5.4.1
5  //Image Processing Design Toolbox 8.3.1-1
6  //Scilab Image and Video Proccessing toolbox
       0.5.3.1-2
7  clc
8  clear
9  //Function to find number of elements in an image
10 function [N] = numel(X)
11     //X-input image
12     //N- number of elements in image X
13     [m,n]= size(X);
14     N = m*n;
15 endfunction
16 //
       ////////////////////////////////////////////////////////////
```

```
17 //Function to calculate peak signal to noise ratio
18 function [psnr,mse,maxerr] = psnr_mse_maxerr(X,Xapp)
19 //PSNR_MSE_MAXERR Peak signal to noise ratio
20 //X - original Image
21 //Xapp - reconstructed image
22 //psnr - peak signal to noise ratio
23 //mse - mean square error
24 //maxerr -  maximum error
25 X    = double(X);
26 Xapp = double(Xapp);
27 absD = abs(X-Xapp);
28 A    = absD.^2;
29 mse  = sum(A(:))/numel(X);
30 psnr = 10*log10(255*255/mse);
31 maxerr = round(max(absD(:)));
32 endfunction
33 //
   ////////////////////////////////////////////////////////////////

34 a=imread('C:\Users\senthilkumar\Desktop\
   Chandra_Mohan_LAB\Digital_Image_ProcessingLab\
   cameraman.jpeg');
35 a=double(a);
36 [m n]=size(a);
37 pre=0;
38 q=input('enter the quantization value');
39 for i=1:m
40     for j=1:n
41         t1=a(i,j)-pre;
42         tq=round(t1/q);
43         pre=pre+tq*q;
44         b(i,j)=tq;
45     end
46 end
47 repre=0;
48 for i=1:m
49     for j=1:n
50         ret=b(i,j);
```

```
51          inq=ret*q;
52          repre=repre+inq;
53          c(i,j)=repre;
54      end
55  end
56  figure
57  ShowImage(a,'Image Before Quantization')
58  figure
59  ShowImage(b,'Quantized Image')
60  figure
61  ShowImage(c,'Reconstructed Image From Quantized
       Image')
62  psnr = psnr_mse_maxerr(a,c);
63  disp(psnr,'PSNR in dB= ')
64  //RESULT
65  //enter the quantization value 2
66  //PSNR in dB =  51.165559
67
68  //enter the quantization value 8
69  //PSNR in dB =  40.698164
70  //
```

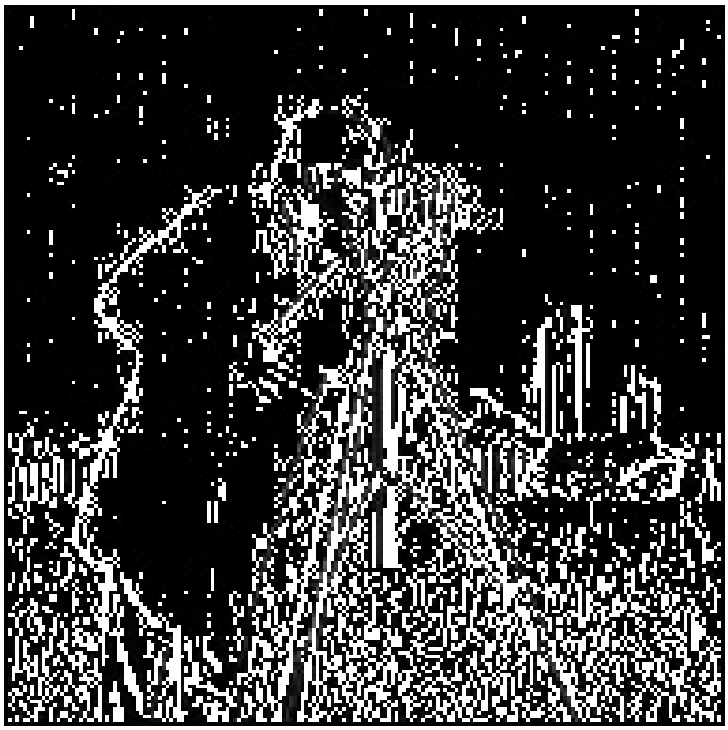


Figure 6.1: Exp6



Figure 6.2: Exp6

# Experiment: 7

# Simple image watermarking algorithms using LSB substitution

**Scilab code Solution 7.1** Exp7

```
1 //Program 7. Simple image watermarking algorithms
    using LSB substitution
2 //Note 1: The imread function in SIVP toolbox read
    the binary image as gray
3 //scale image. During bitset it will create problems
    .
4 //The grayscale image can be converted into binary
    image using the function
5 //gray2bin()
6 //Note 2: The functions bit_set and bit_get are
    written inorder to save the
7 // scilab workspace memory during execution
8 //Software version
9 //OS Windows7
10 //Scilab5.4.1
11 //Image Processing Design Toolbox 8.3.1-1
12 //Scilab Image and Video Proccessing toolbox
```

```
    0.5.3.1 -2
clc
clear
close
//Function to find number of elements in an image
function [N] = numel(X)
    //X-input image
    //N- number of elements in image X
    [m,n]= size(X);
    N = m*n;
endfunction
//Function to calculate peak signal to noise ratio
function [psnr,mse,maxerr] = psnr_mse_maxerr(X,Xapp)
//PSNR_MSE_MAXERR Peak signal to noise ratio
//X - original Image
//Xapp - reconstructed image
//psnr - peak signal to noise ratio
//mse - mean square error
//maxerr -  maximum error
X    = double(X);
Xapp = double(Xapp);
absD = abs(X-Xapp);
A    = absD.^2;
mse  = sum(A(:))/numel(X);
psnr = 10*log10(255*255/mse);
maxerr = round(max(absD(:)));
endfunction
//
   ////////////////////////////////////////////////////////////////

function [A]= gray2bin(B)
    [m,n] = size(B)
    for i = 1:m
        for j = 1:n
            if(B(i,j)>200)
                A(i,j)= 1;
            else
                A(i,j)=0;
```

```
48              end
49
50          end
51
52      end
53  endfunction
54  //
    ////////////////////////////////////////////////////////////////////

55  function [c]= bit_set(c,b)
56      [m,n] = size(c);
57      for i=1:m
58        for j=1:n
59          c(i,j)=bitset(c(i,j),1,b(i,j));
60        end
61      end
62  endfunction
63  //
    ////////////////////////////////////////////////////////////////////

64  function [d] = bit_get(c)
65      [m,n] = size(c);
66      for i=1:m
67          for j=1:n
68              d(i,j)=bitget(c(i,j),1);
69          end
70      end
71  endfunction
72  //
    ////////////////////////////////////////////////////////////////////

73  a = imread('C:\Users\senthilkumar\Desktop\
    Chandra_Mohan_LAB\Digital_Image_ProcessingLab\
    cameraman.jpeg');  // original image
74  b = imread('C:\Users\senthilkumar\Desktop\
    Chandra_Mohan_LAB\Digital_Image_ProcessingLab\wat
    .jpg');          // watermark image
75  b = gray2bin(b);
```

```
76 [m n] = size(a);
77 a = double(a);
78 c = a;
79 c = bit_set(c,b);
80 d = bit_get(c);
81
82 figure
83 ShowImage(a,'Original image');
84 title('Original image');
85 figure
86 ShowImage(b,'watermark image');
87 title('watermark image');
88 figure
89 ShowImage(uint8(c),'watermarked image');
90 title('watermarked image');
91 figure
92 ShowImage(d,'extracted watermark');
93 title('extracted watermark');
94 psnr = psnr_mse_maxerr(a,c);
95 correlation = corr2(b,d);
96 disp(correlation,'correlation between watermark
     image and extracted watermark=')
```




Figure 7.1: Exp7




Figure 7.2: Exp7

# Experiment: 8

# Simple content based image retrieval using various distance metrics

**Scilab code Solution 8.1** Exp8

```
1 //Program 8: Simple content based image retrieval
    using various distance metrics.
2 //Based on Similarity matrix
3 //Using Colormaps of different images
4 //Note 1: Other methods like wavelet based
    decomposition along with Euclidean distance
5 //comparison of sub images can be used for image
    retrieval
6 //Note 2: Principal Component Analysis (PCA) inbuilt
     function is available to
7 //get eigen vectors and eigen values for image
    retrieval
8 //Software version
9 //OS Windows7
10 //Scilab5.4.1
11 //Image Processing Design Toolbox 8.3.1-1
12 //Scilab Image and Video Proccessing toolbox
```

```
    0.5.3.1 -2
clear;
clc;
close;
I1 = imread('C:\Users\senthilkumar\Desktop\
    Chandra_Mohan_LAB\Digital_Image_ProcessingLab\
    Picture1.png');//257x257x3.
I1 = imresize(I1,0.5);
[IndexedImage_I1, ColorMap] = RGB2Ind(I1); //IPD
    toolbox
I = ColorMap;//66049x3
J1 = imread('C:\Users\senthilkumar\Desktop\
    Chandra_Mohan_LAB\Digital_Image_ProcessingLab\
    Picture2.png');//257x257x3.
J1 = imresize(J1,0.5);
[IndexedImage_J1, ColorMap] = RGB2Ind(J1); //IPD
    toolbox
J = ColorMap;//66049x3
//Similarity Matrix Method
[r,c]= size(I);
A = [];
I = double(I);
J = double(J);
for i = 1:r
    for j = 1:c
        M1(i,j) = (I(i,2)*sin(I(i,1))-J(j,2)*sin(J(j
            ,1)))^2;
        M2(i,j) = (I(i,2)*cos(I(i,1))-J(j,2)*cos(J(j
            ,1)))^2;
        M3(i,j) = (I(i,3)-J(i,3))^2;
        M(i,j)= sqrt(M1(i,j)+M2(i,j)+M3(i,j));
        A(i,j) = 1-M(i,j)/sqrt(5);
    end
end
I1_rec = Ind2RGB(IndexedImage_I1,A)
I1_rec = imresize(I1_rec,2);
J1_rec = Ind2RGB(IndexedImage_J1,A)
J1_rec = imresize(J1_rec,2);
```

```
42 figure
43 ShowColorImage(I1,'original first image');
44 figure
45 ShowColorImage(I1_rec,'Reconstructed first image');
46 figure
47 ShowColorImage(J1,'original second image');
48 figure
49 ShowColorImage(J1_rec,'Reconstructed second image');
```



Figure 8.1: Exp8



Figure 8.2: Exp8

# Experiment: 9

# Image segmentation algorithms using Snakes

**Scilab code Solution 9.1** Exp9

```
1 // Program 9.Image segmentation algorithms using
    snakes.
2 //Note: Incomplete.
3 //So many functions are not avilable in Scilab
4 //Image segmentation algorithms using snakes is
    impossible with current
5 //version of scilab and scilab image processing
    atoms.
6 //I tried my best
7 //Software version
8 //OS Windows7
9 //Scilab5.4.1
10 //Image Processing Design Toolbox 8.3.1-1
11 //Scilab Image and Video Proccessing toolbox
    0.5.3.1-2
12 close;
13 clear;
14 clc;
15 J = imread('C:\Users\senthilkumar\Desktop\
```

```
    Chandra_Mohan_LAB\Digital_Image_ProcessingLab\
    binary_image.jpg');
J = rgb2gray(J);
J = imresize(J,[256,256]);
J = double(J);
[h,w] = size(J);
for i = 1:h
    for j= 1:w
        if(J(i,j)>200)
            J(i,j)= 1;
        else
            J(i,j) =0;
        end
    end
end
I = imfilter(J,fspecial('gaussian',[17 17],3));
figure
ShowImage(I,'Snakes')
N=500; // number of snake points
alpha=1;
tstep=1;
N_iter=500;
f=50;
global EDGE_SOBEL;
gradient = EdgeFilter(I,EDGE_SOBEL);
[m,n] = size(gradient);
Ix =  gradient(:,:);
Iy =  gradient(:,:)';
S = -f*(Ix.*Ix + Iy.*Iy);
gradient = EdgeFilter(S,EDGE_SOBEL);
Sx = gradient(:,:);
Sy = gradient(:,:)';
eps =  2.2204e-016;
Smag = sqrt(Sx.^2 + Sy.^2)+eps;
Sx(:) = Sx./Smag;
Sy(:) = Sy./Smag;
D=[-tstep*alpha*ones(N,1) (1+2*tstep*alpha)*ones(N
    ,1) -tstep*alpha*ones(N,1)];
```

```
51 D(2,3)=D(2,3)-tstep*alpha;
52 D($-1,1)=D($-1,1)-tstep*alpha;
53 theta = linspace(0,2*%pi,N);
54 theta = theta(:);
55 x = w/2 + 10 + (h/3)*cos(theta);
56 y = h/2 - 10 + (h/4)*sin(theta);
57 plot(x,y,'r');
```

# Experiment: 10

# Color images manipulations, reading and writing of color images

**Scilab code Solution 10.1** Exp10

```
1 //Program 10.Color images manipulations, reading and
      writing of color images
2 //Software version
3 //OS Windows7
4 //Scilab5.4.1
5 //Image Processing Design Toolbox 8.3.1-1
6 //Scilab Image and Video Proccessing toolbox
     0.5.3.1-2
7 clc
8 clear
9 close
10 //Showing RGB components of a color RGB image.
11 //Splitting the color image (RGB Image) into three
     planes
12   a=imread('C:\Users\senthilkumar\Desktop\
       Chandra_Mohan_LAB\DIP_Scilab_Programs\peppers.
       png'); //this image is 348x512x3 size
```

```
13    figure
14    ar=a(:,:,1);
15    ShowImage(ar,'RED Matrix')
16    figure
17    ag=a(:,:,2);
18      ShowImage(ag,'GREEN Matrix')
19    figure
20    ab=a(:,:,3);
21    ShowImage(ab,'BLUE Matrix')
22  //Reconstruction of original color image from three
      RGB planes
23
24    RGB = imread('C:\Users\senthilkumar\Desktop\
        Chandra_Mohan_LAB\DIP_Scilab_Programs\peppers.
        png');//SIVP toolbox
25    RGB_128 = RGB/2;
26    RGB_128 = round(RGB_128)
27    [X,map] = RGB2Ind(RGB_128);
28    figure
29    ShowImage(X,'Indexed Image',map)
30  //Limiting no of colours to 8 without dithering
31    figure
32    RGB_8 = RGB/7;
33    RGB_8 = round(RGB_8)
34    [X1,map1]=RGB2Ind(RGB);
35    ShowImage(X1,'Without Dither',map1)
36
37  figure
38  ShowColorImage(RGB,'RGB Color Image')
39  YIQ = rgb2ntsc(RGB);
40  figure
41  ShowColorImage(YIQ,'NTSC image YIQ')
42  RGB = ntsc2rgb(YIQ);
43  YCC = rgb2ycbcr(RGB);
44  figure
45  ShowColorImage(YCC,'equivalent HSV image YCbCr')
46  RGB = ycbcr2rgb(YCC);
47  HSV = rgb2hsv(RGB);
```

```
48 figure
49 ShowColorImage(HSV,'equivalent HSV image')
50 RGB = hsv2rgb(HSV);
```

# Experiment: 11

# Color image enhancements

check Appendix AP 3 for dependency:

`imgenh11.sci`

**Scilab code Solution 11.1** Exp11

```
1  //Program 11. Color image enhancements
2  //Software version
3  //OS Windows7
4  //Scilab5.4.1
5  //Image Processing Design Toolbox 8.3.1-1
6  //Scilab Image and Video Proccessing toolbox
       0.5.3.1-2
7  clc
8  clear
9  close
10 a=imread('C:\Users\senthilkumar\Desktop\
       Chandra_Mohan_LAB\Digital_Image_ProcessingLab\
       balloonsnoisy.png');
11 ks=input('enter the size of the kernel 1 for 1 1  3
       for 3 3  ...');
12 exec('C:\Users\senthilkumar\Desktop\
       Chandra_Mohan_LAB\Digital_Image_ProcessingLab\
       imgenh_11.sci')
```

```
13 for i=1:3
14     b(:,:,i)=imgenh11(a(:,:,i),ks);
15 end
16
17 figure
18 ShowColorImage(a,'Noised image(before enhancement)')
     ;
19 title('Noised image(before enhancement)');
20 figure
21 ShowColorImage(uint8(b),'enhancement with mean
     filtering');
22 title('enhancement with mean filtering');
23 //RESULT
24 //enter the size of the kernel 1 for 1 1  3 for 3 3
     ...3
25
26 //NOTE: since the image is large [480   640] it will
     take some time to
27 //show the result
```

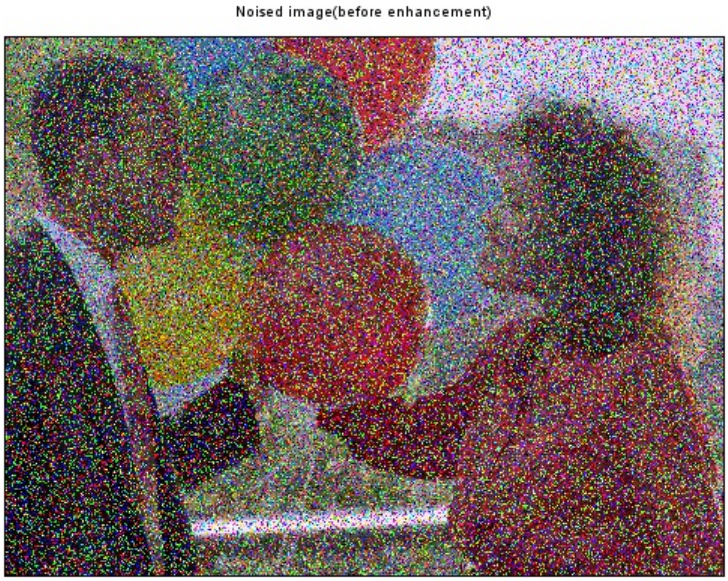



Figure 11.1: Exp11




Figure 11.2: Exp11

# Experiment: 12

# Color image histogram manipulation

check Appendix AP 2 for dependency:

`histbw12.sci`

**Scilab code Solution 12.1** Exp12

```
1  //Program 12 Color image histogram manipulation
2  //Software version
3  //OS Windows7
4  //Scilab5.4.1
5  //Image Processing Design Toolbox 8.3.1-1
6  //Scilab Image and Video Proccessing toolbox
       0.5.3.1-2
7  clc
8  close
9  a=imread('C:\Users\senthilkumar\Desktop\
       Chandra_Mohan_LAB\Digital_Image_ProcessingLab\
       peppers.png');
10 a1=uint8(a);
11 exec('C:\Users\senthilkumar\Desktop\
       Chandra_Mohan_LAB\Digital_Image_ProcessingLab\
       histbw_12.sci')
```

```
12 for i=1:3
13     b(:,:,i)=histbw12(a1(:,:,i));
14 end
15 figure
16 ShowColorImage(a,'original color image');
17 title('original color image');
18 figure
19 ShowColorImage(b,'histogram equalization of color
     image');
20 title('histogram equalization of color image');
21 rgbhist_12(a);
22 //exec('C:\Users\senthilkumar\Desktop\
     Chandra_Mohan_LAB\Digital_Image_ProcessingLab\
     rgbhist_12.sci')
```




Figure 12.1: Exp12




Figure 12.2: Exp12

# Experiment: 13

# LOG Masks implementation for gray and color images

**Scilab code Solution 13.1** Exp13

```
1 //Program 13. LOG Masks implementation for gray and
    color images
2 //Software version
3 //OS Windows7
4 //Scilab5.4.1
5 //Image Processing Design Toolbox 8.3.1-1
6 //Scilab Image and Video Proccessing toolbox
    0.5.3.1-2
7 clc
8 clear
9 close
10 a=imread('C:\Users\senthilkumar\Desktop\
    Chandra_Mohan_LAB\Digital_Image_ProcessingLab\
    cameraman.jpeg');
11 a=double(a);
12 logmask=[0 1 1 2 2 2 1 1 0;1 2 4 5 5 5 4 2 1;1 4 5 3
     0 3 5 4 1;2 5 3 -12 -24 -12 3 5 2;2 5 0 -24 -40
    -24 0 5 2;
13           2 5 3 -12 -24 -12 3 5 2;1 4 5 3 0 3 5 4 1;1
```

```
                2 4 5 5 5 4 2 1;0 1 1 2 2 2 1 1 0];
[m n]=size(a);
[m1 n1]=size(logmask);
b=zeros(m+m1-1,n+n1-1);
m2=floor(m1/2);
n2=floor(n1/2);
b(m2+1:m+m2,n2+1:n+n2)=a;
for i=m2+1:m+m2
    for j=n2+1:n+n2
        c=b(i-m2:i+m2,j-n2:j+n2);
        d=sum(sum(c.*logmask));
        if d>150
            e(i-m2,j-n2)=0;
        else
            e(i-m2,j-n2)=1;
        end

    end
end
title('Camerman image after LOG masked')
imshow(e)//SIVP toolbox
```

# Experiment: 14

# Special effects implementation on grey and color images

check Appendix AP 1 for dependency:

```
rot90f.sci
```

**Scilab code Solution 14.1** Exp14

```
1 //Program 14. Special effects implementation on gray
    and color images
2 //Note: The functions like entropfilt().m are not
    available in scilab
3 //But similar effects can be produced in scilab
    using other methods.
4 //Software version
5 //OS Windows7
6 //Scilab5.4.1
7 //Image Processing Design Toolbox 8.3.1-1
8 //Scilab Image and Video Proccessing toolbox
    0.5.3.1-2
9 clc;
10 clear;
11 close;
```

```
12 exec('C:\Users\senthilkumar\Desktop\
      Chandra_Mohan_LAB\Digital_Image_ProcessingLab\
      rot90_f.sci')
13 img1 = imread('C:\Users\senthilkumar\Desktop\
      Chandra_Mohan_LAB\Digital_Image_ProcessingLab\
      mandrill.jpeg');//colour image
14 img2 = imread('C:\Users\senthilkumar\Desktop\
      Chandra_Mohan_LAB\Digital_Image_ProcessingLab\
      twozebras.jpg');//colour image
15 img3 = imread('C:\Users\senthilkumar\Desktop\
      Chandra_Mohan_LAB\Digital_Image_ProcessingLab\
      cameraman.jpeg');//gray image
16 filter1 = fspecial('sobel');
17 img1_filter = imfilter(img1,filter1);
18 img2_filter = imfilter(img2,filter1);
19 ShowColorImage(img1,'original image 1');
20 figure
21 ShowColorImage(img1_filter,'Special effect in
      Mandrill Image')
22 figure
23 ShowColorImage(img2,'original image 2');
24 figure
25 ShowColorImage(img2_filter,'Special effect in
      twozebras Image')
26 img3_negative = 255-double(img3); //image negative
27 img3_rotate = rot90f(img3,3);
28 //Image contrat adjustment
29 [m,n] = size(img3);
30 for i = 1:m
31     for j = 1:n
32         if img3(i,j)>70 then
33             img3_adjust(i,j) = img3(i,j)+(255-img3(i
                  ,j));
34         else
35             img3_adjust(i,j) = img3(i,j);
36         end
37
38     end
```

```
39 end
40 figure
41 ShowImage(img3,'Cameraman original Image');
42 figure
43 ShowImage(img3_negative,'Cameraman Negative Image')
44 figure
45 ShowImage(img3_rotate,'270 degree rotation of
      camerman image')
46 figure
47 ShowImage(img3_adjust,'Cameraman Image Contrast
      Adjustment')
```

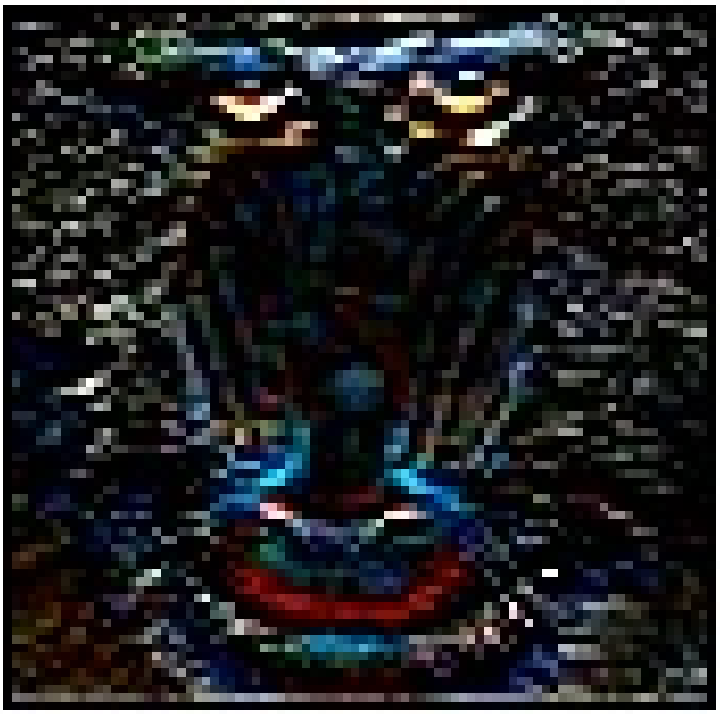


Figure 14.1: Exp14



Figure 14.2: Exp14

# Experiment: 15

# Simple video reading and writing .avi formats and manipulation of video frames

**Scilab code Solution 15.1** Exp15a

```
1 //Program 15. Simple video reading and writing .avi
    formats and manipulation of video frames.
2 //Note 1: Install xvid codec for read and write
    video files from
3 //http://www.xvid.org/Downloads.15.0.html
4 //Note 2: very large can not be read by scilab
5 //Note 3: shuttle.avi is a large file more 100
    frames. use shuttlenew.avi file
6 //for video processing applications
7 //Using SIVP Atom
8 //Software version
9 //OS Windows7
10 //Scilab5.4.1
11 //Image Processing Design Toolbox 8.3.1-1
12 //Scilab Image and Video Proccessing toolbox
    0.5.3.1-2
13 clear;
```

```
14 clc;
15 close;
16 //n = aviopen(SCI+'/contrib/sivp/images/video.avi');
17 n = aviopen('C:\Users\senthilkumar\Desktop\
     Chandra_Mohan_LAB\Digital_Image_ProcessingLab\red
     -car-video.avi');
18 im = avireadframe(n); //get a frame
19 imshow(im);
20 avilistopened()
21 aviclose(n);
```

**Scilab code Solution 15.2** Exp15b

```
1 //Program 15. Simple video reading and writing .avi
     formats and manipulation of video frames.
2 //Note 1: Install xvid codec for read and write
     video files from
3 //http://www.xvid.org/Downloads.15.0.html
4 //Note 2: very large can not be read by scilab
5 //Note 3: shuttle.avi is a large file more 100
     frames. use red-car-video.avi file
6 //for video processing applications
7 //Using Image Processing Design Atom (IPD)
8 //Software version
9 //OS Windows7
10 //Scilab5.4.1
11 //Image Processing Design Toolbox 8.3.1-1
12 //Scilab Image and Video Proccessing toolbox
     0.5.3.1-2
13 clear;
14 clc;
15 close;
16 VideoPath = 'C:\Users\senthilkumar\Desktop\
     Chandra_Mohan_LAB\Digital_Image_ProcessingLab\red
     -car-video.avi';
```

```
17
18 VideoInfo = GetVideoStruct('C:\Users\senthilkumar\
      Desktop\Chandra_Mohan_LAB\
      Digital_Image_ProcessingLab\red-car-video.avi');
19
20 VideoFilePointer = OpenVideoFile('C:\Users\
      senthilkumar\Desktop\Chandra_Mohan_LAB\
      Digital_Image_ProcessingLab\red-car-video.avi');
21
22 figure();
23
24 for n = 1 : VideoInfo.NumberOfFrames
25
26  RGB = ReadImage(VideoFilePointer);
27
28  ShowColorImage(RGB, VideoPath);
29
30 end;
31
32 CloseVideoFile(VideoFilePointer);
```

# Appendix

**Scilab code AP 1**

```
function [B] = rot90f(A,k)
[%nargout,%nargin] = argn(0)
//ROT90  Rotate matrix 90 degrees.
//   ROT90(A) is the 90 degree counterclockwise rotation of matrix A.
//   ROT90(A,K) is the K*90 degree rotation of A, K = +-1,+-2,...
[m,n] = size(A);
if %nargin==1 then
  k = 1;
else
  k = k-fix(k/4).*4;
  if(k<0) then
    k = k+4;
  end
end
if k == 1
             A = A.';
             B = A(n:-1:1,:);
elseif k == 2
             B = A(m:-1:1,n:-1:1);
elseif k == 3
             B = A(m:-1:1,:);
             B = B.';
else
             B = A;
end
```

```
endfunction
```

Rotate Image 90 degree

**Scilab code AP 12**

```
function [hea,b]=histbw12(a)
    //a=imread('C:\Users\senthilkumar\Desktop\
       Chandra_Mohan_LAB\Digital_Image_ProcessingLab
       \tire.jpeg')
    //a- original image
    //b- histogram
    //hea- histogram equalized image
    [m n]=size(a);
    for i=1:256
        b(i)=length(find(a==(i-1)));
    end
    pbb=b/(m*n);
    pb(1)=pbb(1);
    for i=2:256
        pb(i)=pb(i-1)+pbb(i);
    end

    s=pb*255;
    sb=uint8(round(s));
    index =0;
    for i=1:m
        for j=1:n
            index = double(a(i,j))+1;//convert it to
                double
            //otherwise index = 255+1 =0
            hea(i,j)= sb(index);//histogram
               equalization
        end
    end
endfunction
//note:
//First run this function
//type the following commands in scilab console
   window
```

```
30 //a=imread('C:\Users\senthilkumar\Desktop\
     Chandra_Mohan_LAB\Digital_Image_ProcessingLab\
     tire.jpeg')
31 //[hea,b] = histbw_12(a);
32 //figure,
33 //ShowImage(a,'Original Image')//IPD toolbox
34 //title('Original Image')
35 //figure
36 //plot2d3('gnn',[1:256],b)
37 //title('Histogram of the Image')
38 //figure
39 //ShowImage(hea,'Image after Histogram equalization
     ')//IPD toolbox
40 //title('Image after Histogram equalization')
```

Histogram of Gray images

**Scilab code AP 13** `function [out] =imgenh11(a,ks)`

```
2     [m n]=size(a);
3     a1=zeros(m+ks-1,n+ks-1);
4     [m1 n1]=size(a1);
5     x=floor(ks/2);
6     a1(1+x:m1-x,1+x:n1-x)=a;
7     out=[];
8
9     for i=1+x:m1-x
10        for j=1+x:n1-x
11            t=a1(i-x:i+x,j-x:j+x);
12            med=median(t(:));
13            out(i-x,j-x)=med;
14        end
15    end
16 endfunction
```

Image Enhancement

**Scilab code AP 14** `function [result] = izigzag5(data)`

```
2 //inverse ZigZag scanning of input data
```

```
3   N= sqrt(size(data,1));
4   z = 1;
5   count = 0;
6   row = 1;
7   col = 0;
8   for (x = 2:2*N),
9       if (x <= N+1),
10          y = x + 1;
11          if(modulo(x,2) == 0)
12            col = col + 1;
13          else
14            row = row + 1;
15          end
16      else
17          y = N+1;
18          if (modulo(x,2) == 0)
19             row = row - 1;
20             col = col + 2;
21          else
22             row = row + 2;
23             col = col - 1;
24          end
25      end
26
27      while((row < y)&(col < y)&(row > 0)&(col > 0))
28          result(row,col) = data(z);
29          z = z + 1;
30
31          if(modulo(x,2) == 0)
32            row = row - 1;
33            col = col + 1;
34          else
35            row = row + 1;
36            col = col - 1;
37          end
38      end
39  end
40  endfunction
```

**Scilab code AP 15**

```
function [result] = zigzag5(data)
// ZigZag scanning of input data
N= size(data,1);
z = 1;
count = 0;
row = 1;
col = 0;

for (x = 2:2*N),
     if (x <= N+1)
         y = x + 1;
         if(modulo(x,2) == 0)
            col = col + 1;
         else
             row = row + 1;
         end
     else
         y = N+1;
         if(modulo(x,2) == 0)
            row = row - 1;
            col = col + 2;
         else
            row = row + 2;
            col = col - 1;
         end
     end

     while((row < y)&(col < y)&(row > 0)&(col > 0))
         result(z) = data(row,col);
         z = z + 1;
         if(modulo(x,2) == 0)
            row = row - 1;
            col = col + 1;
         else
            row = row + 1;
```

```
36                col = col - 1;
37            end
38        end
39    end
40    endfunction
```

Zig Zag Scanning of Pixels

**Scilab code AP 16**

```
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(
               i,j)))<0.0001))
18             a2(i,j)=0;
19         elseif(abs(real(a2(i,j)))<0.0001)
20             a2(i,j)= 0+%i*imag(a2(i,j));
21         elseif(abs(imag(a2(i,j)))<0.0001)
22             a2(i,j)= real(a2(i,j))+0;
23         end
24     end
25 end
```

2D Fast Fourier Trasnform

**Scilab code AP 17**

```
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
```

Inverse 2D Fast Fourier Transform