Scilab Manual for
Digital Image Processing
by Dr Abhishek Choubey
Electronics Engineering
Sreenidhi Institute Of Science And Technology[1]

Solutions provided by
Dr Abhishek Choubey
Electronics Engineering
Sreenidhi Institute Of Science And Technology

May 14, 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 Experiments

List of Figures

Experiment: 1

Image analysis on 16 X 16 image size

Scilab code Solution 1.1 1

```
//Image analysis on 16*16 image size
//Scilab 5.4.1
//Windows 10
//Requires SIVP, IPD toolboxes

clear;
clc;

//Form an image of dimension 16x16 containing 16
   vertical strips
A=[0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15;
    0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15;
    0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15;
```

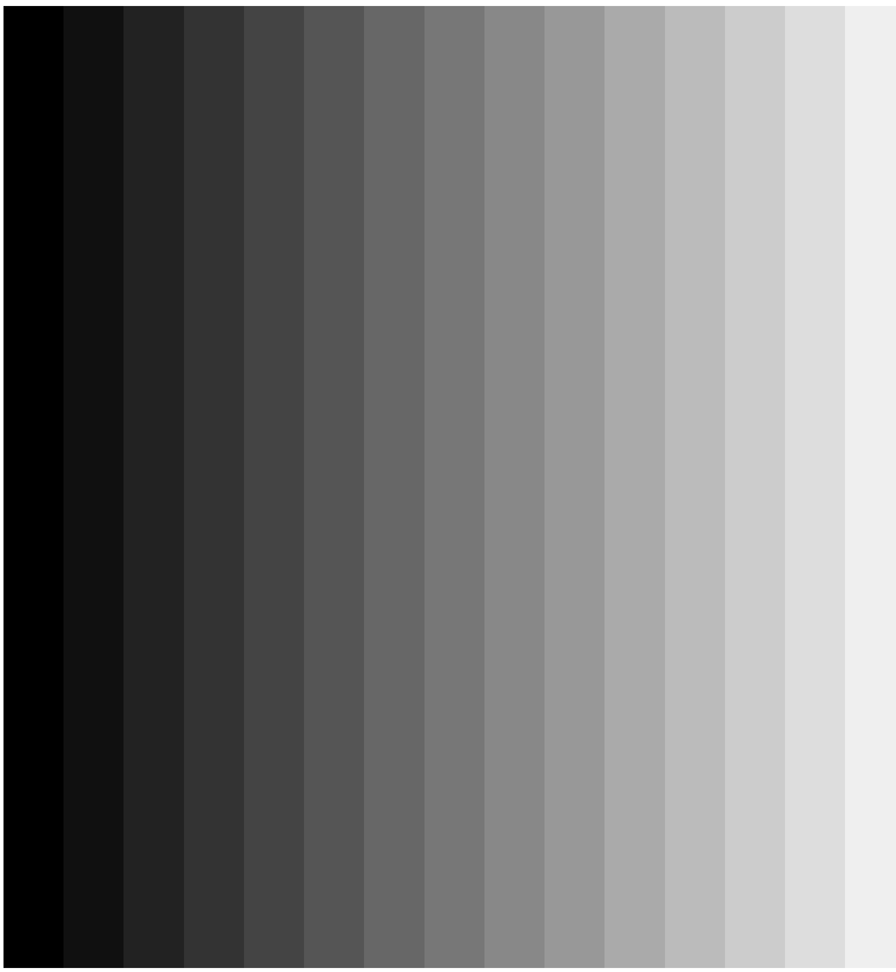

Figure 1.1: 1

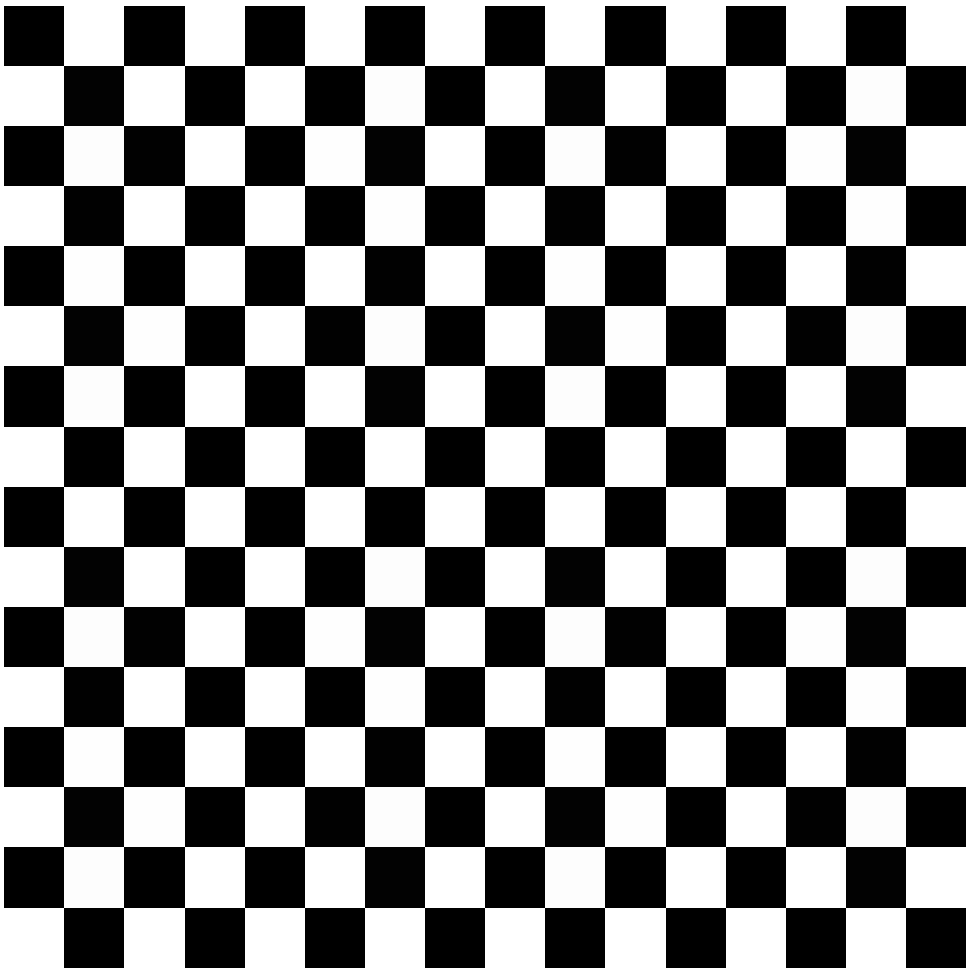

Figure 1.2: 1

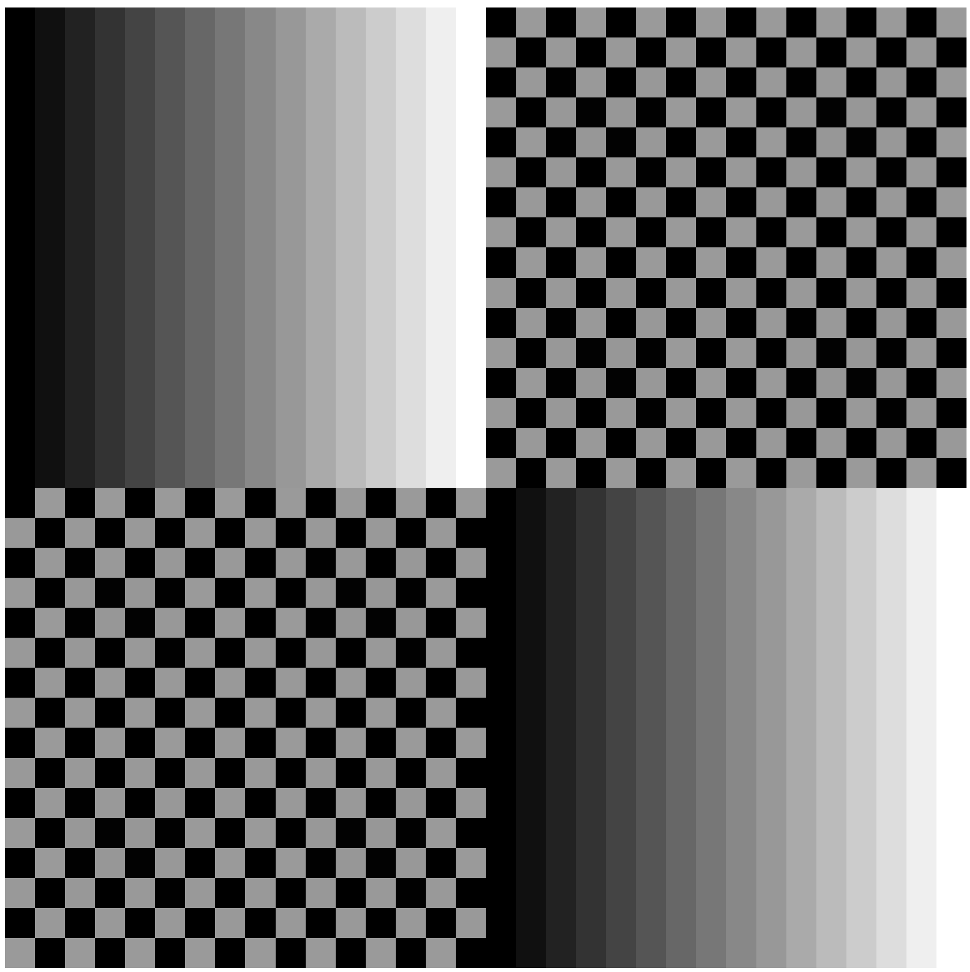

Figure 1.3: 1

```
     0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15;
     0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15;
     0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15;
     0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15;
     0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15;
     0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15;
     0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15;
     0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15;
     0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15;
     0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15;
     0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15;
     0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15;
     0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15];
A1=mat2gray(A);
imwrite(A1,'VerticalStrips.jpeg');

//Form a check-board "B" of dimension 16x16
   containing 16 blocks
a=[0 9; 9 0];
b=[a a; a a];
c=[b b; b b];
B=[c c; c c];
B1=mat2gray(B);
imwrite(B1,'Check-board.jpeg');

//Form image containing top-left and bottom-right
   quarter parts A and top-right & bottom-left
   quarters B.
C=[ A B; B A];
C1=mat2gray(C);
imwrite(C1,'Quarter.jpeg');
```

Experiment: 2

Image analysis on 256 X 256 image size

check Appendix AP 1 for dependency:

Cameramanimg.jpg

Scilab code Solution 2.2 2

```
//Image analysis on 256*256 image size
//Scilab 5.4.1
//Windows 10
```

Figure 2.1: 2

Figure 2.2: 2

Figure 2.3: 2

Figure 2.4: 2

Figure 2.5: 2

Figure 2.6: 2

```
//Requires SIVP, IPD toolboxes

clear;
clc;

I=imread('cameraman.jpg');

//Break the cameraman image of dimension 256x256 into four equal
//square shapes C11, C12, C21 & C22 and display all into a single
//figure of 2x2 dimensions.
C11=I(1:128, 1:128);
C12=I(1:128, 129:256);
C21=I(129:256, 1:128);
C22=I(129:256, 129:256);
imwrite(C11,'C11.jpeg');//Top Left
imwrite(C12,'C12.jpeg');//Top Right
imwrite(C21,'C21.jpeg');//Bottom Left
imwrite(C22,'C22.jpeg');//Bottom Right
J=[C11 C12; C21 C22];//Reconstruct original image from the squares
imwrite(J,'Single.jpeg');//Reconstructed image from squares

//Interchange the C11 & C22 and C12 & C21 and show the image
K=[C22 C21; C12 C11];
imwrite(K,'Interchange.jpeg');//Interchanged image
```

Experiment: 3

Singular Value Decomposition

Scilab code Solution 3.3 3

```
//Singular Value Decomposition
//Scilab 5.4.1
//Windows 10
//Requires SIVP, IPD toolboxes

clear;
clc;

A = [1,-2,3;3,2,-1];
[U,S,V]= svd(A);
A_recon = U*S*V';
disp(U,'U =')
disp(S,'S =')
disp(V,'V =')
disp(A_recon,'A matrix from svd =')

//Output
//
// U =
//
//  - 0.7071068    0.7071068
```

```
//     0.7071068    0.7071068
//
// S =
//
//     4.2426407    0.           0.
//     0.           3.1622777    0.
//
// V =
//
//     0.3333333    0.8944272  - 0.2981424
//     0.6666667    1.110D-16    0.7453560
//   - 0.6666667    0.4472136    0.5962848
//
// A matrix from svd =
//
//     1.  - 2.    3.
//     3.    2.  - 1.
```

Experiment: 4

Performing KL transform

Scilab code Solution 4.4 4

```
//Performing KL transform
//Scilab 5.4.1
//Windows 10
//Requires SIVP, IPD toolboxes

clear;
clc;

X = [3,5,6,7;5,6,3,3;4,6,7,5];
[m,n]= size(X);
A = [];
E = [];
for i =1:n
    A = A+X(:,i);
    E = E+X(:,i)*X(:,i)';
end
mx = A/n;                       //mean matrix
E = E/n;
C = E - mx*mx';                //covariance matrix C = E[xx']-mx*mx'
[V,D] = spec(C);               //eigen values and eigen
```

```
vectors
d = diag(D);                //diagonal elements od eigen values
[d,i] = gsort(d);           //sorting the elements of D in descending order
for j = 1:length(d)
    T(:,j)= V(:,i(j));
end
T =T'
disp(d,'Eigen Values are U = ')
disp(T,'The eigen vector matrix T =')
disp(T,'The KL tranform basis is =')

//KL transform
for i = 1:n
    Y(:,i)= T*X(:,i);
end
disp(Y,'KL transformation of the input matrix Y =')

//Reconstruction
for i = 1:n
    x(:,i)= T'*Y(:,i);
end
disp(x,'Reconstruct matrix of the given sample matrix X =')

//Output
//
// Eigen Values are U =
//
//    3.6278623
//    1.0409979
//    0.4561398
//
// The eigen vector matrix T =
//
//    0.7383786  - 0.5693168    0.3614907
//    0.0603190    0.5896337    0.8054152
```

```
//     0.6716835     0.5728966   - 0.4697135
//
// The KL tranform basis is =
//
//     0.7383786   - 0.5693168     0.3614907
//     0.0603190     0.5896337     0.8054152
//     0.6716835     0.5728966   - 0.4697135
//
// KL transformation of the input matrix Y =
//
//     0.8145143     2.444936      5.2527556
   5.2681528
//     6.3507865     8.6718888     7.7687219
   6.2182105
//     3.0006794     3.977516      2.4607962
   4.0719067
//
// Reconstruct matrix of the given sample matrix X =
//
//     3.     5.     6.     7.
//     5.     6.     3.     3.
//     4.     6.     7.     5.
//
```

Experiment: 5

Brightness enhancement and supression

check Appendix AP 5 for dependency:

`baboon.png`

Scilab code Solution 5.5 5

```
//Brightness enhancement and supression
//Scilab 5.4.1
//Windows 10
//Requires SIVP, IPD toolboxes

clear;
clc;

a=imread('baboon.png');

//Brightness Enhancement
```

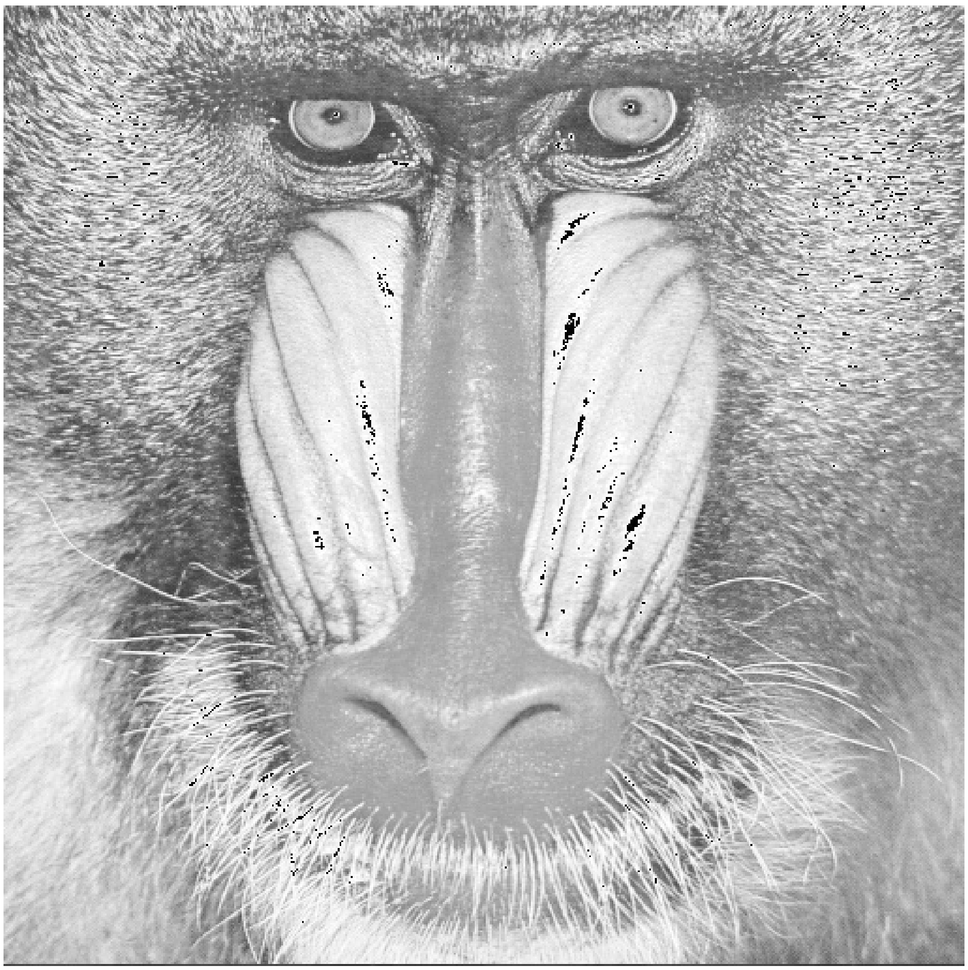

Figure 5.1: 5

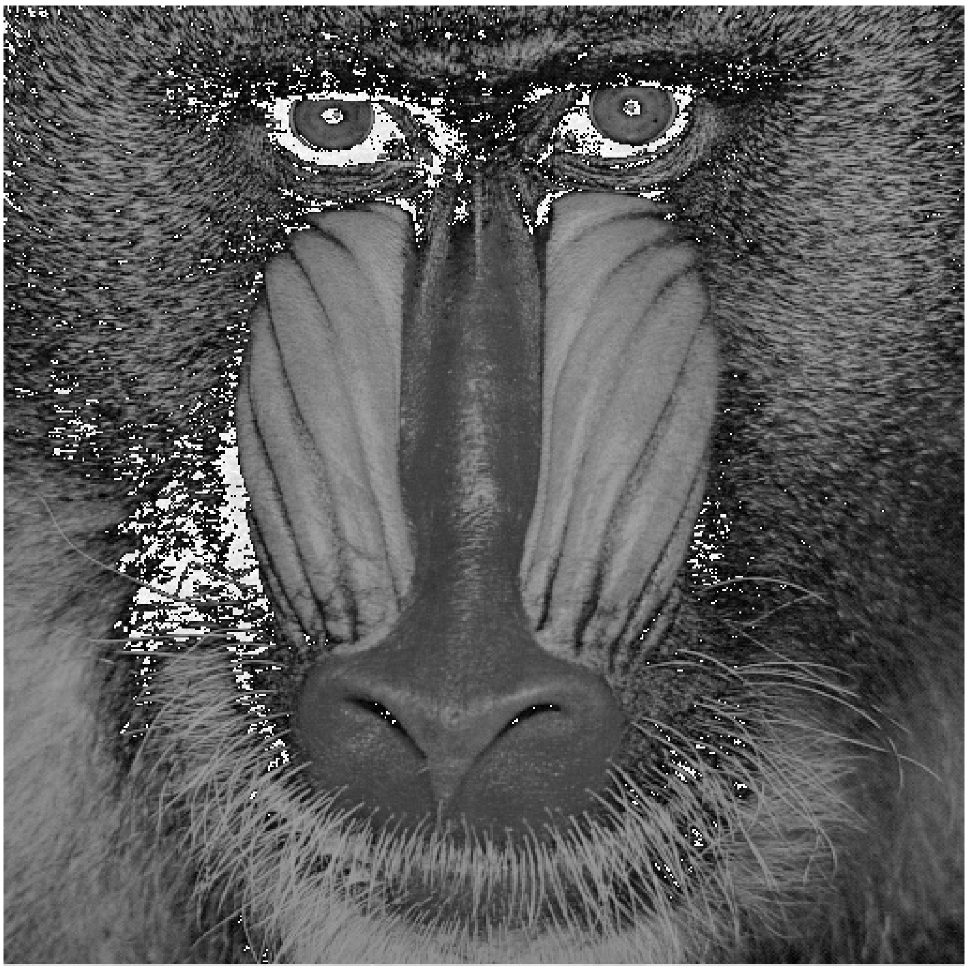

Figure 5.2: 5

```
a = rgb2gray(a);
b = double(a)+50;
b = uint8(b);
imwrite(b,'BrightnessEnhancedImage.jpeg');

a=imread('baboon.png');

//Brightness suppression
a = rgb2gray(a);
b = double(a)-50;
b = uint8(b);
imwrite(b,'BrightnessSupressedImage.jpeg');
```

Experiment: 6

Contrast Manipulation

check Appendix AP 3 for dependency:

`Lenna.png`

Scilab code Solution 6.6 6

```
//Contrast Manipulation
//Scilab 5.4.1
//Windows 10
//Requires SIVP, IPD toolboxes

clear;
clc;

a=imread('Lenna.png');

a = rgb2gray(a);
b = double(a)*0.5;
b = uint8(b)
```

Figure 6.1: 6

Figure 6.2: 6

```
c = double(b)*2;
c = uint8(c)

imwrite(b,'DecreaseinContrast.jpeg');
imwrite(c,'IncreaseinContrast.jpeg');
```

Experiment: 7

Color separation into R,B,G

check Appendix AP 2 for dependency:

peppers.png

Scilab code Solution 7.7 7

```
//Color separation into R,B,G
//Scilab 5.4.1
//Windows 10
//Requires SIVP, IPD toolboxes

clear;
clc;

a=imread('peppers.png');

a1 = a;
```

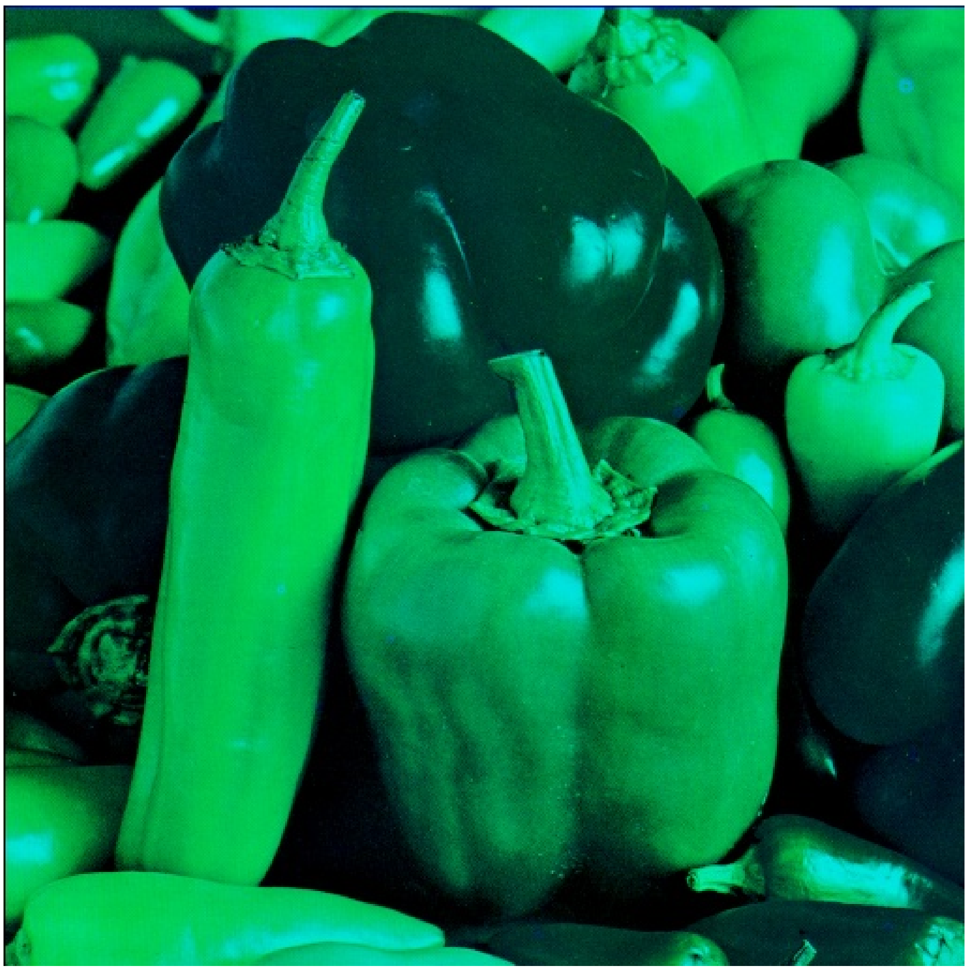

Figure 7.1: 7

Figure 7.2: 7

Figure 7.3: 7

```
b1 = a;
c1 = a;
a1(:,:,1)=0;
b1(:,:,2)=0;
c1(:,:,3)=0;

imwrite(a1, 'RedMissing.jpeg');
imwrite(b1, 'GreenMissing.jpeg');
imwrite(c1, 'BlueMissing.jpeg');
```

Experiment: 8

Gamma correction

check Appendix AP 4 for dependency:

`ararauna.png`

Scilab code Solution 8.8 8

```
//Gamma correction
//Scilab 5.4.1
//Windows 10
//Requires SIVP, IPD toolboxes

clear;
clc;

I=imread('ararauna.png');

gamma_Value = 0.5;
max_intensity = 255; //for uint8 image
//Look up table creation
LUT = max_intensity.*(([0:max_intensity]./
   max_intensity).^gamma_Value);
```

Figure 8.1: 8

```
LUT = floor(LUT);
//Mapping of input pixels into lookup table values
K = double(I)+1;
J = zeros(I);
[m,n,p]= size(K);
for i = 1:m
 for j =1:n
     for k = 1:p
          J(i,j,k)= LUT(K(i,j,k));
      end
    end
end

imwrite(uint8(J), 'GammaCorrectedImage.jpeg');    // IPD toolbox
```

Experiment: 9

Adding various types of noises to image

check Appendix AP 3 for dependency:

`Lenna.png`

Scilab code Solution 9.9 9

```
//Add various types of noises to image
//Scilab 5.4.1
//Windows 10
//Requires SIVP, IPD toolboxes

clear;
clc;

lenna=imread('Lenna.png');
```

Figure 9.1: 9

Figure 9.2: 9

Figure 9.3: 9

```
//Gaussian
lenaNgaussian = imnoise(lenna,'gaussian');
imwrite(lenaNgaussian,'lenaNgaussian.jpeg');

//Speckle
lenaNspeckle = imnoise(lenna,'speckle');
imwrite(lenaNspeckle,'lenaNspeckle.jpeg');

//Salt & Pepper
d=0.25  //drop out noise
lenaNsalpep = imnoise(lenna,'salt & pepper',d);
imwrite(lenaNsalpep,'lenaNsalpep.jpeg');
```

Experiment: 10

Demonstrate the various Image Conversions

check Appendix AP 1 for dependency:

`Cameramanimg.jpg`

check Appendix AP 3 for dependency:

`Lenna.png`

check Appendix AP 4 for dependency:

`ararauna.png`

check Appendix AP 5 for dependency:

`baboon.png`

check Appendix AP 2 for dependency:

`peppers.png`

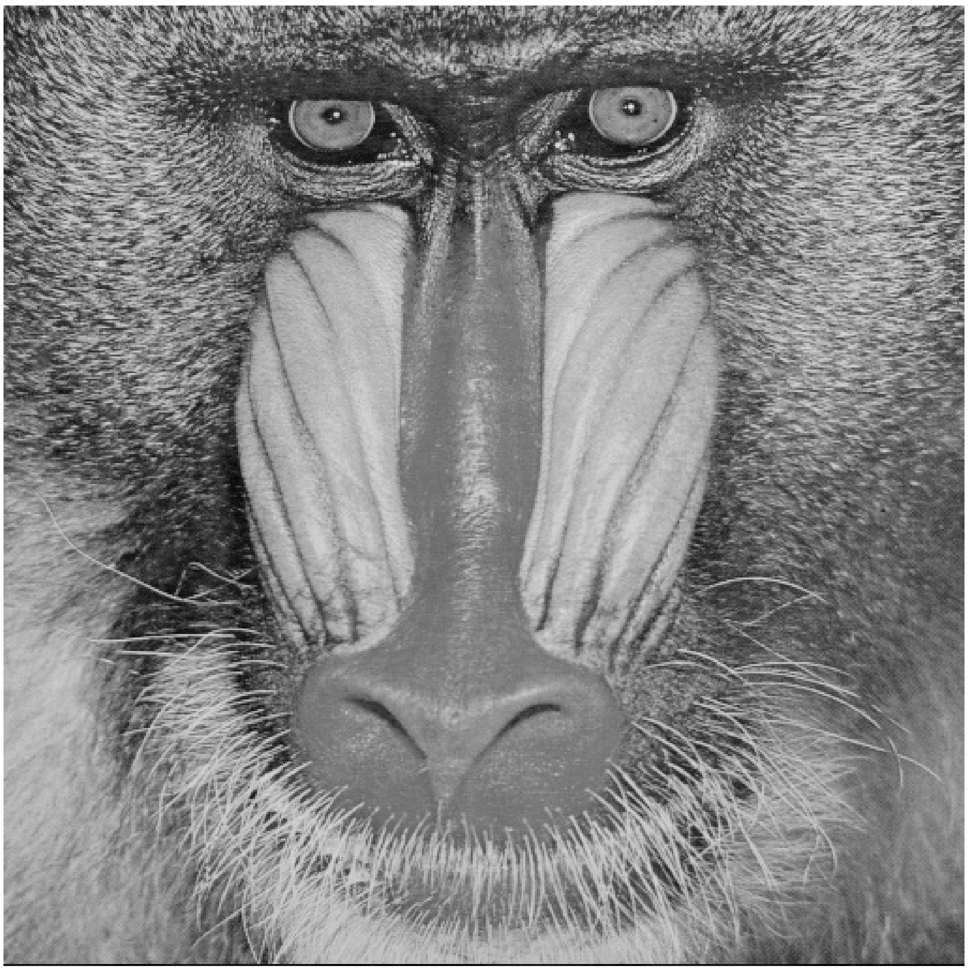

Figure 10.1: 10

Figure 10.2: 10

Figure 10.3: 10

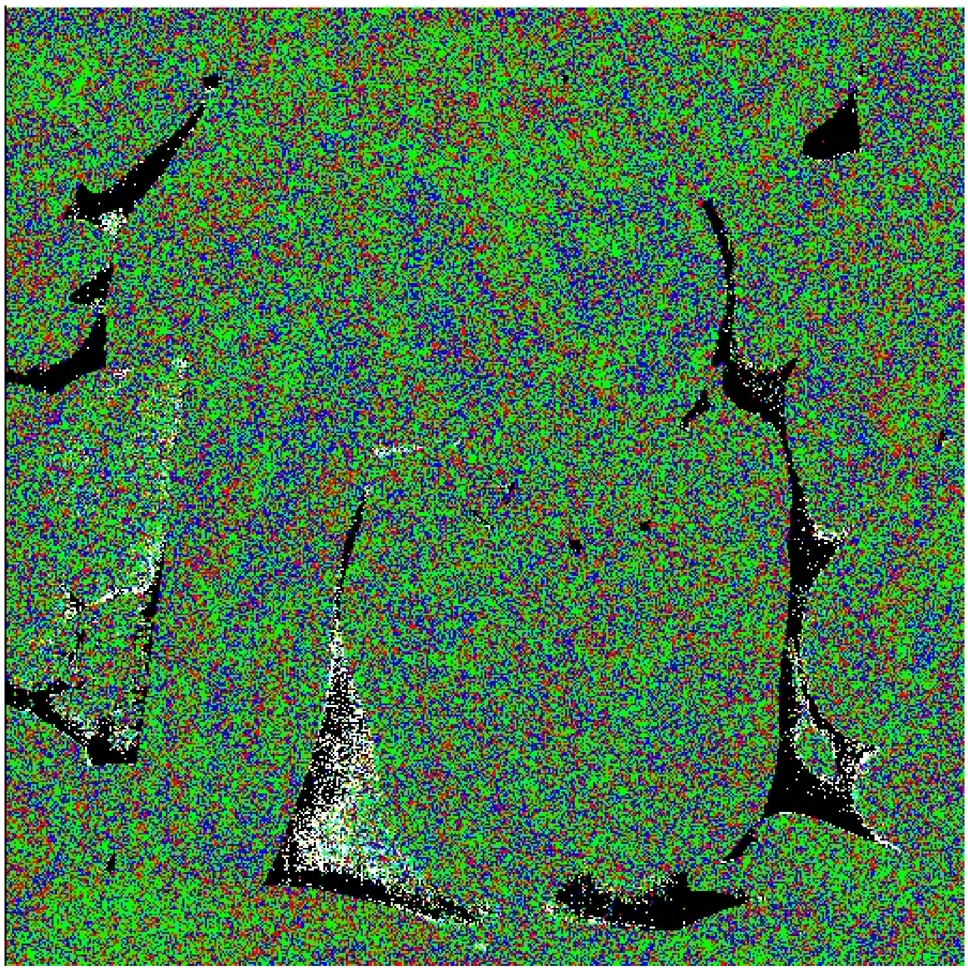

Figure 10.4: 10

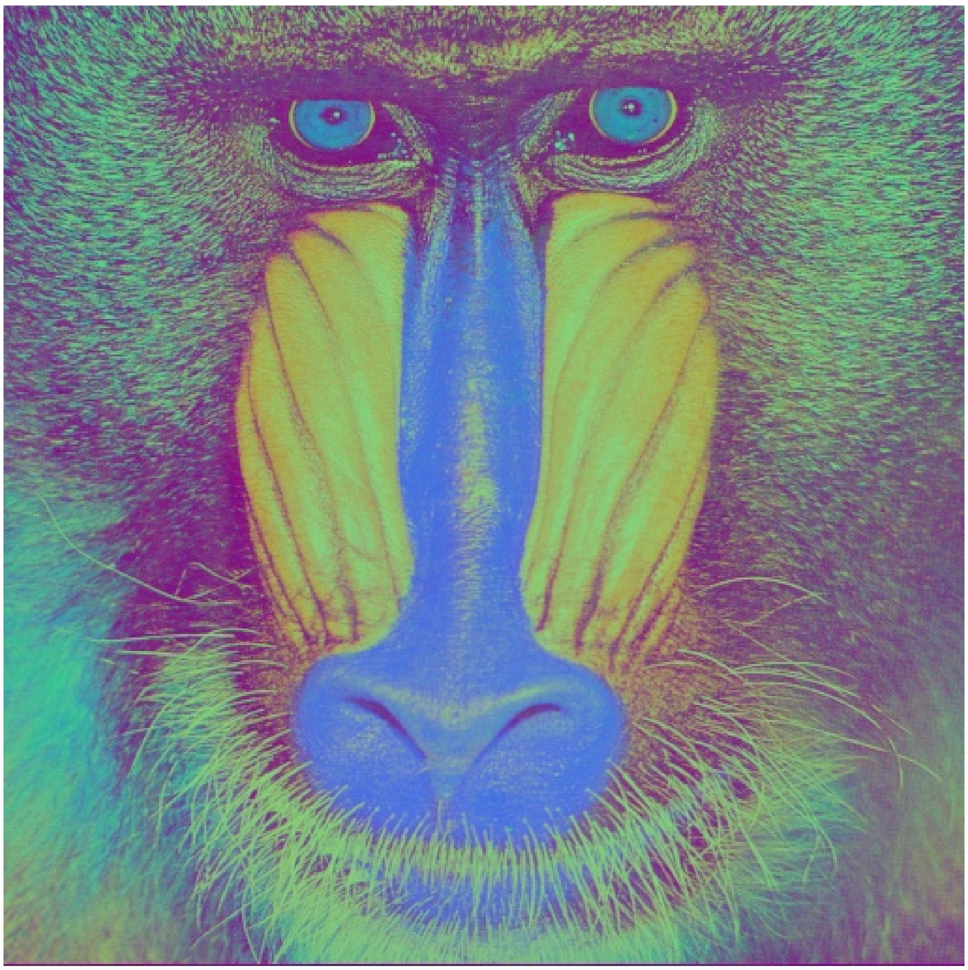

Figure 10.5: 10

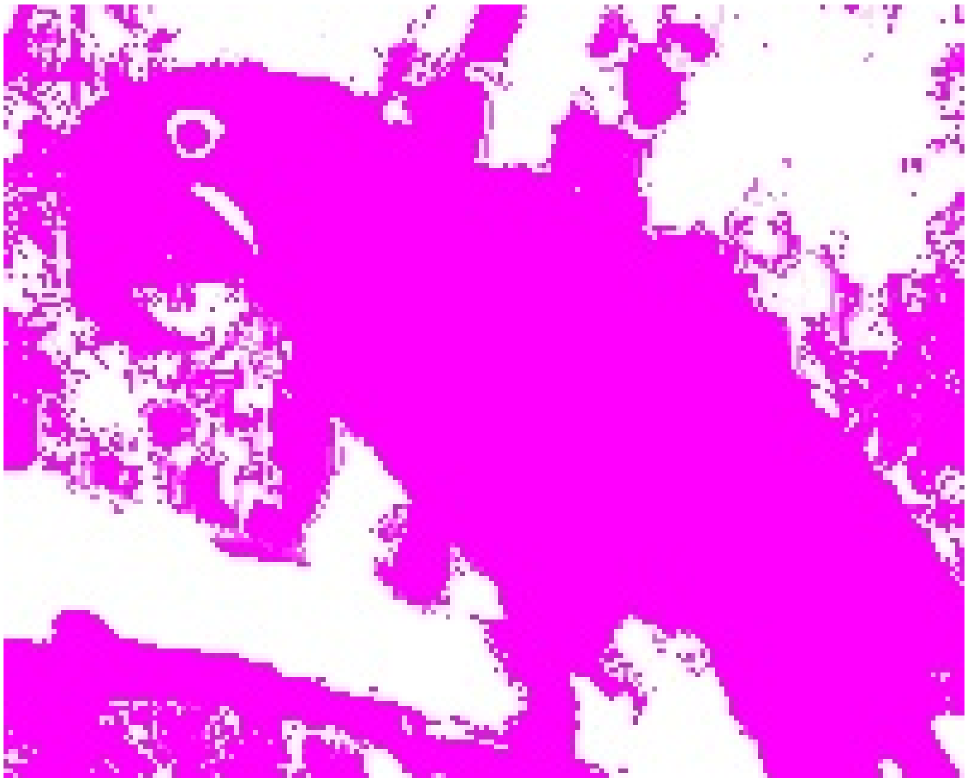

Figure 10.6: 10

Scilab code Solution 10.10 10

```
//Demonstrate the various Image Conversions
//Scilab 5.4.1
//Windows 10
```

```
//Requires SIVP, IPD toolboxes

clear;
clc;

//RBG to Gray scale
baboon = imread('baboon.png');
babgray = rgb2gray(baboon);
imwrite(babgray,'babgray.jpeg');

//RBG to Binary
lena = imread('Lenna.png');
lenabw = im2bw(lena,0.5);
imwrite(lenabw,'lenabw.jpeg');

//RBG to HSV
cameraman = imread('cameraman.jpg');
cameramanhsv = rgb2hsv(cameraman);
imwrite(cameramanhsv,'cameramanhsv.jpeg');

//HSV to RGB
peppers = imread('peppers.png');
peppersrgb = hsv2rgb(peppers);
imwrite(peppersrgb,'peppersrgb.jpeg');

//RBG to YCbCr
baboon = imread('baboon.png');
baboonycbcr = rgb2ycbcr(baboon);
imwrite(baboonycbcr,'baboonycbcr.jpeg');

//YCbCr to RGB
ararauna = imread('ararauna.png');
araraunargb = ycbcr2rgb(ararauna);
imwrite(araraunargb,'araraunargb.jpeg');
```

Experiment: 11

Demonstrate Spatial Domain Processing

check Appendix AP 3 for dependency:

`Lenna.png`

Scilab code Solution 11.11 11

```
//Demonstrate Spatial Domain Processing
//Scilab 5.4.1
//Windows 10
//Requires SIVP, IPD toolboxes

clear;
clc;

lenna=imread('Lenna.png');
```

Figure 11.1: 11

Figure 11.2: 11

Figure 11.3: 11

```

//Sobel
h = fspecial('sobel');
lenaSobel = imfilter(lenna,h)
imwrite(lenaSobel,'lenaSobel.jpeg');

//Prewitt
h = fspecial('prewitt');
lenaPrewitt = imfilter(lenna,h)
imwrite(lenaPrewitt,'lenaPrewitt.jpeg');

//Laplacian
h = fspecial('laplacian');
lenaLaplacian = imfilter(lenna,h)
imwrite(lenaLaplacian,'lenaLaplacian.jpeg');
```

Experiment: 12

Motion blur of an image

check Appendix AP 4 for dependency:

araraura.png

Scilab code Solution 12.12 12

```
//Motion blur of an image
//Scilab 5.4.1
//Windows 10
//Requires SIVP, IPD toolboxes

clear;
clc;

a=imread('ararauna.png');

//filter coefficients of fspecial('motion',10,25)
H =[0,0,0,0,0,0,0,0.0032,0.0449,0.0865,0.0072;...
0,0,0,0,0,0.0092,0.0509,0.0925,0.0629,0.0213,0;...
0,0,0,0.0152,0.0569,0.0985,0.0569,0.0152,0,0,0;...
0,0.0213,0.0629,0.0925,0.0509,0.0092,0,0,0,0,0;...
```

Figure 12.1: 12

```
0.0072,0.0865,0.0449,0.0032,0,0,0,0,0,0,0];
Motion_Blur = imfilter(a,H);
Motion_Blur =uint8(Motion_Blur);

imwrite(Motion_Blur,'MotionBlurredImage.jpeg')
```

Experiment: 13

Trimmed Average Filter

check Appendix AP 3 for dependency:

`Lenna.png`

Scilab code Solution 13.13 13

```
//Trimmed Average Filter
//Scilab 5.4.1
//Windows 10
//Requires SIVP, IPD toolboxes

clear;
clc;

c=imread('Lenna.png');
s = 1; //s denotes the number of values to be left in the end
r = 1;
N = 9; //3x3 window
```

Figure 13.1: 13

Figure 13.2: 13

```
a = double(imnoise(c,'gaussian'));
[m,n] = size(a);
b = zeros(m,n);
for i= 2:m-1
    for j = 2:n-1
        mat = [a(i,j),a(i,j-1),a(i,j+1),a(i-1,j),a(i+1,j),a(i-1,j-1),...
                a(i-1,j+1),a(i-1,j+1),a(i+1,j+1)];
        sorted_mat = gsort(mat);
        Sum=0;
        for k=r+s:(N-s)
            Sum = Sum+mat(k);
        end
        b(i,j)= Sum/(N-r-s);
    end
end
a = uint8(a);
b = uint8(b);
//figure
//imshow(c)
//title('Original Image')

imwrite(a,'noisyimage.jpeg')
imwrite(b,'TrimmedAverageFilteredImage.jpeg')
```

Experiment: 14

Determine image negative

check Appendix AP 2 for dependency:

`peppers.png`

Scilab code Solution 14.14 14

```
//Determine image negative
//Scilab 5.4.1
//Windows 10
//Requires SIVP, IPD toolboxes

clear;
clc;

a=imread('peppers.png');
k = 255-double(a);
k = uint8(k);
imwrite(k,'ImageNegative.jpeg')
```

Figure 14.1: 14

Experiment: 15

Image operations to perform clockwise and anti-clockwise operations

check Appendix AP 1 for dependency:

`Cameramanimg.jpg`

Scilab code Solution 15.15 15

```
//Image operations to perform clockwise and anti-
    clockwise operations
//Scilab 5.4.1
//Windows 10
//Requires SIVP, IPD toolboxes

clear;
clc;

```

Figure 15.1: 15

Figure 15.2: 15

```
A = imread('Cameramanimg.jpg');

//Rotate the Image anticlockwise by an angle of 90
   degrees
[M,N]=size(A);
for i=1:N
    for j=1:M
        B(j,i)=A(i,j);
    end
end
NM=B(N:-1:1,:);
imwrite(NM,'anticlockwise90.jpeg')

//Rotate the Image by an angle of 180 degrees
B= A(size(A,1):-1:1,size(A,1):-1:1,:);
imwrite(B,'clockwise180.jpeg')
```

Appendix


Cameramanimg

 pep-
pers

Lenna

 ara-
rauna

 ba-
boon