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

November 6, 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 Image analysis on 16 X 16 image size 7

2 Image analysis on 256 X 256 image size 12

3 Singular Value Decomposition 20

4 Performing KL transform 22

5 Brightness enhancement and supression 25

6 Contrast Manipulation 29

7 Color separation into R,B,G 33

8 Gamma correction 38

9 Adding various types of noises to image 41

10 Demonstrate the various Image Conversions 46

11 Demonstrate Spatial Domain Processing 54

12 Motion blur of an image 59

13 Trimmed Average Filter 62

14 Determine image negative 66

15 Image operations to perform clockwise and anti-clockwise operations **68**

List of Experiments

Solution 1.1	1	7
Solution 2.2	2	12
Solution 3.3	3	20
Solution 4.4	4	22
Solution 5.5	5	25
Solution 6.6	6	29
Solution 7.7	7	33
Solution 8.8	8	38
Solution 9.9	9	41
Solution 10.10	10	52
Solution 11.11	11	54
Solution 12.12	12	59
Solution 13.13	13	62
Solution 14.14	14	66
Solution 15.15	15	68
AP 1	Cameramanimg	73
AP 2	peppers	74
AP 3	Lenna	75
AP 4	ararauna	76
AP 5	baboon	77

List of Figures

1.1 1 8
1.2 1 9
1.3 1 10

2.1 2 13
2.2 2 14
2.3 2 15
2.4 2 16
2.5 2 17
2.6 2 18

5.1 5 26
5.2 5 27

6.1 6 30
6.2 6 31

7.1 7 34
7.2 7 35
7.3 7 36

8.1 8 39

9.1 9 42
9.2 9 43
9.3 9 44

10.1 10 47
10.2 10 48
10.3 10 49

10.4 10 50
10.5 10 51
10.6 10 52

11.1 11 55
11.2 11 56
11.3 11 57

12.1 12 60

13.1 13 63
13.2 13 64

14.1 14 67

15.1 15 69
15.2 15 70

Experiment: 1

Image analysis on 16 X 16 image size

Scilab code Solution 1.1 1

```
1 //Image analysis on 16*16 image size
2 //Scilab 5.4.1
3 //Windows 10
4 //Requires SIVP, IPD toolboxes
5
6 clear;
7 clc;
8
9 //Form an image of dimension 16x16 containing 16
     vertical strips
10 A=[0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15;
11      0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15;
12      0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15;
```

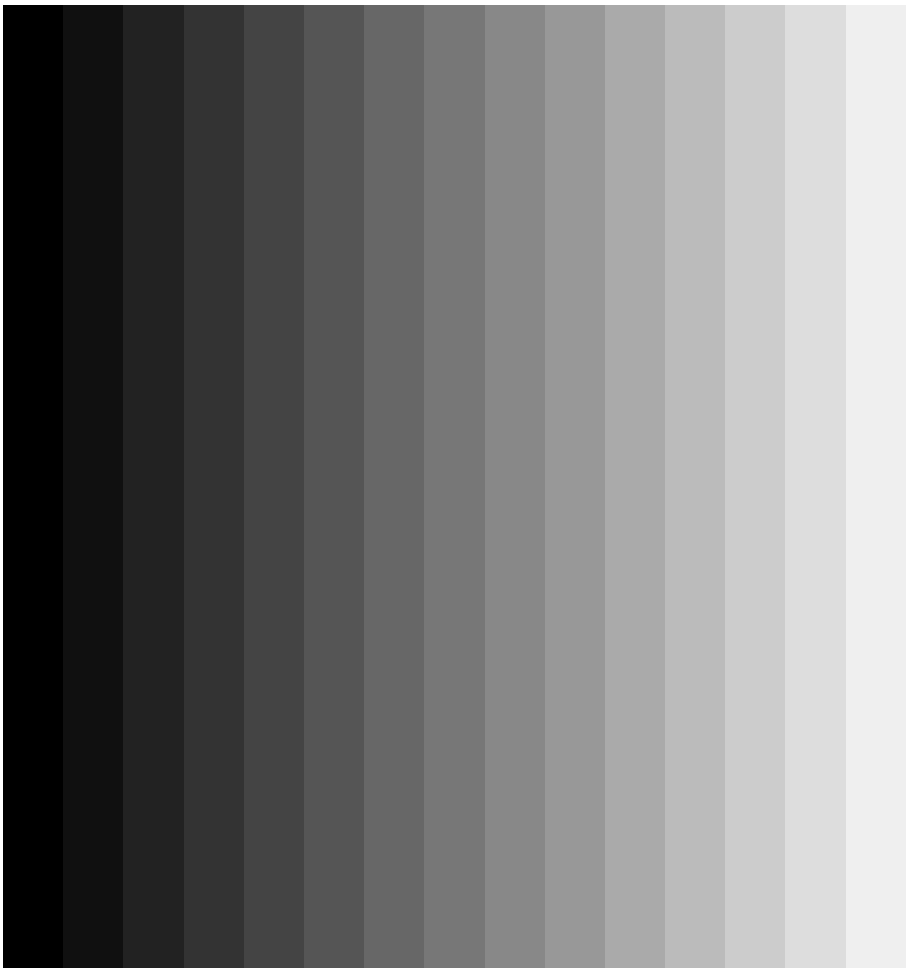


Figure 1.1: 1

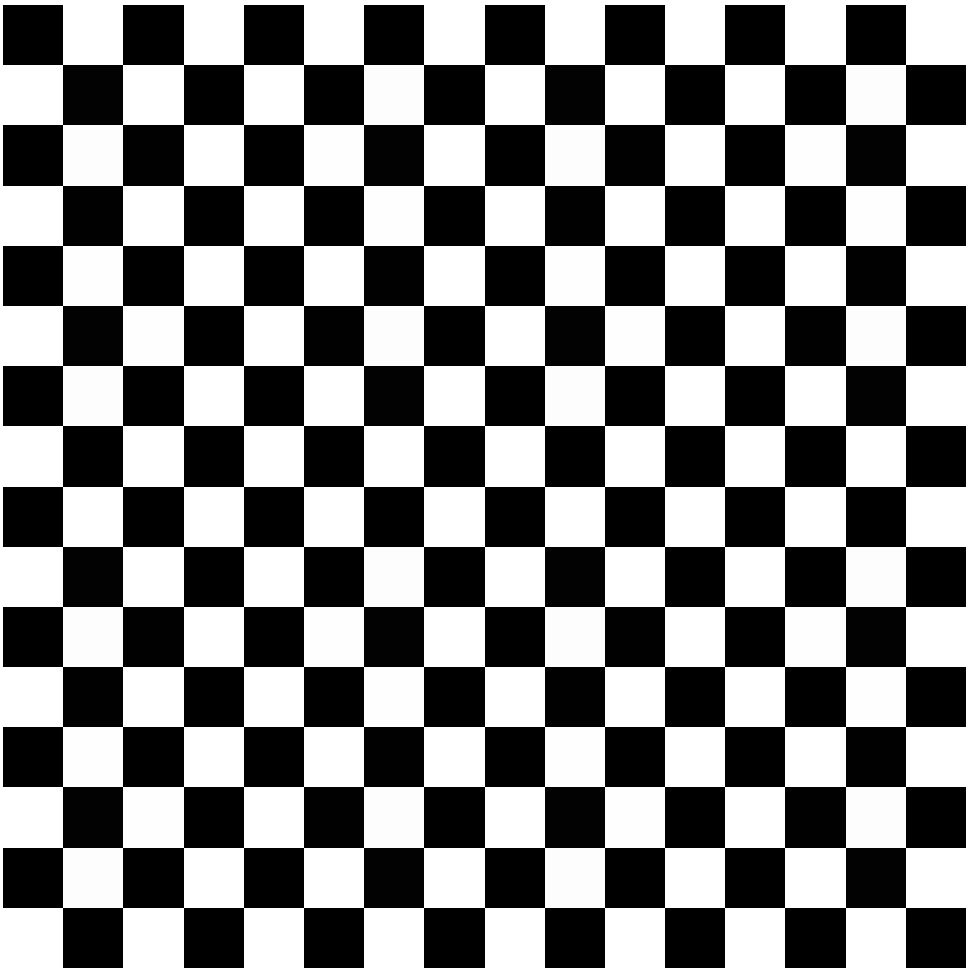


Figure 1.2: 1

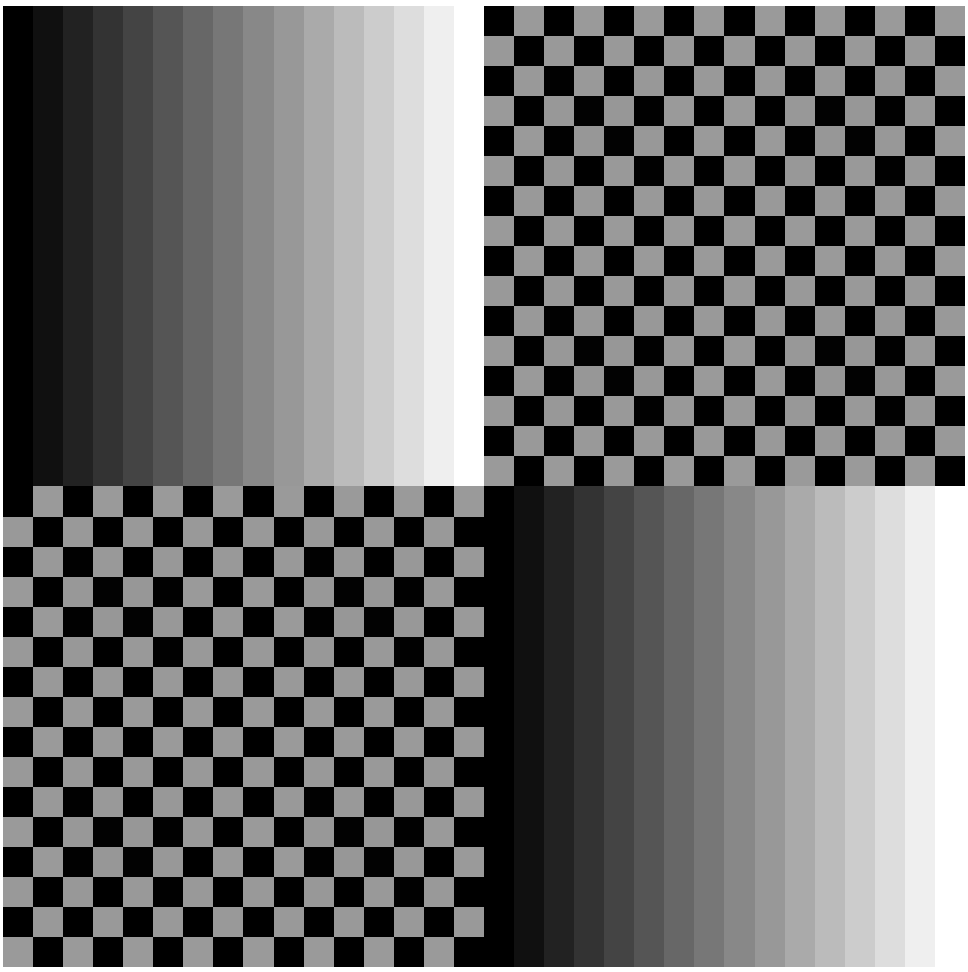


Figure 1.3: 1

```
13      0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15;
14      0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15;
15      0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15;
16      0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15;
17      0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15;
18      0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15;
19      0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15;
20      0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15;
21      0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15;
22      0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15;
23      0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15;
24      0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15;
25      0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15];
26 A1=mat2gray(A);
27 imwrite(A1,'VerticalStrips.jpeg');
28
29 //Form a check-board "B" of dimension 16x16
      containing 16 blocks
30 a=[0 9; 9 0];
31 b=[a a; a a];
32 c=[b b; b b];
33 B=[c c; c c];
34 B1=mat2gray(B);
35 imwrite(B1,'Check-board.jpeg');
36
37 //Form image containing top-left and bottom-right
      quarter parts A and top-right & bottom-left
      quarters B.
38 C=[ A B; B A];
39 C1=mat2gray(C);
40 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

```
1 //Image analysis on 256*256 image size
2 //Scilab 5.4.1
3 //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

```
4  //Requires SIVP, IPD toolboxes
5
6  clear;
7  clc;
8
9  I=imread('cameraman.jpg');
10
11 //Break the cameraman image of dimension 256x256
       into four equal
12 //square shapes C11, C12, C21 & C22 and display all
       into a single
13 //figure of 2x2 dimensions.
14 C11=I(1:128, 1:128);
15 C12=I(1:128, 129:256);
16 C21=I(129:256, 1:128);
17 C22=I(129:256, 129:256);
18 imwrite(C11,'C11.jpeg');//Top Left
19 imwrite(C12,'C12.jpeg');//Top Right
20 imwrite(C21,'C21.jpeg');//Bottom Left
21 imwrite(C22,'C22.jpeg');//Bottom Right
22 J=[C11 C12; C21 C22];//Reconstruct original image
       from the squares
23 imwrite(J,'Single.jpeg');//Reconstructed image from
       squares
24
25 //Interchange the C11 & C22 and C12 & C21 and show
       the image
26 K=[C22 C21; C12 C11];
27 imwrite(K,'Interchange.jpeg');//Interchanged image
```

Experiment: 3

Singular Value Decomposition

Scilab code Solution 3.3 3

```
1  //Singular Value Decomposition
2  //Scilab 5.4.1
3  //Windows 10
4  //Requires SIVP, IPD toolboxes
5
6  clear;
7  clc;
8
9  A = [1,-2,3;3,2,-1];
10 [U,S,V]= svd(A);
11 A_recon = U*S*V';
12 disp(U,'U =')
13 disp(S,'S =')
14 disp(V,'V =')
15 disp(A_recon,'A matrix from svd =')
16
17 //Output
18 //
19 // U =
20 //
21 //  - 0.7071068    0.7071068
```

```
22 //     0.7071068     0.7071068
23 //
24 // S =
25 //
26 //     4.2426407     0.            0.
27 //     0.            3.1622777     0.
28 //
29 // V =
30 //
31 //     0.3333333     0.8944272   - 0.2981424
32 //     0.6666667     1.110D-16     0.7453560
33 //   - 0.6666667     0.4472136     0.5962848
34 //
35 // A matrix from svd =
36 //
37 //     1.   - 2.     3.
38 //     3.     2.   - 1.
```

Experiment: 4

Performing KL transform

Scilab code Solution 4.4 4

```
1  //Performing KL transform
2  //Scilab 5.4.1
3  //Windows 10
4  //Requires SIVP, IPD toolboxes
5
6  clear;
7  clc;
8
9  X = [3,5,6,7;5,6,3,3;4,6,7,5];
10 [m,n]= size(X);
11 A = [];
12 E = [];
13 for i =1:n
14     A = A+X(:,i);
15     E = E+X(:,i)*X(:,i)';
16 end
17 mx = A/n;                          //mean matrix
18 E = E/n;
19 C = E - mx*mx';                    //covariance matrix C =
     E[xx']-mx*mx'
20 [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
```

```
55 //    0.6716835    0.5728966  - 0.4697135
56 //
57 // The KL tranform basis is =
58 //
59 //    0.7383786  - 0.5693168    0.3614907
60 //    0.0603190    0.5896337    0.8054152
61 //    0.6716835    0.5728966  - 0.4697135
62 //
63 // KL transformation of the input matrix Y =
64 //
65 //    0.8145143    2.444936     5.2527556
      5.2681528
66 //    6.3507865    8.6718888    7.7687219
      6.2182105
67 //    3.0006794    3.977516     2.4607962
      4.0719067
68 //
69 // Reconstruct matrix of the given sample matrix X =
70 //
71 //    3.    5.    6.    7.
72 //    5.    6.    3.    3.
73 //    4.    6.    7.    5.
74 //
```

Experiment: 5

Brightness enhancement and supression

check Appendix AP 5 for dependency:

baboon.png

Scilab code Solution 5.5 5

```
1 //Brightness enhancement and supression
2 //Scilab 5.4.1
3 //Windows 10
4 //Requires SIVP, IPD toolboxes
5
6 clear;
7 clc;
8
9 a=imread('baboon.png');
10
11 //Brightness Enhancement
```

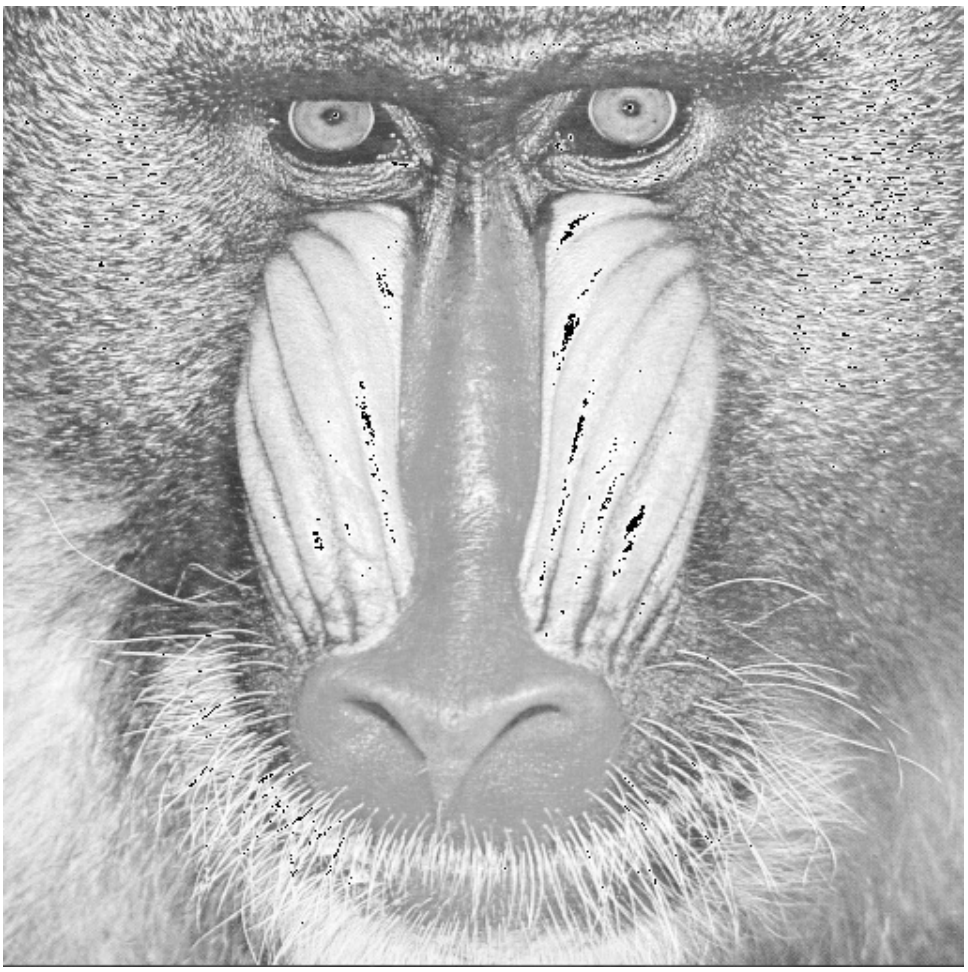


Figure 5.1: 5

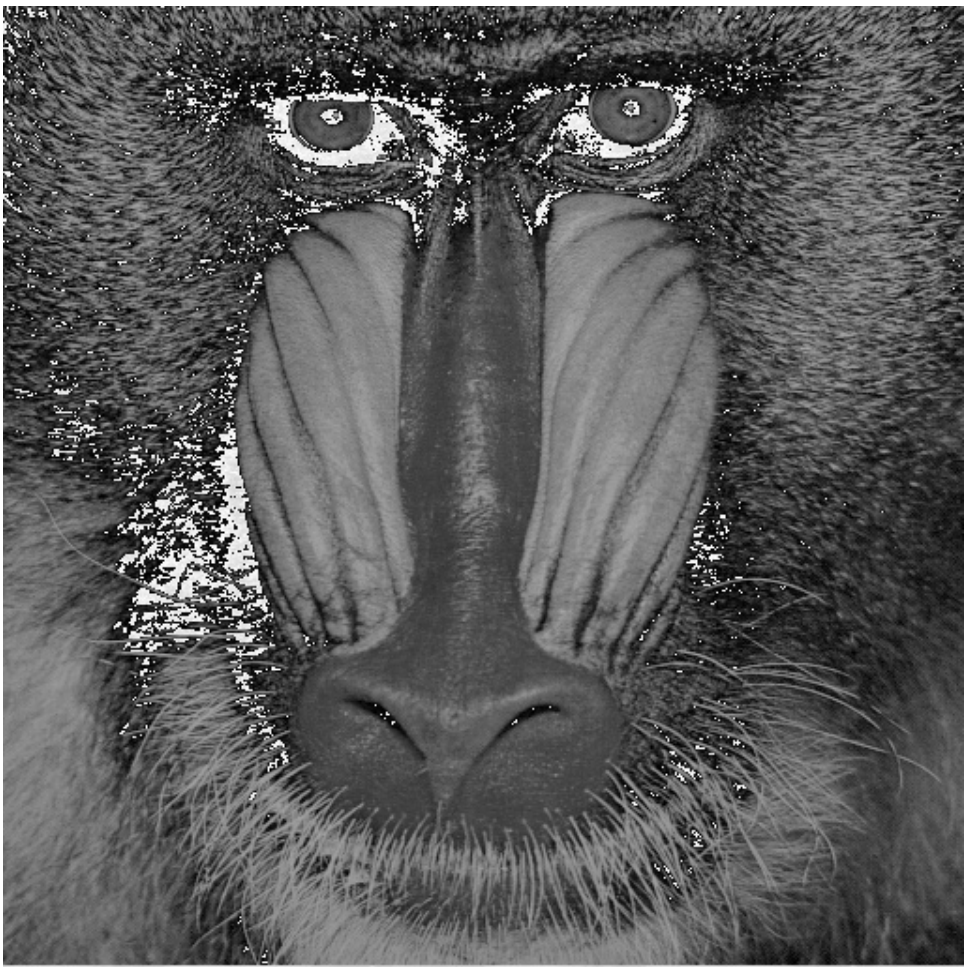


Figure 5.2: 5

```
12 a = rgb2gray(a);
13 b = double(a)+50;
14 b = uint8(b);
15 imwrite(b,'BrightnessEnhancedImage.jpeg');
16
17 a=imread('baboon.png');
18
19 //Brightness suppression
20 a = rgb2gray(a);
21 b = double(a)-50;
22 b = uint8(b);
23 imwrite(b,'BrightnessSupressedImage.jpeg');
```

Experiment: 6

Contrast Manipulation

check Appendix AP 3 for dependency:

Lenna.png

Scilab code Solution 6.6 6

```
1  //Contrast Manipulation
2  //Scilab 5.4.1
3  //Windows 10
4  //Requires SIVP, IPD toolboxes
5
6  clear;
7  clc;
8
9  a=imread('Lenna.png');
10
11 a = rgb2gray(a);
12 b = double(a)*0.5;
13 b = uint8(b)
```



Figure 6.1: 6



Figure 6.2: 6

```
14 c = double(b)*2;
15 c = uint8(c)
16
17 imwrite(b,'DecreaseinContrast.jpeg');
18 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

```
1 //Color separation into R,B,G
2 //Scilab 5.4.1
3 //Windows 10
4 //Requires SIVP, IPD toolboxes
5
6 clear;
7 clc;
8
9 a=imread('peppers.png');
10
11 a1 = a;
```



Figure 7.1: 7



Figure 7.2: 7



Figure 7.3: 7

```
12 b1 = a;
13 c1 = a;
14 a1(:,:,1)=0;
15 b1(:,:,2)=0;
16 c1(:,:,3)=0;
17
18 imwrite(a1, 'RedMissing.jpeg');
19 imwrite(b1, 'GreenMissing.jpeg');
20 imwrite(c1, 'BlueMissing.jpeg');
```

Experiment: 8

Gamma correction

check Appendix AP 4 for dependency:

ararauna.png

Scilab code Solution 8.8 8

```
1  //Gamma correction
2  //Scilab 5.4.1
3  //Windows 10
4  //Requires SIVP, IPD toolboxes
5
6  clear;
7  clc;
8
9  I=imread('ararauna.png');
10
11 gamma_Value = 0.5;
12 max_intensity = 255; //for uint8 image
13 //Look up table creation
14 LUT = max_intensity.*(([0:max_intensity]./
       max_intensity).^gamma_Value);
```



Figure 8.1: 8

```
15 LUT = floor(LUT);
16 //Mapping of input pixels into lookup table values
17 K = double(I)+1;
18 J = zeros(I);
19 [m,n,p]= size(K);
20 for i = 1:m
21  for j =1:n
22      for k = 1:p
23           J(i,j,k)= LUT(K(i,j,k));
24       end
25     end
26 end
27
28 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

```
1 //Add various types of noises to image
2 //Scilab 5.4.1
3 //Windows 10
4 //Requires SIVP, IPD toolboxes
5
6 clear;
7 clc;
8
9 lenna=imread('Lenna.png');
```



Figure 9.1: 9



Figure 9.2: 9

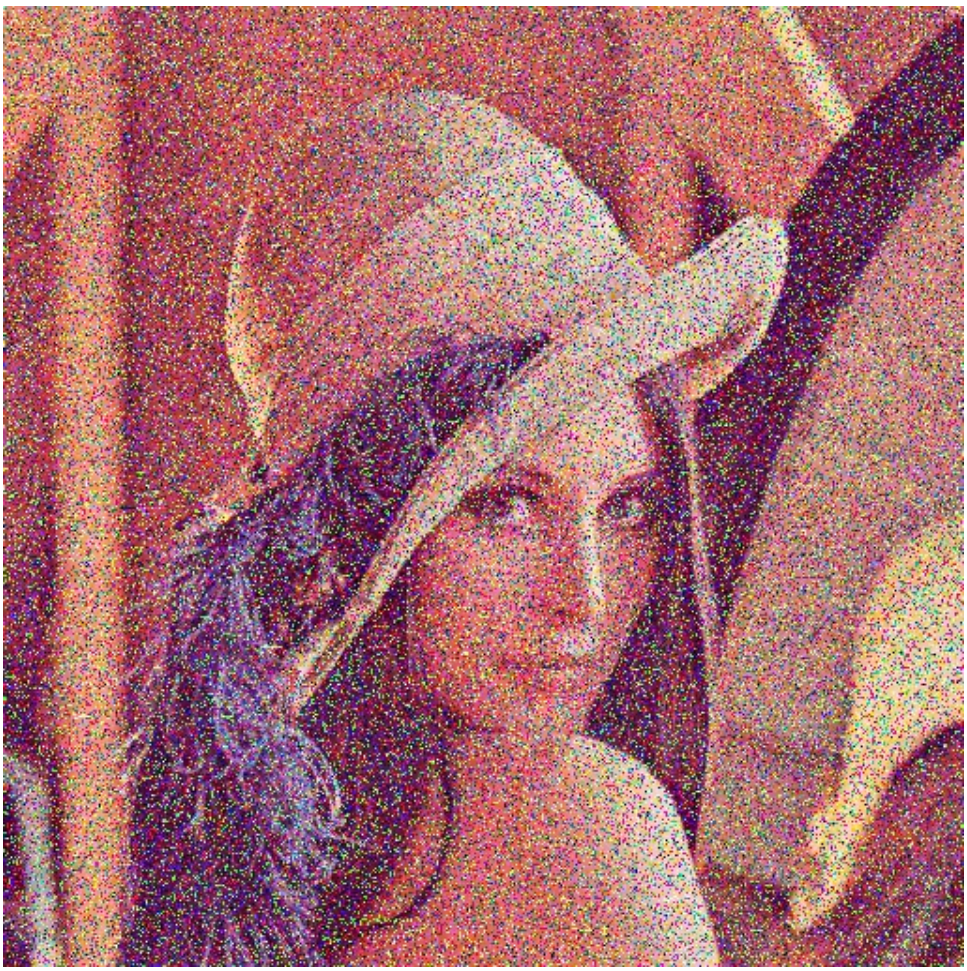


Figure 9.3: 9

```
10
11 //Gaussian
12 lenaNgaussian = imnoise(lenna,'gaussian');
13 imwrite(lenaNgaussian,'lenaNgaussian.jpeg');
14
15 //Speckle
16 lenaNspeckle = imnoise(lenna,'speckle');
17 imwrite(lenaNspeckle,'lenaNspeckle.jpeg');
18
19 //Salt & Pepper
20 d=0.25  //drop out noise
21 lenaNsalpep = imnoise(lenna,'salt & pepper',d);
22 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`

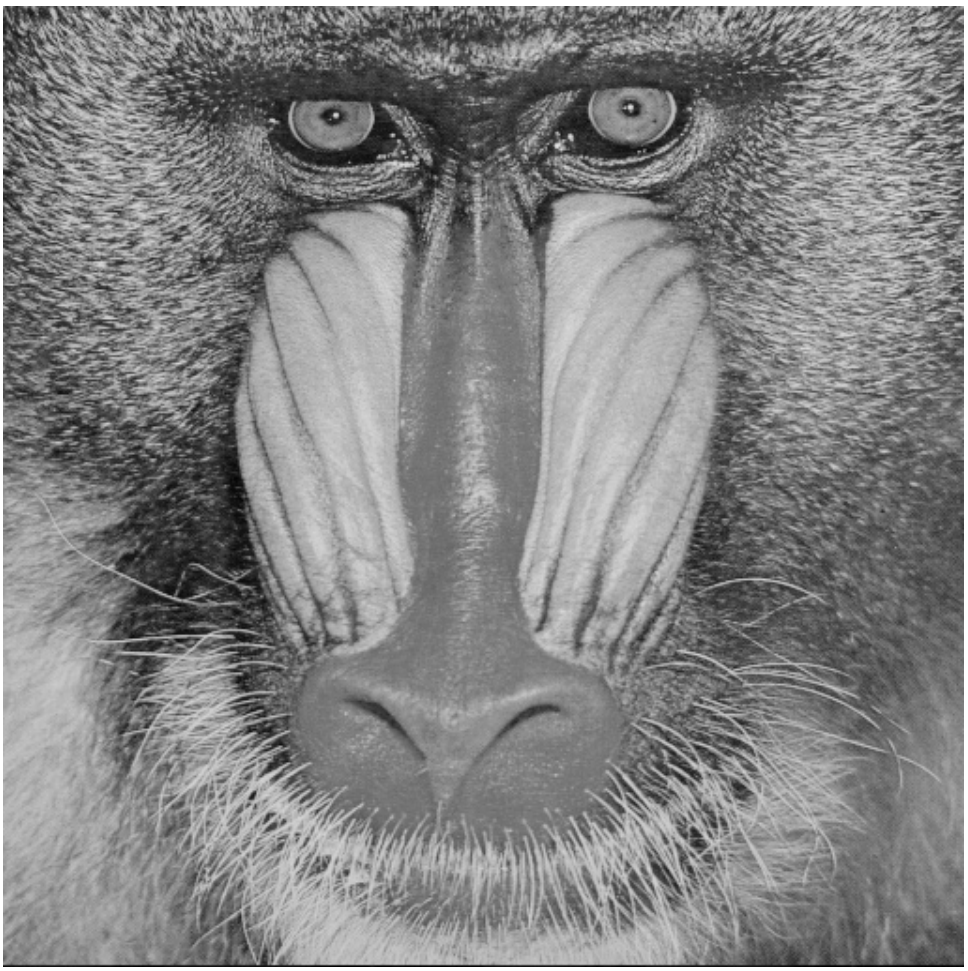


Figure 10.1: 10



Figure 10.2: 10



Figure 10.3: 10

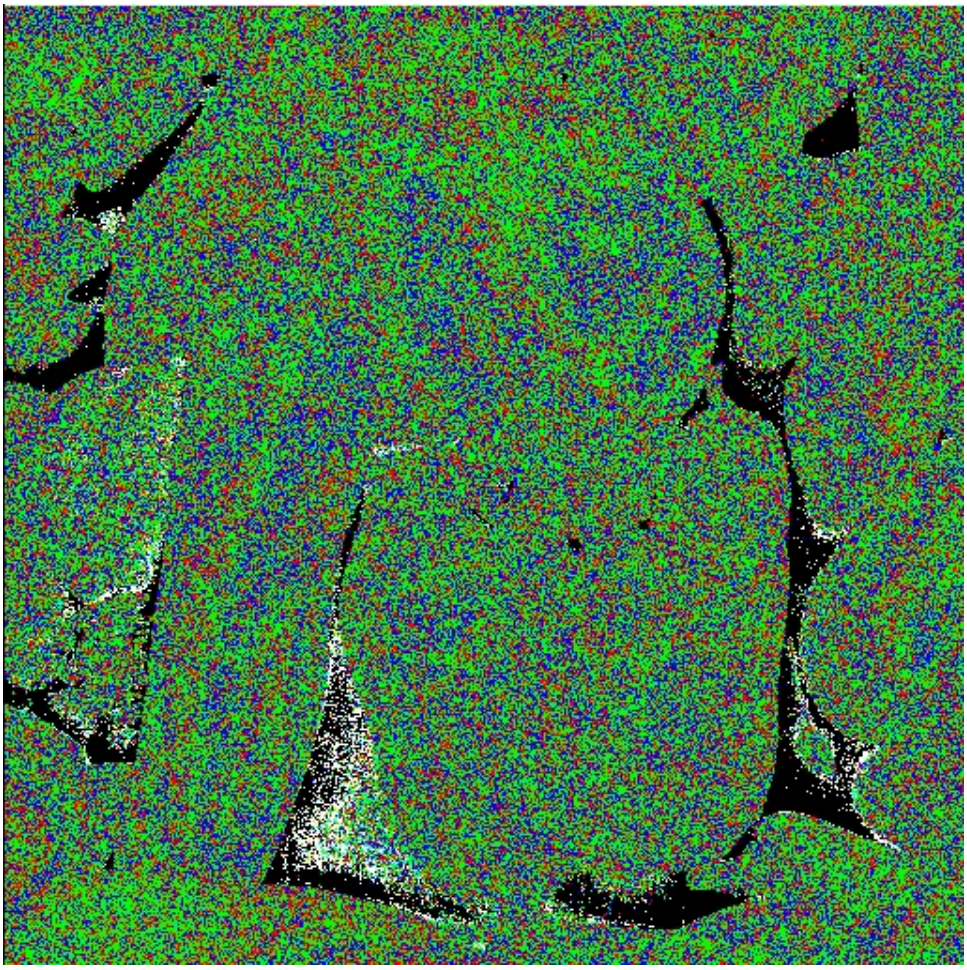


Figure 10.4: 10

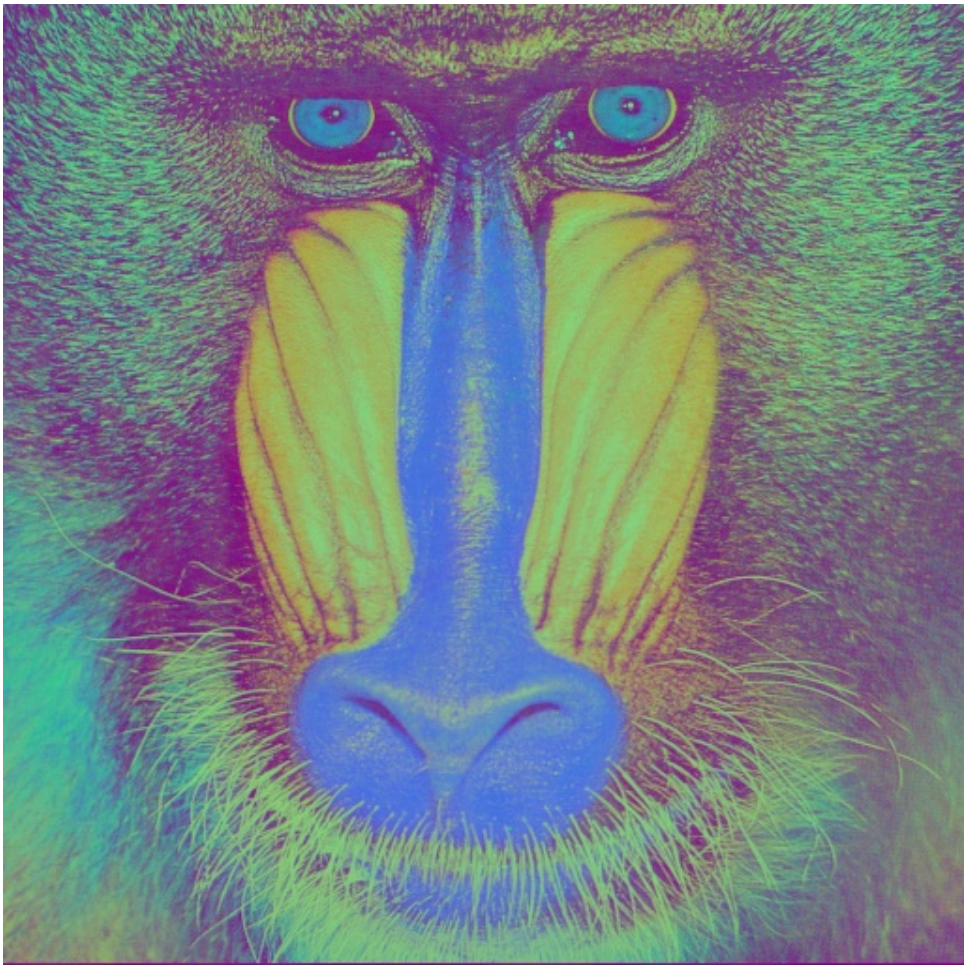


Figure 10.5: 10



Figure 10.6: 10

Scilab code Solution 10.10 10

```
1 //Demonstrate the various Image Conversions
2 //Scilab 5.4.1
3 //Windows 10
```

```
4  //Requires SIVP, IPD toolboxes
5
6  clear;
7  clc;
8
9  //RBG to Gray scale
10 baboon = imread('baboon.png');
11 babgray = rgb2gray(baboon);
12 imwrite(babgray,'babgray.jpeg');
13
14 //RBG to Binary
15 lena = imread('Lenna.png');
16 lenabw = im2bw(lena,0.5);
17 imwrite(lenabw,'lenabw.jpeg');
18
19 //RBG to HSV
20 cameraman = imread('cameraman.jpg');
21 cameramanhsv = rgb2hsv(cameraman);
22 imwrite(cameramanhsv,'cameramanhsv.jpeg');
23
24 //HSV to RGB
25 peppers = imread('peppers.png');
26 peppersrgb = hsv2rgb(peppers);
27 imwrite(peppersrgb,'peppersrgb.jpeg');
28
29 //RBG to YCbCr
30 baboon = imread('baboon.png');
31 baboonycbcr = rgb2ycbcr(baboon);
32 imwrite(baboonycbcr,'baboonycbcr.jpeg');
33
34 //YCbCr to RGB
35 ararauna = imread('ararauna.png');
36 araraunargb = ycbcr2rgb(ararauna);
37 imwrite(araraunargb,'araraunargb.jpeg');
```

Experiment: 11

Demonstrate Spatial Domain Processing

check Appendix AP 3 for dependency:

Lenna.png

Scilab code Solution 11.11 11

```
1 //Demonstrate Spatial Domain Processing
2 //Scilab 5.4.1
3 //Windows 10
4 //Requires SIVP, IPD toolboxes
5
6 clear;
7 clc;
8
9 lenna=imread('Lenna.png');
```



Figure 11.1: 11



Figure 11.2: 11



Figure 11.3: 11

```
10
11 //Sobel
12 h = fspecial('sobel');
13 lenaSobel = imfilter(lenna,h)
14 imwrite(lenaSobel,'lenaSobel.jpeg');
15
16 //Prewitt
17 h = fspecial('prewitt');
18 lenaPrewitt = imfilter(lenna,h)
19 imwrite(lenaPrewitt,'lenaPrewitt.jpeg');
20
21 //Laplacian
22 h = fspecial('laplacian');
23 lenaLaplacian = imfilter(lenna,h)
24 imwrite(lenaLaplacian,'lenaLaplacian.jpeg');
```

Experiment: 12

Motion blur of an image

check Appendix AP 4 for dependency:

ararauna.png

Scilab code Solution 12.12 12

```
1  //Motion blur of an image
2  //Scilab 5.4.1
3  //Windows 10
4  //Requires SIVP, IPD toolboxes
5
6  clear;
7  clc;
8
9  a=imread('ararauna.png');
10
11 //filter coefficients of fspecial('motion',10,25)
12 H =[0,0,0,0,0,0,0,0.0032,0.0449,0.0865,0.0072;...
13 0,0,0,0,0,0.0092,0.0509,0.0925,0.0629,0.0213,0;...
14 0,0,0,0.0152,0.0569,0.0985,0.0569,0.0152,0,0,0;...
15 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

```
1  //Trimmed Average Filter
2  //Scilab 5.4.1
3  //Windows 10
4  //Requires SIVP, IPD toolboxes
5
6  clear;
7  clc;
8
9  c=imread('Lenna.png');
10 s = 1; //s denotes the number of values to be left
       in the end
11 r = 1;
12 N = 9; //3x3 window
```



Figure 13.1: 13



Figure 13.2: 13

```
13 a = double(imnoise(c,'gaussian'));
14 [m,n] = size(a);
15 b = zeros(m,n);
16 for i= 2:m-1
17     for j = 2:n-1
18         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),...
19                a(i-1,j+1),a(i-1,j+1),a(i+1,j+1)];
20         sorted_mat = gsort(mat);
21         Sum=0;
22         for k=r+s:(N-s)
23             Sum = Sum+mat(k);
24         end
25         b(i,j)= Sum/(N-r-s);
26     end
27 end
28 a = uint8(a);
29 b = uint8(b);
30 //figure
31 //imshow(c)
32 //title('Original Image')
33
34 imwrite(a,'noisyimage.jpeg')
35 imwrite(b,'TrimmedAverageFilteredImage.jpeg')
```

Experiment: 14

Determine image negative

check Appendix AP 2 for dependency:

peppers.png

Scilab code Solution 14.14 14

```
1  //Determine image negative
2  //Scilab 5.4.1
3  //Windows 10
4  //Requires SIVP, IPD toolboxes
5
6  clear;
7  clc;
8
9  a=imread('peppers.png');
10 k = 255-double(a);
11 k = uint8(k);
12 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

```
1 //Image operations to perform clockwise and anti-
    clockwise operations
2 //Scilab 5.4.1
3 //Windows 10
4 //Requires SIVP, IPD toolboxes
5
6 clear;
7 clc;
8
```



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



peppers



Lenna



arara-rauna



baboon