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

February 24, 2021

[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

Histogram display and histogram equalization

Scilab code Solution 1.1 Exp1

```
//Program 1 Histogram display and histogram equalization
//Software version
//OS Windows7
//Scilab5.4.1
//Image Processing Design Toolbox 8.3.1-1
//Scilab Image and Video Proccessing toolbox 0.5.3.1-2
clc;
clear;
close;
//a=imread('C:\Users\senthilkumar\Desktop\Chandra_Mohan_LAB\Digital_Image_ProcessingLab\tire.tif');//Image Path
a=imread('C:\Users\senthilkumar\Desktop\Chandra_Mohan_LAB\Digital_Image_ProcessingLab\tire.jpeg')
[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
figure,
ShowImage(a,'Original Image')//IPD toolbox
title('Original Image')
figure
plot2d3('gnn',[1:256],b)
title('Histogram of the Image')
figure
ShowImage(hea,'Image after Histogram equalization')//IPD toolbox
title('Image after Histogram equalization')
```

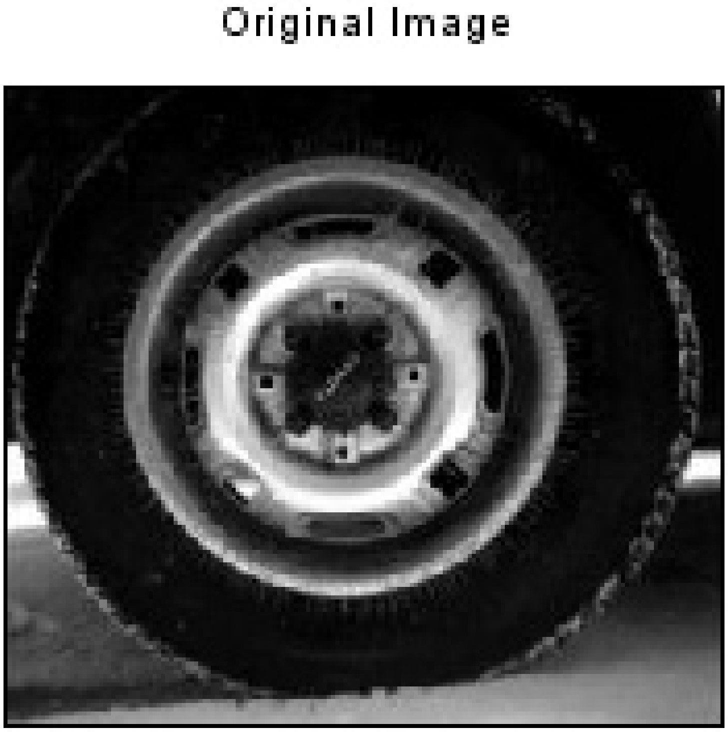

Figure 1.1: Exp1

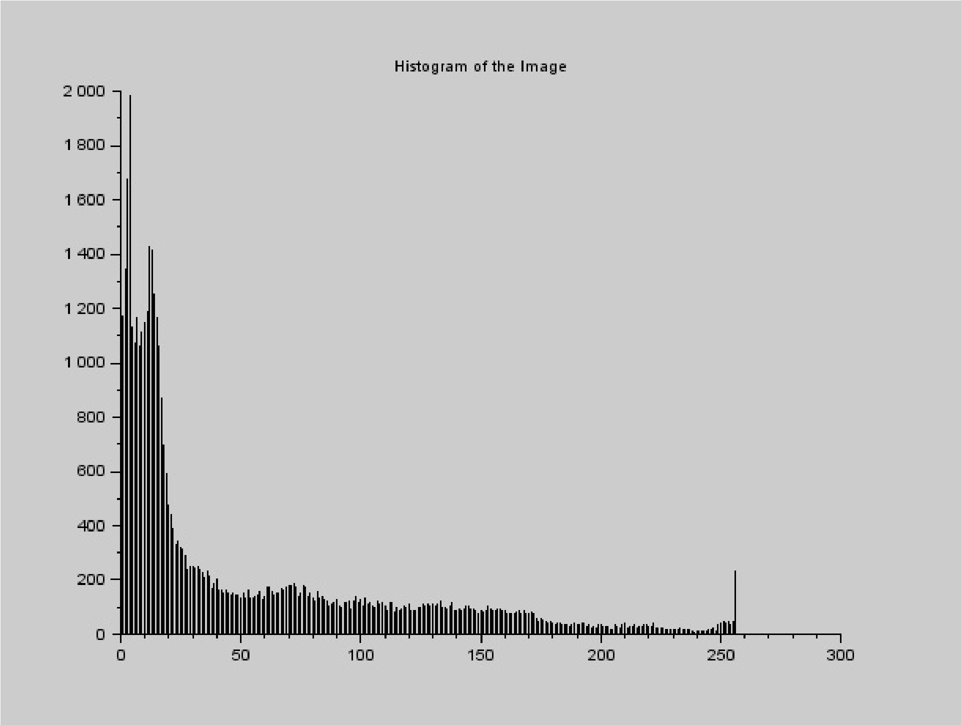

Figure 1.2: Exp1

Experiment: 2

Kernel processing on images leading to image enhancement

Scilab code Solution 2.1 Exp2

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

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
```

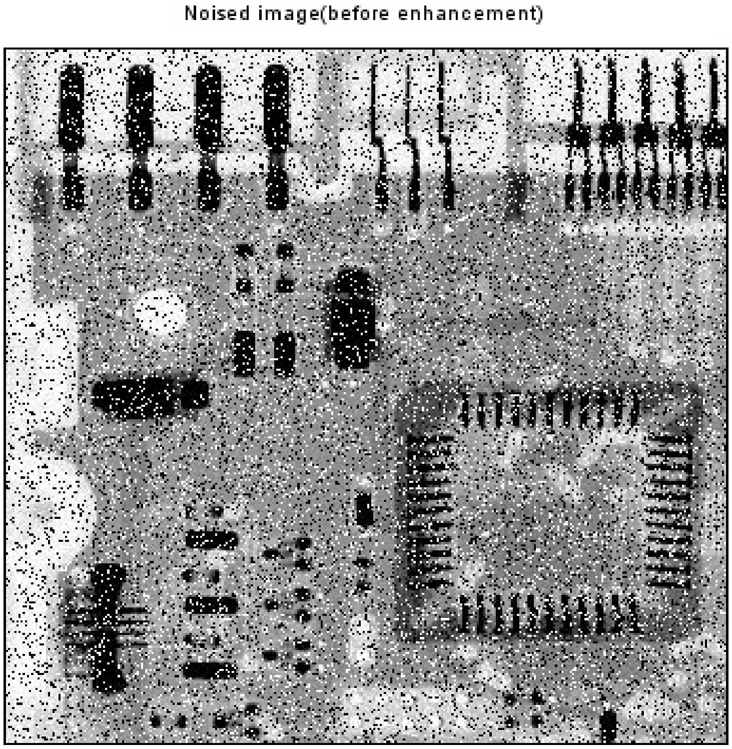

Figure 2.1: Exp2

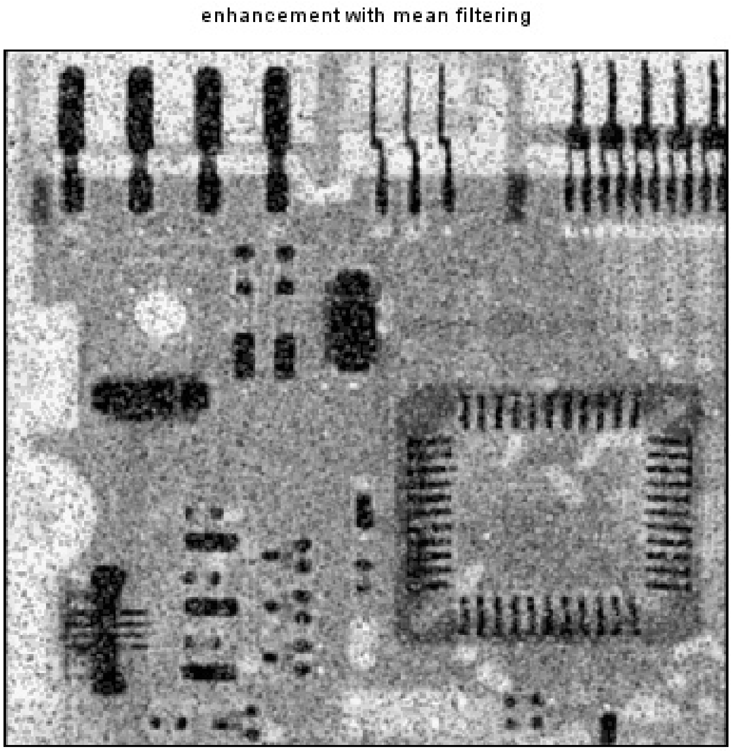

Figure 2.2: Exp2

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

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

```
//OS Windows7
//Scilab5.4.1
//Image Processing Design Toolbox 8.3.1-1
//Scilab Image and Video Proccessing toolbox 0.5.3.1-2
clc;
close;
clear;
exec('C:\Users\senthilkumar\Desktop\Chandra_Mohan_LAB\Digital_Image_ProcessingLab\fft2d.sce')
exec('C:\Users\senthilkumar\Desktop\Chandra_Mohan_LAB\Digital_Image_ProcessingLab\ifft2d.sce')
im1 = imread('C:\Users\senthilkumar\Desktop\Chandra_Mohan_LAB\Digital_Image_ProcessingLab\balloonsnoisy.png');//colour noise image
im = rgb2gray(im1);//gray noise image
fc = 100; //cut off frequency -more features choose high cutoff frequency
n = 1;//filter order =1
[co,ro]= size(im);
cx = round(co/2); //centre of the image
cy = round(ro/2);
IM = fft2d(double(im));
imf = fftshift(IM);
H = zeros(co,ro);
for i = 1:co
    for j = 1:ro
        d = (i-cx).^2+(j-cy).^2;
        H(i,j) = 1/(1+((d/fc/fc).^(2*n))); //Low Pass Butterworth First Order filter
    end
end
out_im = imf.*H;
out = abs(ifft2d(out_im));
out = uint8(out);
figure
```

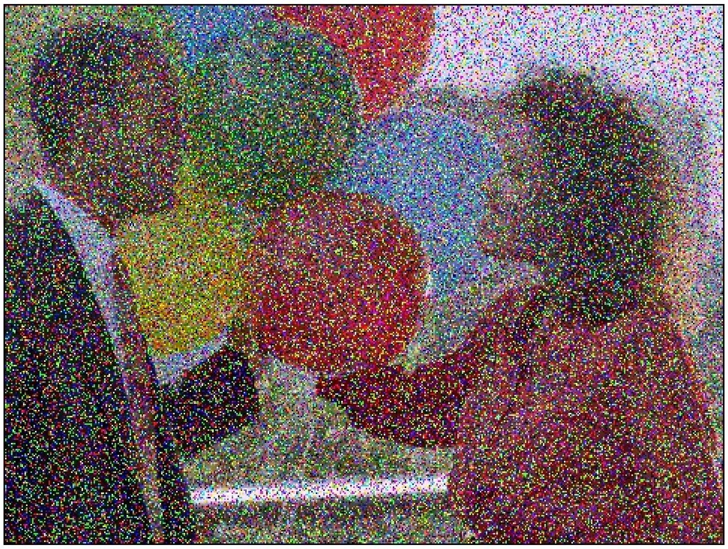

Figure 3.1: Exp3

```
ShowColorImage(im1,'Colour Noisy Image')
figure
ShowImage(im,'Gray Noise Image')
figure
ShowImage(H,'Low Pass Filter Frequency Response')
figure
ShowImage(out,'Filtered Image')
```

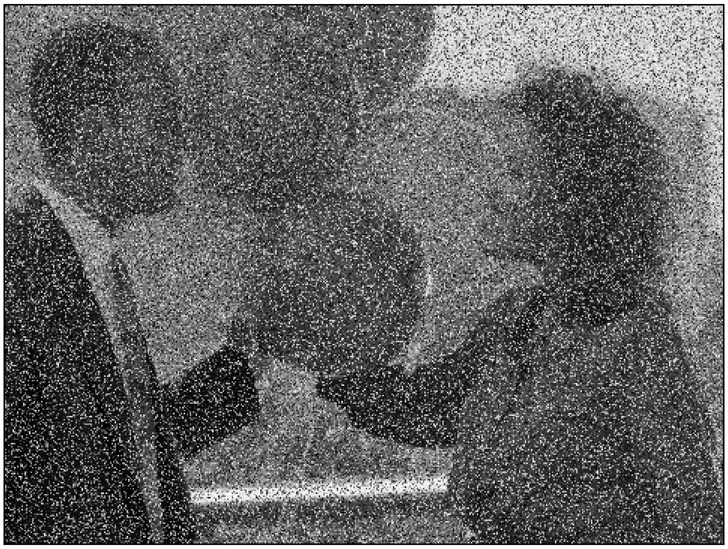

Figure 3.2: Exp3

Experiment: 4

Implementation of Airthmetic Coding for images

Scilab code Solution 4.1 Exp4

```
// Program 4. Implementation of arithmetic coding for images
//Note 1: In order to run this program download Huffman toolbox from
//scilab atoms
//Note 2: The Huffman atom is used to encode images of small size only
//Software version
//OS Windows7
//Scilab5.4.1
//Image Processing Design Toolbox 8.3.1-1
//Scilab Image and Video Proccessing toolbox 0.5.3.1-2
clear;
clc;
close;
//A=testmatrix('frk',10)+1;
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

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

```
exec('C:\Users\senthilkumar\Desktop\Chandra_Mohan_LAB\Digital_Image_ProcessingLab\izigzag_5.sci')
I = imread('C:\Users\senthilkumar\Desktop\Chandra_Mohan_LAB\Digital_Image_ProcessingLab\cameraman.jpeg'); //256x256 image
I = imresize(I,0.25);//reduced to 64x64 image [in order to reduce the computation time]
[m,n]=size(I);// Finding the dimensions of the image file.
I=double(I);
q= [16 11 10 16 24 40 51 61;
    12 12 14 19 26 58 60 55;
    14 13 16 24 40 57 69 56;
    14 17 22 29 51 87 80 62;
    18 22 37 56 68 109 103 77;
    24 35 55 64 81 104 113 92;
    49 64 78 87 103 121 120 101;
    72 92 95 98 112 100 103 99];
N=8;                        // Block size for which DCT is Computed.
M=8;
I_Trsfrm.block=zeros(N,M);  // Initialising the DCT Coefficients Structure Matrix "I_Trsfrm" with the required dimensions.
for a=1:m/N
    for b=1:n/M
        for k=1:N
            for l=1:M
                Mean_Sum=0;
                //2D−Discrete Cosine Transform ///////
                for i=1:N
                    for j=1:M
                        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));
                    end
                end
                ///////////////
                if k==1
                    Mean_Sum = Mean_Sum*sqrt(1/N);
                else
                    Mean_Sum = Mean_Sum*sqrt(2/N);
                end
                if l==1
                    Mean_Sum = Mean_Sum*sqrt(1/M);
                else
                    Mean_Sum = Mean_Sum*sqrt(2/M);
                end
                I_Trsfrm(a,b).block(k,l)= Mean_Sum;
            end
        end
        // Normalizing the DCT Matrix and Quantizing the resulting values.
        I_Trsfrm(a,b).block=round(I_Trsfrm(a,b).block./q);
    end
end
I_zigzag.block = zeros(N,M);
for a= 1:m/N
    for b = 1:n/M
        I_zigzag(a,b).block = zigzag_5(I_Trsfrm(a,b).block);
    end
end
I_rec_Trnsfm.block = zeros(N,M);
for a= 1:m/N
    for b = 1:n/M
        I_rec_Trnsfm(a,b).block = izigzag_5(I_zigzag(a,b).block);
    end
end
// 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));
                I_rec((a-1)*N+i,(b-1)*M+j)= Mean_Sum;
            end
        end
    end
end
// Displaying the Reconstructed Image.
diff_image = im2double(I)*255-I_rec;
diff_image = diff_image/max(max(diff_image));
diff_image = im2uint8(diff_image);
I_rec = I_rec/max(max(I_rec));
I_rec = im2uint8(I_rec);
figure
ShowImage(I_rec,'Recovered Image');
figure
ShowImage(diff_image,'Difference Image')
figure
imhist(I_rec);
figure
imhist(diff);
```

Experiment: 6

DPCM encoding and decoding of images

Scilab code Solution 6.1 Exp6

```
// Program 6 DPCM encoding and decoding of images
//Software version
//OS Windows7
//Scilab5.4.1
//Image Processing Design Toolbox 8.3.1−1
//Scilab Image and Video Proccessing toolbox
   0.5.3.1−2
clc
clear
//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
//////////////////////////////////////////////////////////////////////

a=imread('C:\Users\senthilkumar\Desktop\Chandra_Mohan_LAB\Digital_Image_ProcessingLab\cameraman.jpeg');
a=double(a);
[m n]=size(a);
pre=0;
q=input('enter the quantization value');
for i=1:m
    for j=1:n
        t1=a(i,j)-pre;
        tq=round(t1/q);
        pre=pre+tq*q;
        b(i,j)=tq;
    end
end
repre=0;
for i=1:m
    for j=1:n
        ret=b(i,j);
        inq=ret*q;
        repre=repre+inq;
        c(i,j)=repre;
    end
end
figure
ShowImage(a,'Image Before Quantization')
figure
ShowImage(b,'Quantized Image')
figure
ShowImage(c,'Reconstructed Image From Quantized Image')
psnr = psnr_mse_maxerr(a,c);
disp(psnr,'PSNR in dB= ')
//RESULT
//enter the quantization value 2
//PSNR in dB =  51.165559

//enter the quantization value 8
//PSNR in dB =  40.698164
//
```

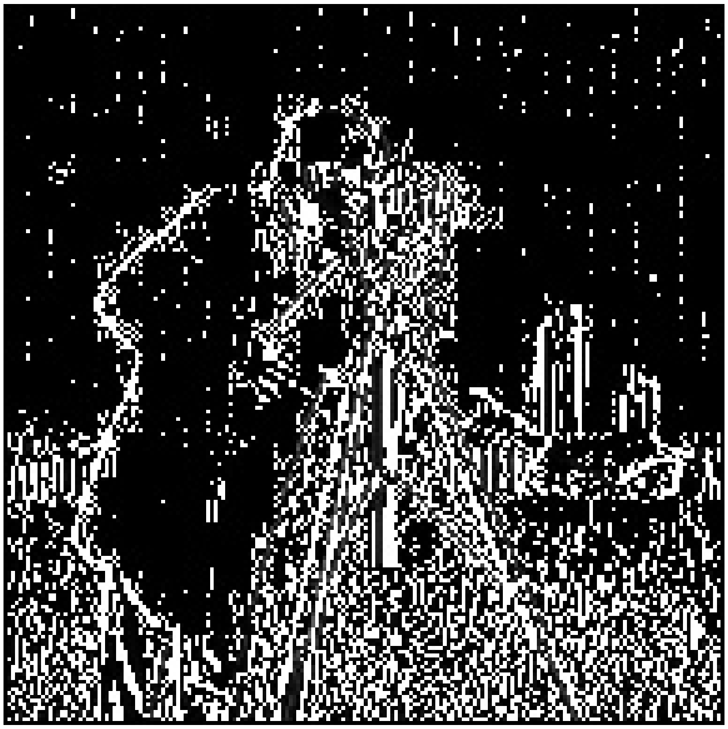

Figure 6.1: Exp6

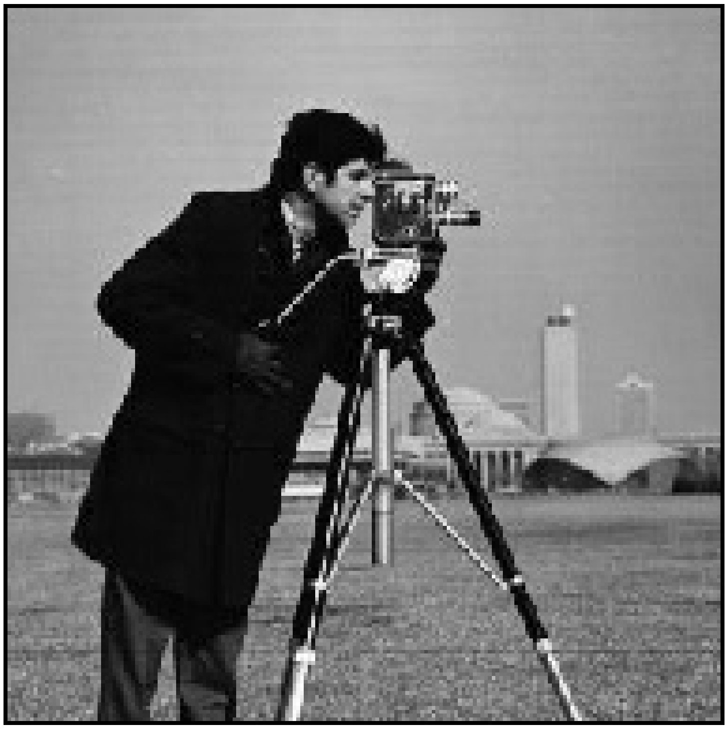

Figure 6.2: Exp6

Experiment: 7

Simple image watermarking algorithms using LSB substitution

Scilab code Solution 7.1 Exp7

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

        end

    end
endfunction
//
   //////////////////////////////////////////////////////////////////

function [c]= bit_set(c,b)
    [m,n] = size(c);
    for i=1:m
      for j=1:n
        c(i,j)=bitset(c(i,j),1,b(i,j));
      end
    end
endfunction
//
   //////////////////////////////////////////////////////////////////

function [d] = bit_get(c)
    [m,n] = size(c);
    for i=1:m
        for j=1:n
            d(i,j)=bitget(c(i,j),1);
        end
    end
endfunction
//
   //////////////////////////////////////////////////////////////////

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

```
[m n] = size(a);
a = double(a);
c = a;
c = bit_set(c,b);
d = bit_get(c);

figure
ShowImage(a,'Original image');
title('Original image');
figure
ShowImage(b,'watermark image');
title('watermark image');
figure
ShowImage(uint8(c),'watermarked image');
title('watermarked image');
figure
ShowImage(d,'extracted watermark');
title('extracted watermark');
psnr = psnr_mse_maxerr(a,c);
correlation = corr2(b,d);
disp(correlation,'correlation between watermark image and extracted watermark=')
```

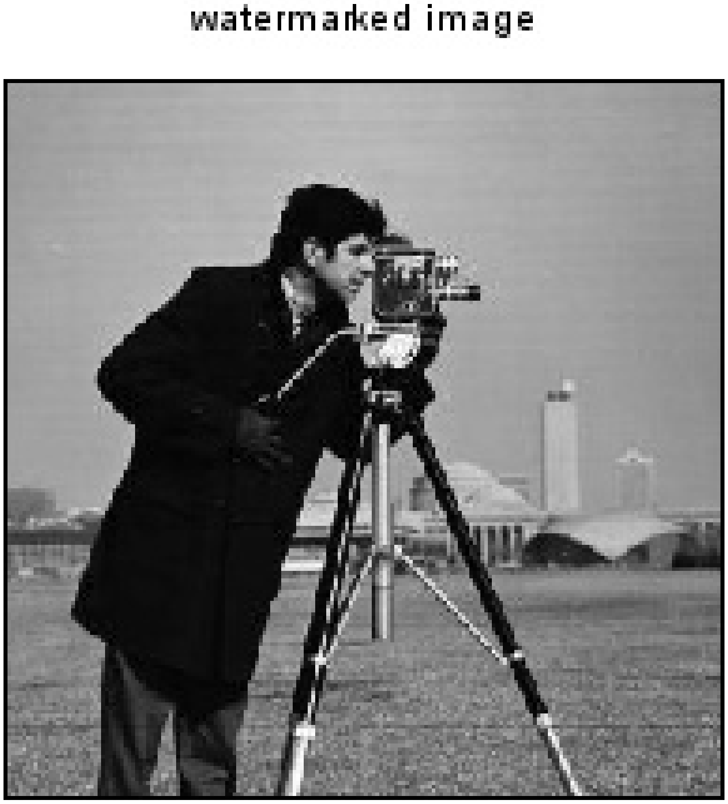

Figure 7.1: Exp7

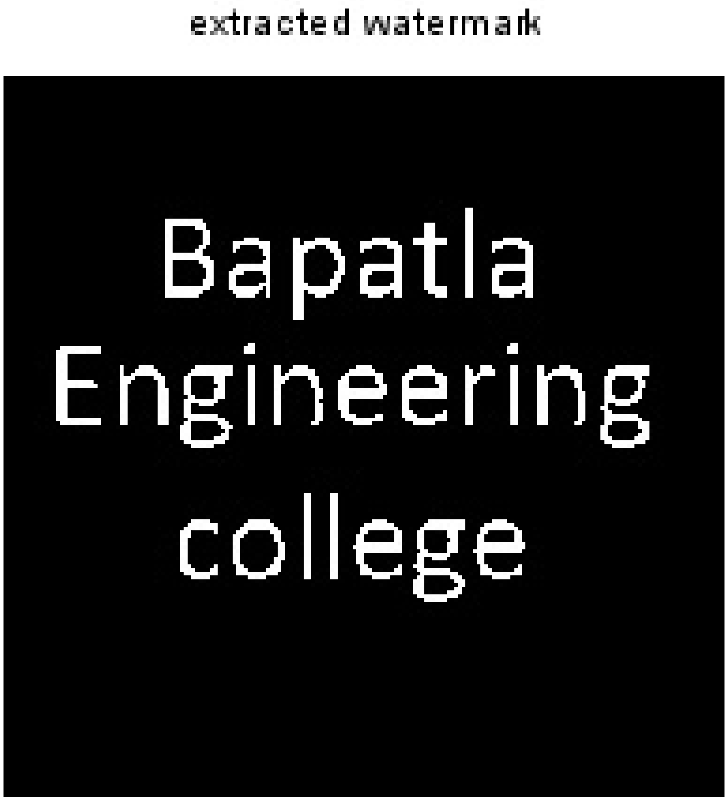

Figure 7.2: Exp7

Experiment: 8

Simple content based image retrieval using various distance metrics

Scilab code Solution 8.1 Exp8

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

```
figure
ShowColorImage(I1,'original first image');
figure
ShowColorImage(I1_rec,'Reconstructed first image');
figure
ShowColorImage(J1,'original second image');
figure
ShowColorImage(J1_rec,'Reconstructed second image');
```

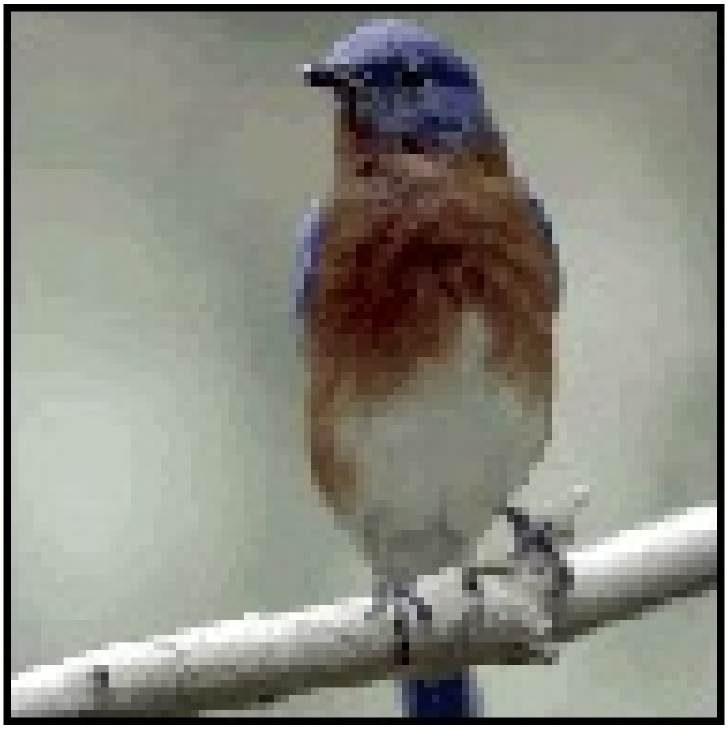

Figure 8.1: Exp8

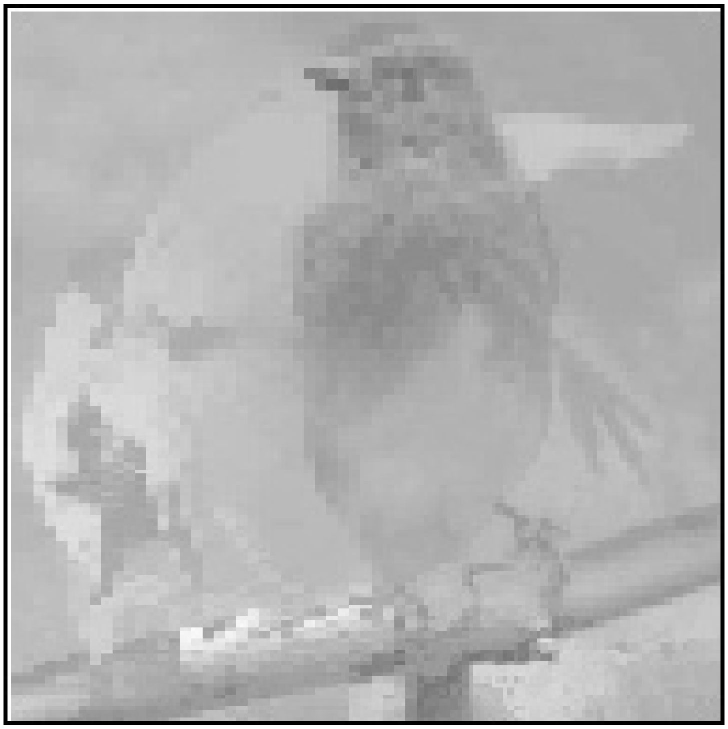

Figure 8.2: Exp8

Experiment: 9

Image segmentation algorithms using Snakes

Scilab code Solution 9.1 Exp9

```
// Program 9.Image segmentation algorithms using
    snakes.
//Note: Incomplete.
//So many functions are not avilable in Scilab
//Image segmentation algorithms using snakes is
    impossible with current
//version of scilab and scilab image processing
    atoms.
//I tried my best
//Software version
//OS Windows7
//Scilab5.4.1
//Image Processing Design Toolbox 8.3.1-1
//Scilab Image and Video Proccessing toolbox
    0.5.3.1-2
close;
clear;
clc;
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)];
```

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

Experiment: 10

Color images manipulations, reading and writing of color images

Scilab code Solution 10.1 Exp10

```
//Program 10.Color images manipulations, reading and writing of color images
//Software version
//OS Windows7
//Scilab5.4.1
//Image Processing Design Toolbox 8.3.1-1
//Scilab Image and Video Proccessing toolbox 0.5.3.1-2
clc
clear
close
//Showing RGB components of a color RGB image.
//Splitting the color image (RGB Image) into three planes
  a=imread('C:\Users\senthilkumar\Desktop\Chandra_Mohan_LAB\DIP_Scilab_Programs\peppers.png'); //this image is 348x512x3 size
  figure
  ar=a(:,:,1);
  ShowImage(ar,'RED Matrix')
  figure
  ag=a(:,:,2);
    ShowImage(ag,'GREEN Matrix')
  figure
  ab=a(:,:,3);
  ShowImage(ab,'BLUE Matrix')
//Reconstruction of original color image from three RGB planes

  RGB = imread('C:\Users\senthilkumar\Desktop\Chandra_Mohan_LAB\DIP_Scilab_Programs\peppers.png');//SIVP toolbox
  RGB_128 = RGB/2;
  RGB_128 = round(RGB_128)
  [X,map] = RGB2Ind(RGB_128);
  figure
  ShowImage(X,'Indexed Image',map)
//Limiting no of colours to 8 without dithering
  figure
  RGB_8 = RGB/7;
  RGB_8 = round(RGB_8)
  [X1,map1]=RGB2Ind(RGB);
  ShowImage(X1,'Without Dither',map1)

figure
ShowColorImage(RGB,'RGB Color Image')
YIQ = rgb2ntsc(RGB);
figure
ShowColorImage(YIQ,'NTSC image YIQ')
RGB = ntsc2rgb(YIQ);
YCC = rgb2ycbcr(RGB);
figure
ShowColorImage(YCC,'equivalent HSV image YCbCr')
RGB = ycbcr2rgb(YCC);
HSV = rgb2hsv(RGB);
```

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

Experiment: 11

Color image enhancements

check Appendix AP 3 for dependency:

`imgenh11.sci`

Scilab code Solution 11.1 Exp11

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

```
for i=1:3
    b(:,:,i)=imgenh11(a(:,:,i),ks);
end

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

//NOTE: since the image is large [480   640] it will take some time to
//show the result
```

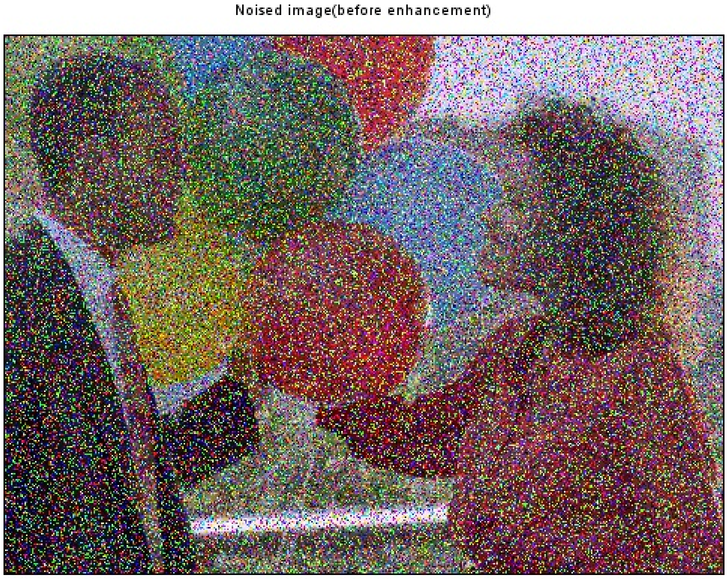

Figure 11.1: Exp11

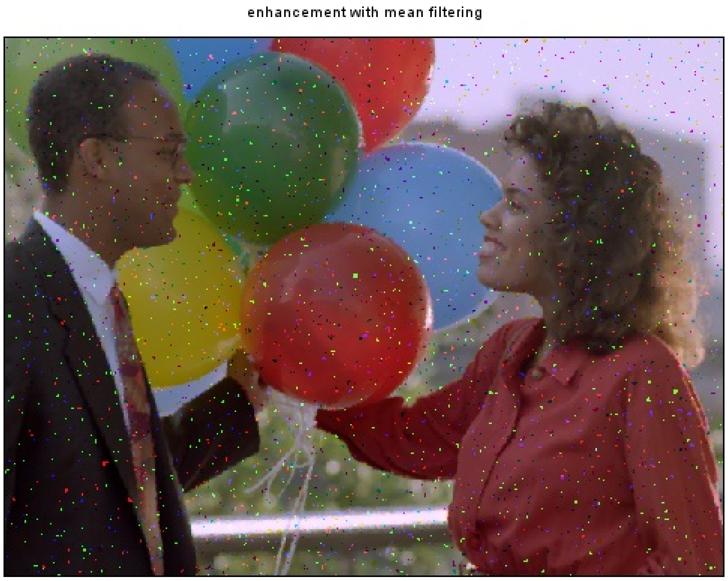

Figure 11.2: Exp11

Experiment: 12

Color image histogram manipulation

check Appendix AP 2 for dependency:

```
histbw12.sci
```

Scilab code Solution 12.1 Exp12

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

```
for i=1:3
    b(:,:,i)=histbw12(a1(:,:,i));
end
figure
ShowColorImage(a,'original color image');
title('original color image');
figure
ShowColorImage(b,'histogram equalization of color image');
title('histogram equalization of color image');
rgbhist_12(a);
//exec('C:\Users\senthilkumar\Desktop\Chandra_Mohan_LAB\Digital_Image_ProcessingLab\rgbhist_12.sci')
```

Figure 12.1: Exp12

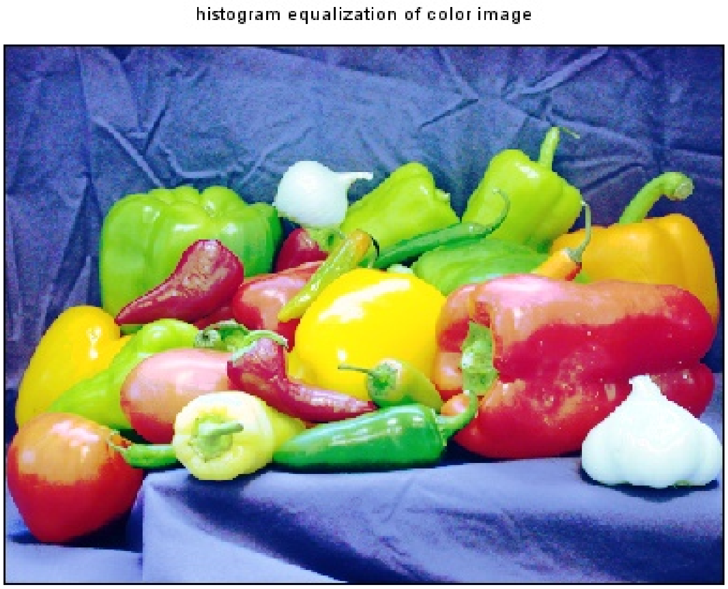

Figure 12.2: Exp12

Experiment: 13

LOG Masks implementation for gray and color images

Scilab code Solution 13.1 Exp13

```
//Program 13. LOG Masks implementation for gray and color images
//Software version
//OS Windows7
//Scilab5.4.1
//Image Processing Design Toolbox 8.3.1-1
//Scilab Image and Video Proccessing toolbox 0.5.3.1-2
clc
clear
close
a=imread('C:\Users\senthilkumar\Desktop\Chandra_Mohan_LAB\Digital_Image_ProcessingLab\cameraman.jpeg');
a=double(a);
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;
        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

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

```
exec('C:\Users\senthilkumar\Desktop\Chandra_Mohan_LAB\Digital_Image_ProcessingLab\rot90_f.sci')
img1 = imread('C:\Users\senthilkumar\Desktop\Chandra_Mohan_LAB\Digital_Image_ProcessingLab\mandrill.jpeg');//colour image
img2 = imread('C:\Users\senthilkumar\Desktop\Chandra_Mohan_LAB\Digital_Image_ProcessingLab\twozebras.jpg');//colour image
img3 = imread('C:\Users\senthilkumar\Desktop\Chandra_Mohan_LAB\Digital_Image_ProcessingLab\cameraman.jpeg');//gray image
filter1 = fspecial('sobel');
img1_filter = imfilter(img1,filter1);
img2_filter = imfilter(img2,filter1);
ShowColorImage(img1,'original image 1');
figure
ShowColorImage(img1_filter,'Special effect in Mandrill Image')
figure
ShowColorImage(img2,'original image 2');
figure
ShowColorImage(img2_filter,'Special effect in twozebras Image')
img3_negative = 255-double(img3); //image negative
img3_rotate = rot90f(img3,3);
//Image contrat adjustment
[m,n] = size(img3);
for i = 1:m
    for j = 1:n
        if img3(i,j)>70 then
            img3_adjust(i,j) = img3(i,j)+(255-img3(i,j));
        else
            img3_adjust(i,j) = img3(i,j);
        end

    end
```

```
end
figure
ShowImage(img3,'Cameraman original Image');
figure
ShowImage(img3_negative,'Cameraman Negative Image')
figure
ShowImage(img3_rotate,'270 degree rotation of camerman image')
figure
ShowImage(img3_adjust,'Cameraman Image Contrast Adjustment')
```

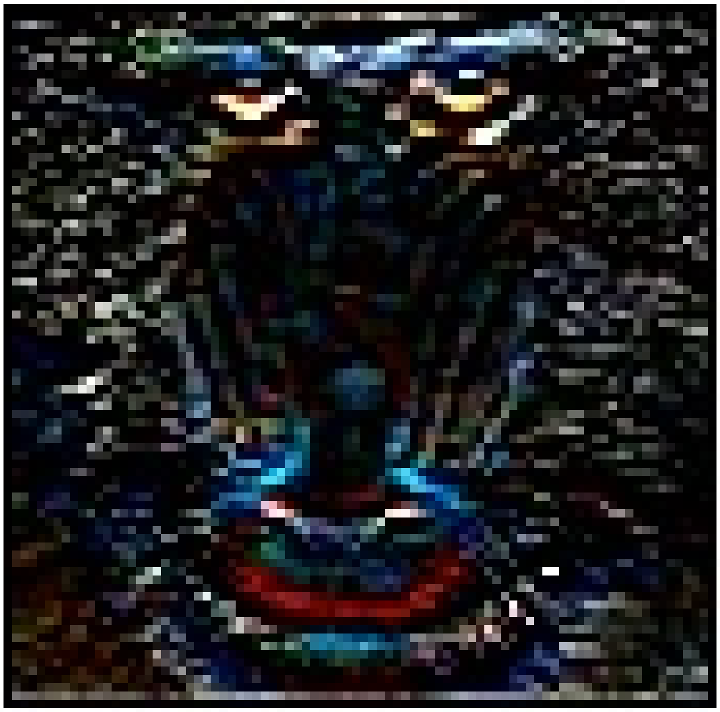

Figure 14.1: Exp14

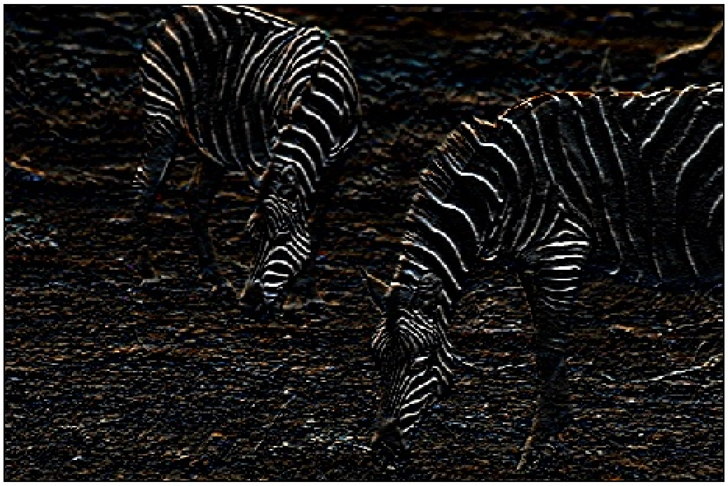

Figure 14.2: Exp14

Experiment: 15

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

Scilab code Solution 15.1 Exp15a

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

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

Scilab code Solution 15.2 Exp15b

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

```

VideoInfo = GetVideoStruct('C:\Users\senthilkumar\
    Desktop\Chandra_Mohan_LAB\
    Digital_Image_ProcessingLab\red-car-video.avi');

VideoFilePointer = OpenVideoFile('C:\Users\
    senthilkumar\Desktop\Chandra_Mohan_LAB\
    Digital_Image_ProcessingLab\red-car-video.avi');

figure();

for n = 1 : VideoInfo.NumberOfFrames

 RGB = ReadImage(VideoFilePointer);

 ShowColorImage(RGB, VideoPath);

end;

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
```

```
//a=imread('C:\Users\senthilkumar\Desktop\Chandra_Mohan_LAB\Digital_Image_ProcessingLab\tire.jpeg')
//[hea,b] = histbw_12(a);
//figure,
//ShowImage(a,'Original Image')//IPD toolbox
//title('Original Image')
//figure
//plot2d3('gnn',[1:256],b)
//title('Histogram of the Image')
//figure
//ShowImage(hea,'Image after Histogram equalization')//IPD toolbox
//title('Image after Histogram equalization')
```

Histogram of Gray images

Scilab code AP 13

```
function [out] =imgenh11(a,ks)
    [m n]=size(a);
    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;
    out=[];

    for i=1+x:m1-x
        for j=1+x:n1-x
            t=a1(i-x:i+x,j-x:j+x);
            med=median(t(:));
            out(i-x,j-x)=med;
        end
    end
endfunction
```

Image Enhancement

Scilab code AP 14

```
function [result] = izigzag5(data)
//inverse ZigZag scanning of input data
```

```
N= sqrt(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(row,col) = data(z);
        z = z + 1;

        if(modulo(x,2) == 0)
          row = row - 1;
          col = col + 1;
        else
          row = row + 1;
          col = col - 1;
        end
    end
end
endfunction
```

Inverse Zig Zag scanning of pixels

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;
```

```
                col = col - 1;
            end
        end
end
endfunction
```

Zig Zag Scanning of Pixels

Scilab code AP 16
```
function [a2] = fft2d(a)
//a = any real or complex 2D matrix
//a2 = 2D-DFT of 2D matrix 'a'
m=size(a,1)
n=size(a,2)
// fourier transform along the rows
for i=1:n
a1(:,i)=exp(-2*%i*%pi*(0:m-1)'.*.(0:m-1)/m)*a(:,i)
end
// fourier transform along the columns
for j=1:m
a2temp=exp(-2*%i*%pi*(0:n-1)'.*.(0:n-1)/n)*(a1(j,:))
   .'
a2(j,:)=a2temp.'
end
for i = 1:m
    for j = 1:n
        if((abs(real(a2(i,j)))<0.0001)&(abs(imag(a2(
           i,j)))<0.0001))
            a2(i,j)=0;
        elseif(abs(real(a2(i,j)))<0.0001)
            a2(i,j)= 0+%i*imag(a2(i,j));
        elseif(abs(imag(a2(i,j)))<0.0001)
            a2(i,j)= real(a2(i,j))+0;
        end
    end
end
```

2D Fast Fourier Trasnform

Scilab code AP 7

```
function [a] =ifft2d(a2)
//a2 = 2D–DFT of any real or complex 2D matrix
//a = 2D–IDFT of a2
m=size(a2,1)
n=size(a2,2)
//Inverse Fourier transform along the rows
for i=1:n
a1(:,i)=exp(2*%i*%pi*(0:m-1)'.*.(0:m-1)/m)*a2(:,i)
end
//Inverse fourier transform along the columns
for j=1:m
atemp=exp(2*%i*%pi*(0:n-1)'.*.(0:n-1)/n)*(a1(j,:)).'
a(j,:)=atemp.'
end
a = a/(m*n)
a = real(a)
endfunction
```

Inverse 2D Fast Fourier Transform