



INDIAN INSTITUTE OF
INFORMATION TECHNOLOGY,
NAGPUR

Digital Image Processing
(ECL-415)

Final Report

Submitted By :
Sarthak Babra (BT19ECE028)

Semester 6
Electronics and Communication Engineering Dept.

Submitted To :
Dr. Tapan Jain
Course Instructor

INDEX

1. Assignment - 1 Converting colour image to rgb gray layers, gray scale image and binary image. Also, performing pixel addition on gray scale image.
2. Assignment - 2 Performing subtraction operation on gray scale image.
3. Assignment - 3 Performing logical operations - OR, AND, NOR, NAND, XOR AND XNOR.
4. Assignment - 4 Extracting r,g,b colour layers from a colour image.
5. Assignment - 5 Histogram generation and equalisation.
6. Assignment - 6 Contrast manipulation of colour and gray scale images.
7. Assignment - 7 Bit Plane Slicing
8. Assignment - 8 Shannon-Fano Coding
9. Assignment - 9 Huffman Coding
10. Assignment - 10 Arithmetic Coding
11. Assignment - 11 Converting a colour image into a gray scale image and performing edge detection.

Assignment - 1

Converting colour image to rgb gray layers, gray scale image and binary image. Also, performing pixel addition on gray scale image.

Code

```

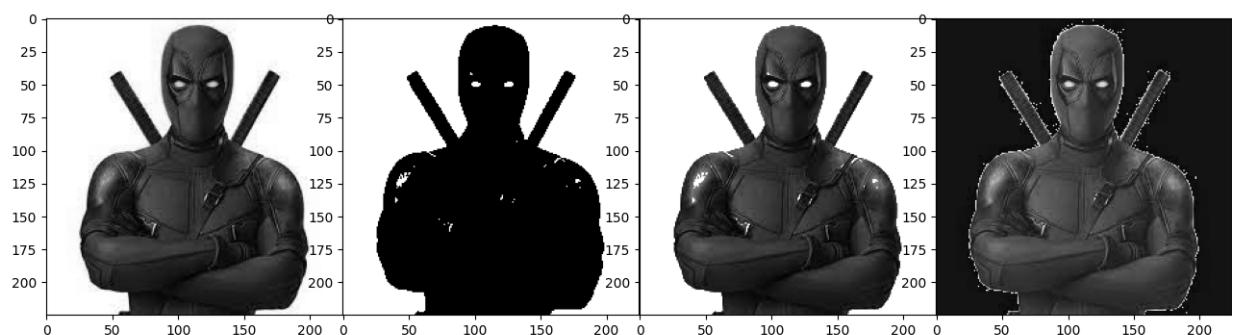
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import matplotlib.image as mpimg
4
5 def rgb_to_gray(img):
6     grayImage = np.zeros(img.shape)
7     R = np.array(img[:, :, 0])
8     G = np.array(img[:, :, 1])
9     B = np.array(img[:, :, 2])
10
11    R = (R *.299)
12    G = (G *.587)
13    B = (B *.114)
14
15    Avg = (R+G+B)
16    grayImage = img.copy()
17
18    for i in range(3):
19        grayImage[:, :, i] = Avg
20
21    return grayImage
22
23 image = mpimg.imread("yoyoy.jpg")
24 grayImage = rgb_to_gray(image)
25
26 img = np.array(grayImage)
27 binarr = np.where(img>128, 255, 0)
28
29 img3 = np.add(grayImage, binarr)
30
31 img4 = np.add(20,grayImage)
32
33
34 #subplot(r,c) provide the no. of rows and columns
35 f, axarr = plt.subplots(1,4)
36 axarr[0].imshow(grayImage)
37 axarr[1].imshow(binarr, cmap='gray')
38 axarr[2].imshow(img3, cmap='gray')
39 axarr[3].imshow(img4, cmap='gray')
```

```
40 plt.show()
```

input



output



Assignment - 2

Performing subtraction operation on gray scale image.

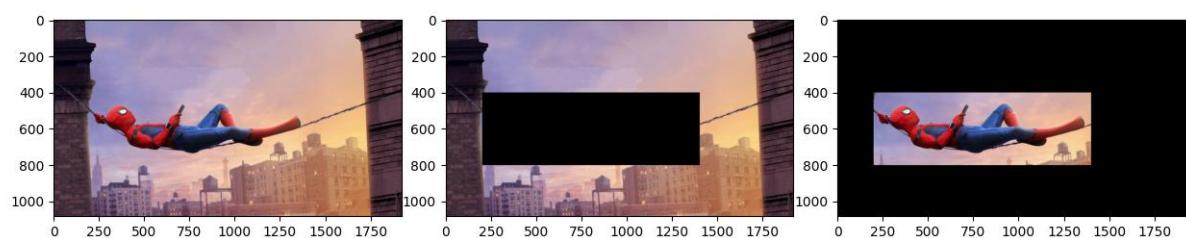
Code

```
 1 import matplotlib.pyplot as plt
 2 import matplotlib.image as mpimg
 3
 4 #read images
 5 img1 = mpimg.imread('input.jpg')
 6 img2 = mpimg.imread('input.jpg')
 7
 8 for i in range(0,400):
 9     for j in range(0,1200):
10         for k in range(0,3):
11             img2[i+400,j+200,k] = 0
12
13 #obtain the final image by subtracting the original and edited ...
14 #image
14 img3= img1 - img2
15 #subplot(r,c) provide the no. of rows and columns
16 f, axarr = plt.subplots(1,3)
17 axarr[0].imshow(img1)
18 axarr[1].imshow(img2)
19 axarr[2].imshow(img3)
20 plt.show()
```

input



output



Assignment - 3

Performing logical operations - OR, AND, NOR, NAND, XOR AND XNOR.

Code

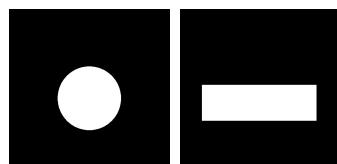
```

1 from matplotlib import pyplot as plt
2 import matplotlib.image as mpimg
3 import numpy as np
4 import operator
5
6 def image_to_binary(img):
7     #code below is to convert image to binary
8
9     R = img[:, :, 0]
10    G = img[:, :, 1]
11    B = img[:, :, 2]
12    binary = (R+G+B)/3
13
14    for i in range(500):
15        for j in range(500):
16            if binary[i, j]>10:
17                binary[i, j]=1
18            else:
19                binary[i, j]=0
20
21    return binary
22
23
24 #read images
25 img_circle = mpimg.imread('circle.jpg')
26 img_rectangle = mpimg.imread('rectangle.jpg')
27
28 # now calling function to convert image in binary
29 binary_circle = image_to_binary(img_circle)
30 binary_rectangle = image_to_binary(img_rectangle)
31
32 #showing circle
33 plt.imshow(binary_circle, cmap='gray')
34 plt.title('Circle', fontsize=24)
35 plt.show()
36 #showing rectangle
37 plt.imshow(binary_rectangle, cmap='gray')
38 plt.title('Rectangle', fontsize=24)
39 plt.show()
40 #logical and

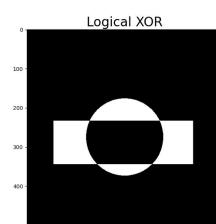
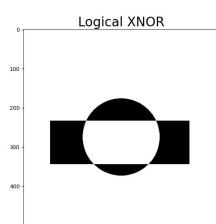
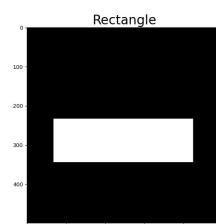
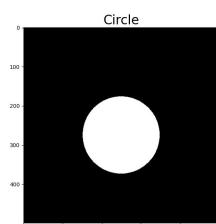
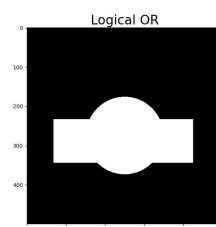
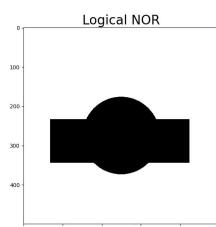
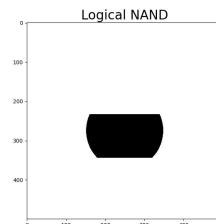
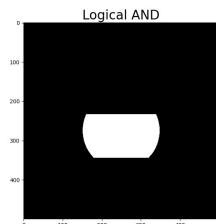
```

```
41 img_and = binary_circle.astype(int) & binary_rectangle.astype(int ...  
42 plt.imshow(img_and, cmap='gray')  
43 plt.title('Logical AND', fontsize=24)  
44 plt.show()  
45  
46 #logical or  
47 img_or = binary_circle.astype(int) | binary_rectangle.astype(int ...  
48 plt.imshow(img_or, cmap='gray')  
49 plt.title('Logical OR', fontsize=24)  
50 plt.show()  
51  
52 #logical xor  
53 img_xor = binary_circle.astype(int) ^ binary_rectangle.astype(int ...  
54 plt.imshow(img_xor, cmap='gray')  
55 plt.title('Logical XOR', fontsize=24)  
56 plt.show()  
57  
58 #logical nand  
59 img_nand = ~img_and  
60 plt.imshow(img_nand, cmap='gray')  
61 plt.title('Logical NAND', fontsize=24)  
62 plt.show()  
63  
64 #logical nor  
65 img_nor = ~img_or  
66 plt.imshow(img_nor, cmap='gray')  
67 plt.title('Logical NOR', fontsize=24)  
68 plt.show()  
69  
70 #logical xnor  
71 img_xnor = ~img_xor  
72 plt.imshow(img_xnor, cmap='gray')  
73 plt.title('Logical XNOR', fontsize=24)  
74 plt.show()
```

input



output



Assignment - 4

Extracting r,g,b colour layers from a colour image.

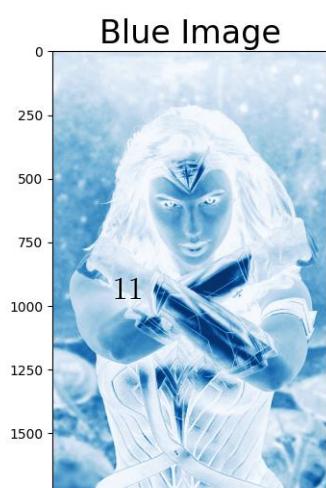
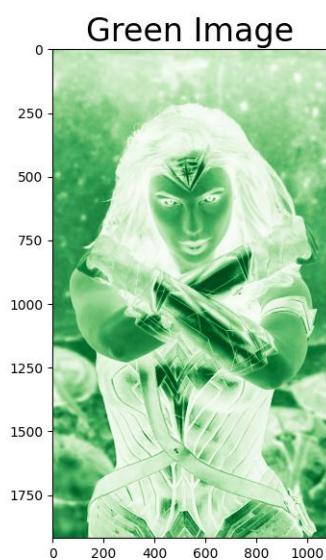
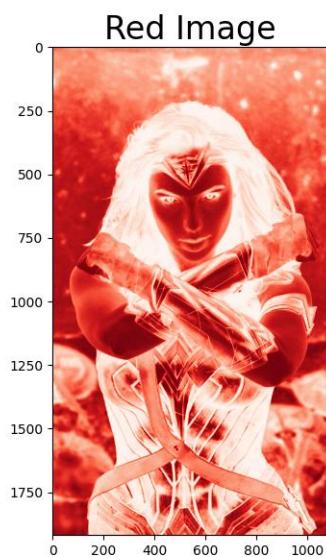
Code

```
 1 from matplotlib import pyplot as plt
 2 import matplotlib.image as mpimg
 3
 4 #read image
 5 img = mpimg.imread('input.jpg')
 6
 7 #read the red, green and blue colourespaces differently
 8 R = img[:, :, 0]
 9 G = img[:, :, 1]
10 B = img[:, :, 2]
11
12 #display the red image
13 plt.imshow(R, cmap='Reds')
14 plt.title('Red Image', fontsize=24)
15 plt.show()
16
17 #display the green image
18 plt.imshow(G, cmap='Greens')
19 plt.title('Green Image', fontsize=24)
20 plt.show()
21
22 #display the blue image
23 plt.imshow(B, cmap='Blues')
24 plt.title('Blue Image', fontsize=24)
25 plt.show()
```

input



output



Assignment - 5

Histogram generation and equalisation.

Code

```

1 import numpy as np
2
3 def imhist(im):
4     # calculates normalized histogram of an image
5     m, n = im.shape
6     h = [0.0] * 256
7     for i in range(m):
8         for j in range(n):
9             h[im[i, j]]+=1
10    return np.array(h)/(m*n)
11
12 def cumsum(h):
13     # finds cumulative sum of a numpy array , list
14     return [sum(h[:i+1]) for i in range(len(h))]
15
16 def histeq(im):
17     #calculate Histogram
18     h = imhist(im)
19     cdf = np.array(cumsum(h)) #cumulative distribution function
20     sk = np.uint8(255 * cdf) #finding transfer function values
21     s1, s2 = im.shape
22     Y = np.zeros_like(im)
23     # applying transferred values for each pixels
24     for i in range(0, s1):
25         for j in range(0, s2):
26             Y[i, j] = sk[im[i, j]]
27     H = imhist(Y)
28     #return transformed image, original and new istogram ,
29     # and transform function
30     return Y, h, H, sk

```

```

1 import pylab as plt
2 import matplotlib.image as mpimg
3 import numpy as np
4 # load image to numpy arrayb
5 # matplotlib 1.3.1 only supports png images
6 # use scipy or PIL for other formats
7 img = np.uint8(mpimg.imread('input.jpg')*255.0)
8 # convert to grayscale

```

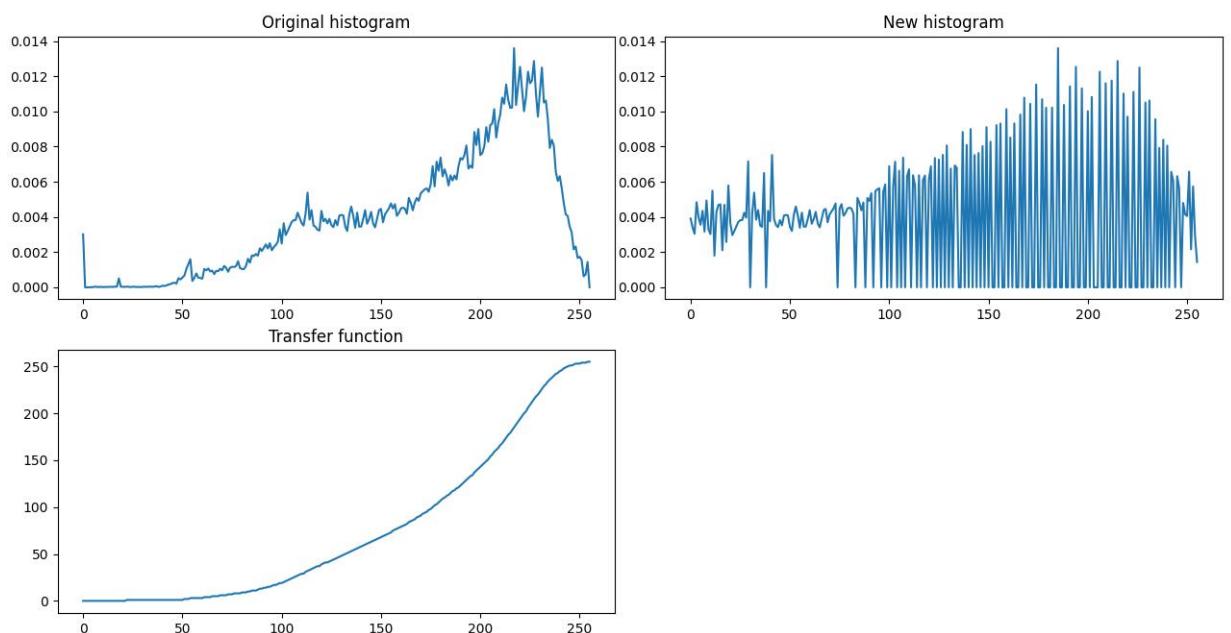
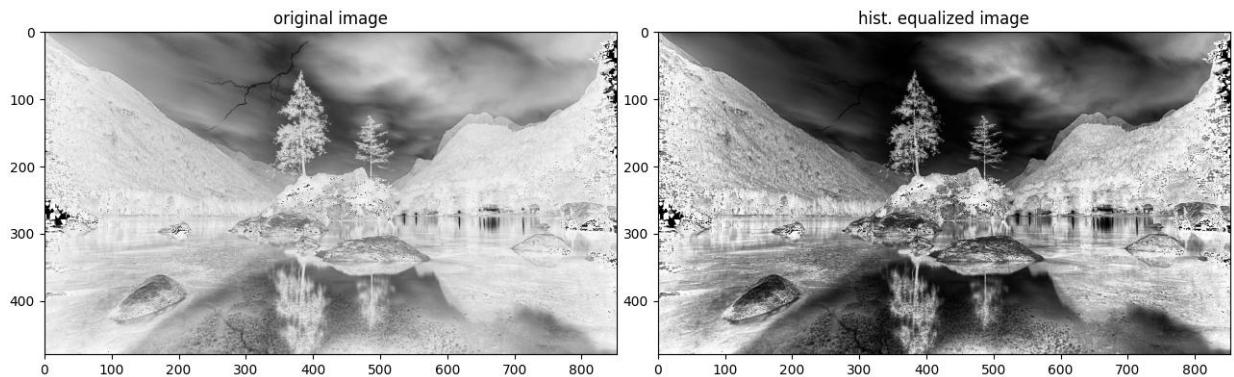
```

 9 # do for individual channels R, G, B, A for nongrayscale images
10
11 img = np.uint8((0.2126* img[:, :, 0]) + \
12     np.uint8(0.7152 * img[:, :, 1]) +\
13     np.uint8(0.0722 * img[:, :, 2]))
14
15 # use hist module from hist.py to perform histogram equalization
16 from hist import histeq
17 new_img, h, new_h, sk = histeq(img)
18
19 # show old and new image
20 # show original image
21 plt.subplot(121)
22 plt.imshow(img)
23 plt.title('original image')
24 plt.set_cmap('gray')
25 # show original image
26 plt.subplot(122)
27 plt.imshow(new_img)
28 plt.title('hist. equalized image')
29 plt.set_cmap('gray')
30 plt.show()
31
32 # plot histograms and transfer function
33 fig = plt.figure()
34 fig.add_subplot(221)
35 plt.plot(h)
36 plt.title('Original histogram') # original histogram
37
38 fig.add_subplot(222)
39 plt.plot(new_h)
40 plt.title('New histogram') #hist of eqalauized image
41
42 fig.add_subplot(223)
43 plt.plot(sk)
44 plt.title('Transfer function') #transfer function
45
46 plt.show()

```

input

output



Assignment - 6

Contrast manipulation of colour and gray scale image.

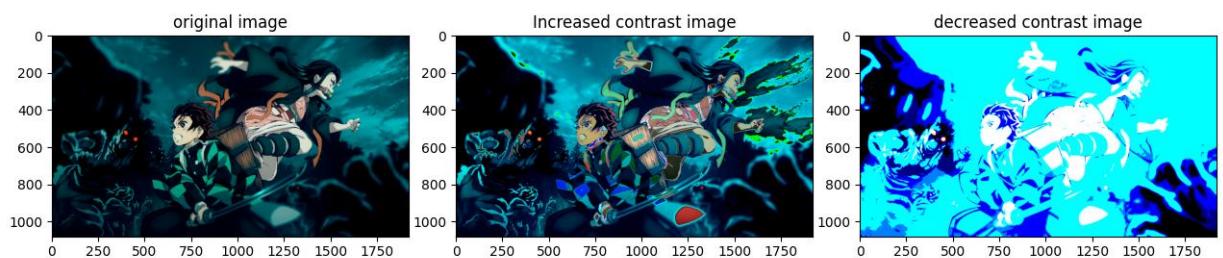
Code

```
 1 from matplotlib import pyplot as plt
 2 import matplotlib.image as mpimg
 3 import numpy as np
 4
 5 #read image
 6 img1 = mpimg.imread('input.jpg')
 7
 8 #Increased contrast
 9 img2 = img1*2
10
11 #decreased contrast
12 img3 = img1/2
13
14 #subplot(r,c) provide the no. of rows and columns
15 f, axarr = plt.subplots(1,3)
16 axarr[0].imshow(img1)
17 axarr[0].set_title('original image')
18 axarr[1].imshow(img2)
19 axarr[1].set_title('Increased contrast image')
20 axarr[2].imshow(img3)
21 axarr[2].set_title('decreased contrast image')
22 plt.show()
```

input



output



Assignment - 7

Bit Plane Slicing.

Code

```

1 clear all; close all; clc;
2
3 % Reading the image
4 Img = imread('Image.jpg');
5 % Converting the image into gray scale
6 gray = rgb2gray(Img);
7 % Extracting the size of the image
8 [m,n] = size(gray);
9 % Converting the unsigned integer to a double type
10 gray1 = double(gray);
11 % Decimal to binary converter
12 bin = de2bi(gray1);
13
14 % Extracting LSB plane
15 plane1 = bin(:,1);
16 p1 = reshape(plane1,m,n);
17 % Extracting 2 bit (from the left) plane
18 plane2 = bin(:,2);
19 p2 = reshape(plane2,m,n);
20 % Extracting 3 bit (from the left) plane
21 plane3 = bin(:,3);
22 p3 = reshape(plane3,m,n);
23 % Extracting 4 bit (from the left) plane
24 plane4 = bin(:,4);
25 p4 = reshape(plane4,m,n);
26 % Extracting 5 bit (from the left) plane
27 plane5 = bin(:,5);
28 p5 = reshape(plane5,m,n);
29 % Extracting 6 bit (from the left) plane
30 plane6 = bin(:,6);
31 p6 = reshape(plane6,m,n);
32 % Extracting 7 bit (from the left) plane
33 plane7 = bin(:,7);
34 p7 = reshape(plane7,m,n);
35 % Extracting MSB plane
36 plane8 = bin(:,8);
37 p8 = reshape(plane8,m,n);
38
39 % Plotting all the bit plane results
40 subplot(2,5,1);
41 imshow(Img); title ('Original Image');

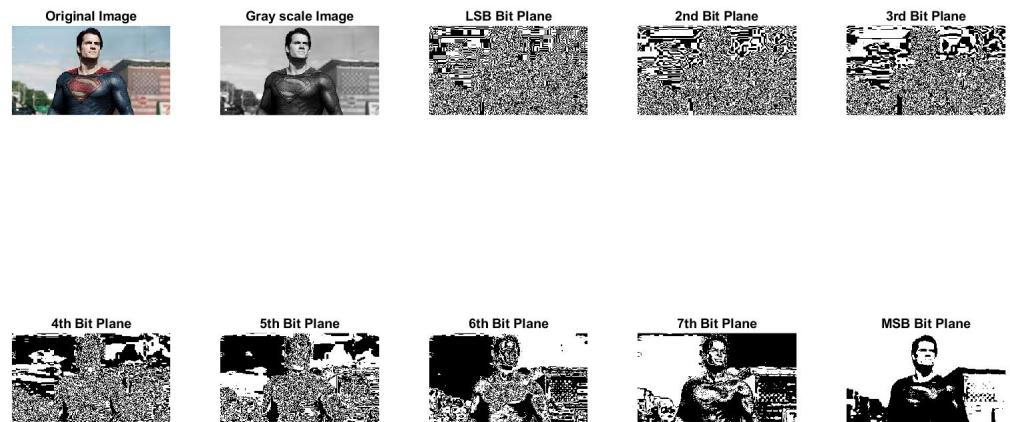
```

```
42 subplot(2,5,2);
43 imshow(gray); title('Gray scale Image');
44 subplot(2,5,3);
45 imshow(p1); title('LSB Bit Plane');
46 subplot(2,5,4);
47 imshow(p2); title('2nd Bit Plane');
48 subplot(2,5,5);
49 imshow(p3); title('3rd Bit Plane');
50 subplot(2,5,6);
51 imshow(p4); title('4th Bit Plane');
52 subplot(2,5,7);
53 imshow(p5); title('5th Bit Plane');
54 subplot(2,5,8);
55 imshow(p6); title('6th Bit Plane');
56 subplot(2,5,9);
57 imshow(p7); title('7th Bit Plane');
58 subplot(2,5,10);
59 imshow(p8); title('MSB Bit Plane');
```

input



output



Assignment - 8

Shannon-Fano Coding.

Code

```

1 % Shannon–Fano Coding
2
3 clc;
4 clear all;
5 close all;
6
7 disp('Enter the probabilities:');
8
9 ss=[0.2 0.2 0.12 0.08 0.08 0.08 0.04];
10
11 %outputs = string of codewords, average codeword length
12 ss=ss./sum(ss); %if occurrences are inputted, probabilities are ...
13 %gained
13 ss=sort(ss, 'descend'); %the probabilities are sorted in ...
14 %descending order
14
15 %siling=ceil(log2(1/ss(1))); %initial length is computed
16
17 siling=log2(1/ss(1)); %initial length is computed
18 siling=round(siling, 1, 'significant');
19
20 sf=0;
21 fano=0;
22 %initializations for Pk
23 n=1;Hx=0; %initializations for entropy H(X)
24
25 for i=1:length(ss)
26     Hx=Hx+ ss(i)*log2(1/ss(i)); %solving for entropy
27 end
28
29 for k=1:length(ss)
30     info(k)=-(log2(ss(k))); %Information
31 end
32
33 for j=1:length(ss)-1
34     fano=fano+ss(j);
35     sf=[sf 0]+[zeros(1,j) fano]; %solving for Information for ...
36     %every codeword
36     siling=[siling 0]+[zeros(1,j) ceil(log2(1/ss(j+1)))]; %solving ...
37     %for length every codeword
37

```

```

38 end
39
40 for r=1:length(sf)
41 esf=sf(r);
42 for p=1:siling(r)
43 esf=mod(esf,1)*2;
44 h(p)=esf-mod(esf,1); %converting Pk into a binary number
45 end
46 hh(r)=h(1)*10^(siling(r)-1); %initializtion for making the ...
47 binary a whole number
48 for t=2:siling(r)
49 hh(r)=hh(r)+h(t)*10^(siling(r)-t); %making the binary ...
50 a whole number
51 end
52 %e.g. 0.1101 ==> ...
53 1101
54 end
55
56 c={'0','1'};
57 disp('Codeword');
58 for i=1:length(hh)
59 u=1; %converting the ...
60 codes into a string
61 for t=siling(i):-1:1
62 f=floor(hh(i)/10^(t-1)); %1001 ==>1 (getting...
63 the first highest unit of a number)
64 hh(i)=mod(hh(i),10^(t-1)); %1001 ==>001(...
65 eliminating the first highest unit of a number)
66 if f==1
67 if u==1
68 d=c{2}; %conversion part (...
69 num(1001) to str(1001))
70 else
71 d=[d c{2}];
72 end
73 else
74 if u==1
75 d=c{1};
76 else
77 d=[d c{1}];
78 end
79 codex{i,:}={d};
80 u=u+1;
81 end
82 disp([d])
83 end
84
85 tao=siling(1)*ss(1); %initialization for codeword length

```

```

80 for u=1:length(ss)-1 %computing for codeword length
81     tao=tao+sileng(u+1)*ss(u+1);
82 end
83
84 T=tao/n; %computing for average codeword length
85 B=[flipud(rot90(ss)), flipud(rot90(sileng)), flipud(rot90(info))];
86 disp(['Probability' , ' Length' , ' Information'])
87 disp(B)
88 disp(['Entropy H(X) = ' , num2str(Hx) , ' bits/symbol'])
89 disp(['Average length,L = ' , num2str(T) , ' bits/symbol'])
90 eff=((Hx/T)*100); %Coding efficiency
91 disp(['Efficiency=' , num2str(eff) , '%'])
92 redu=100-eff; %Redundancy
93 disp(['Redundancy=' , num2str(redu) , '%'])

```

input

```

7 -      disp('Enter the probabilities:');
8 -
9 -      ss=[0.2| 0.2 0.12 0.08 0.08 0.08 0.04];
n

```

output

Command Window

```

Enter the probabilities:
Codeword
00
01
100
1010
1100
1101
1110
11110
Probability    Length    Information
    0.2500    2.0000    2.0000
    0.2500    2.0000    2.0000
    0.1500    3.0000    2.7370
    0.1000    4.0000    3.3219
    0.1000    4.0000    3.3219
    0.1000    4.0000    3.3219
    0.0500    5.0000    4.3219

Entropy H(X) = 2.6232bits/symbol
Average length,L = 2.9bits/symbol
Efficiency=90.4559%
Redundancy=9.5441%
fx >>

```

Assignment - 9

Huffman Coding.

Code

```

1 function [ code , compression]=huffman(p)
2 %HUFFMAN
3 %HUFFMAN CODER FOR V5
4 % Format [CODE,COMPRESSION]=HUFFMAN(P)
5 %
6 % P is the probability (or number of occurences) of each alphabet...
7 % symbol
8 % CODE gives the huffman code in a string format of ones and ...
9 % zeros
10 % COMPRESSION gives the compression rate
11 p=p(:)/sum(p); %normalises probabilities
12 c=huff5(p);
13 code=char(getcodes(c,length(p)));
14 compression=ceil(log(length(p))/log(2))/(sum(code' ~= ' ')*p);
15 %
16 function c=huff5(p);
17 % HUFF5 Creates Huffman Tree
18 % Simulates a tree structure using a nested cell structure
19 % P is a vector with the probability (number of occurences)
20 % of each alphabet symbol
21 % C is the Huffman tree.
22 c=cell(length(p),1); % Generate cell structure
23 for i=1:length(p) % fill cell structure with 1,2,3...n
24 c{i}=i; % (n=number of symbols in alphabet)
25 end
26 while size(c)-2 % Repeat till only two branches
27 [p, i]=sort(p); % Sort to ascending probabilities
28 c=c(i); % Reorder tree.
29 c{2}={c{1},c{2}};c{1}=[]; % join branch 1 to 2 and prune 1
30 p(2)=p(1)+p(2);p(1)=[]; % Merge Probabilities
31 end
32 %
33 function y= getcodes(a,n)
34 % Y=GETCODES(A,N)
35 % Pulls out Huffman Codes for V5
36 % a is the nested cell structure created by huffcode5
37 % n is the number of symbols
38 global y
39 y=cell(n,1);
40 getcodes2(a,[])
41 %

```

```
40 function getcodes2(a,dum)
41 % GETCODES(A,DUM)
42 %getcodes2
43 % called by getcodes
44 % uses Recursion to pull out codes
45 global y
46 if isa(a,'cell')
47     getcodes2(a{1},[dum 0]);
48     getcodes2(a{2},[dum 1]);
49 else
50     y{a}=setstr(48+dum);
51 end
```

output

```
>> p = [0.4 0.3 0.2 0.1];
>> huffman(p)

ans =

4×3 char array

    '1 '
    '01 '
    '001'
    '000'
```

Assignment - 10

Converting a colour image into a gray scale image and performing edge detection

Code

```

1 clc;
2 clear all;
3 close all;
4
5 Image=(imread( 'Image.jpg' ));
6
7 % Converting colour image to grayscale image
8 r = Image(:,:,1);
9 g = Image(:,:,2);
10 b = Image(:,:,3);
11 gray = double(uint8(double(r/3)+double(g/3)+double(b/3)));
12
13 figure;
14 subplot(2,1,1);
15 imshow(Image); title('Original Image');
16
17 In=gray;
18 mask1=[1, 0, -1;1, 0, -1;1, 0, -1];
19 mask2=[1, 1, 1;0, 0, 0;-1, -1, -1];
20 mask3=[0, -1, -1;1, 0, -1;1, 1, 0];
21 mask4=[1, 1, 0;1, 0, -1;0, -1, -1];
22
23 mask1=flipud(mask1);
24 mask1=fliplr(mask1);
25 mask2=flipud(mask2);
26 mask2=fliplr(mask2);
27 mask3=flipud(mask3);
28 mask3=fliplr(mask3);
29 mask4=flipud(mask4);
30 mask4=fliplr(mask4);
31
32 for i=2:size(gray, 1)-1
33     for j=2:size(gray, 2)-1
34         neighbour_matrix1=mask1.*In(i-1:i+1, j-1:j+1);
35         avg_value1=sum(neighbour_matrix1 (:));
36
37         neighbour_matrix2=mask2.*In(i-1:i+1, j-1:j+1);
38         avg_value2=sum(neighbour_matrix2 (:));
39
40         neighbour_matrix3=mask3.*In(i-1:i+1, j-1:j+1);
41         avg_value3=sum(neighbour_matrix3 (:));

```

```
42 neighbour_matrix4=mask4.*In(i-1:i+1, j-1:j+1);  
43 avg_value4=sum(neighbour_matrix4(:));  
44  
45 %using max function for detection of final edges  
46 I(i, j)=max([avg_value1, avg_value2, avg_value3, avg_value4])...  
47 ;  
48  
49 end  
50 end  
51  
52 subplot(2,1,2);  
53 imshow(uint8(I));  
54 title('Edge Detected Image');
```

input



output

