(Gonzalez 3rd edition)

1. Problem 6.5 (10%)

   At the center point

$$\frac{1}{2}R + \frac{1}{2}B + G = \frac{1}{2}(R + B + G) + \frac{1}{2}G = midgray + \frac{1}{2}G$$

   which looks to a viewer like pure green with a boost in the intensity due to additive gray component.

2. Problem 6.8 (10%)

   (a) All pixel values in the Red image are 255. In the Green image, the first column is all 0's; the second column all 1's; and so on until the last column, which is composed of all 255's. In the Blue image, the first row is all 255's; the second row all 254's, and so on until the last row which is composed of all 0's.

   (b) Let the axis numbering be the same as in Fig. 6.7 in the book. Then: (0,0,0) = white, (1,1,1) = black, (1,0,0) = cyan, (1,1,0) = blue, (1,0,1) = green, (0,1,1) = red, (0,0,1) = yellow, (0,1,0) = magenta.

   (c) The ones that do not contain the black or white point are fully saturated. The others decrease in saturation from the corners toward the black or white point.

3. Problem 6.11 (10%)

   (a) The purest green is 00FF00, which corresponding to cell (7, 18)

   (b) The purest blue is 0000FF, which corresponding to cell (12, 13)

4. Problem 6.13 (10%)

   With reference to the HSI color circle in Fig. 6.14(b), deep purple is found at approximately 270°. To generate a color rectangle with the properties required in the problem statement, we choose a fixed intensity I, and maximum saturation (these are spectrum colors, which are supposed to be fully saturated), S. The first column in the rectangle uses these two values and a hue of 270°. The next column (and all subsequent columns) would use the same values of I and S, but the hue would be decreased to 269°, and so on all the way down to a hue of 0°, which corresponds to red. If the image is limited to 8 bits, then we can only have 256 variations in hue in the range from 270° down to 0°, which will require a different uniform spacing than one degree increments or, alternatively, starting at a 255° and proceed in increments of 1, but this would leave out most of the purple. If we have more than eight bits, then the increments can be smaller. Longer strips also can be made by duplicating column values.

5. Problem 6.16 (10%)

   (a) It is given that the colors in Fig. 6.16(a) are primary spectrum colors. It also is given that the gray-level images in the problem statement are 8-bit images. The latter condition means that hue (angle) can only be divided into a maximum number of 256 values. Because hue values are represented in the interval from 0° to 360° this means that for an 8-bit image the increments between contiguous hue values are now 360/255. Another way of looking at this is that the entire [0, 360] hue scale is compressed to the range [0, 255]. Thus, for example,

yellow (the first primary color we encounter), which is 60◦ now becomes 43 (the closest integer) in the integer scale of the 8-bit image shown in the problem statement. Similarly, green, which is 120◦ becomes 85 in this image. From this we easily compute the values of the other two regions as being 170 and 213. The region in the middle is pure white [equal proportions of red green and blue in Fig. 6.61(a)] so its hue by definition is 0. This also is true of the black background.

(b) The colors are spectrum colors, so they are fully saturated. Therefore, the values 255 shown apply to all circle regions. The region in the center of the color image is white, so its saturation is 0.

(c) The key to getting the values in this figure is to realize that the center portion of the color image is white, which means equal intensities of fully saturated red, green, and blue. Therefore, the value of both darker gray regions in the intensity image have value 85 (i.e., the same value as the other corresponding region). Similarly, equal proportions of the secondary yellow, cyan, and magenta produce white, so the two lighter gray regions have the same value (170) as the region shown in the figure. The center of the image is white, so its value is 255.

6. Problem 6.22 (10%)

Based on the discussion is Section 6.5.4 and with reference to the color wheel in Fig. 6.32, we can decrease the proportion of yellow by

(1) decreasing yellow,

(2) increasing blue,

(3) increasing cyan and magenta, or

(4) decreasing red and green.

7. Problem 6.23 (10%)

The L*a*b components are computed using Eqs (6.5-9) through (6.5-12). Reference white is R = G =B =1. The computations are best done in a spreadsheet, as shown below

Color	R	G	B	X	Y	Z	$\frac{X}{X_W}$	$\frac{Y}{Y_W}$	$\frac{Z}{Z_W}$	h($\frac{X}{X_W}$)	h($\frac{Y}{Y_W}$)	h($\frac{Z}{Z_W}$)	L*	a*	b*
Ref	1	1	1	0.95	1.00	1.10	1	1	1	1	1	1	100	0	0
Black	0	0	0	0	0	0	0	0	0	0.14	0.14	0.14	0	0	0
Red	1	0	0	0.59	0.29	0	0.62	0.29	0	0.85	0.66	0.14	61	95	105
Yellow	1	1	0	0.77	0.9	0.07	0.81	0.9	0.06	0.93	0.96	0.4	96	-16	113
Green	0	1	0	0.18	0.61	0.07	0.19	0.61	0.06	0.57	0.85	0.4	82	-136	90
Cyan	0	1	1	0.36	0.71	1.09	0.38	0.71	0.99	0.72	0.89	1	88	-84	-22
Blue	0	0	1	0.18	0.11	1.02	0.19	0.11	0.93	0.58	0.47	0.98	39	53	-101
Magenta	1	0	1	0.77	0.4	1.02	0.81	0.4	0.93	0.93	0.73	0.98	69	100	-49
White	1	1	1	0.95	1	1.1	1	1	1	1	1	1	100	0	0
Gray	0.5	0.5	0.5	0.48	0.5	0.55	0.5	0.5	0.5	0.79	0.79	0.79	76	0	0

8. **Pseudo-Color Image Processing** (10%)

(a) Implement Fig. 6.23, with the characteristic that you can specify two ranges of gray-level values for the input image and your program will output an RGB image whose pixels have a specified color corresponding to one range of gray levels in the

input image, and the remaining pixels in the RGB image have the same shade of gray as they had in the input image. You can limit the input colors to all the colors in Fig. 6.4(a).

(b) Download the image in Fig. 1.10(4) from the course web site and process it with your program so that the river appears yellow and the rest of the pixels are the same shades of gray as in the input image. It is acceptable to have isolated specs in the image that also appear yellow, but these should be kept as few as possible by proper choice of the two gray-level bands that you input into your program.

Ans:

(a)

(Matlab code)

```
clc;clear all;close all;
img = imread('Fig0110(4)(WashingtonDC Band4).tif');
imgsize=size(img);
M = input('1st parameter of range:');
N = input('2ed parameter of range:');
if(isempty(M))
    M = 0;
end
if(isempty(N))
    N = 30;
end
img2 = ind2rgb(img,gray(255));
for i=1:imgsize(1)
    for j=1:imgsize(2)
        if(img(i,j)>=M && img(i,j)<=N)
            img2(i,j,1) = 255;
            img2(i,j,2) = 255;
            img2(i,j,3) = 0;
        end
    end
end
figure,imshow(img);
figure,imshow(img2);
```

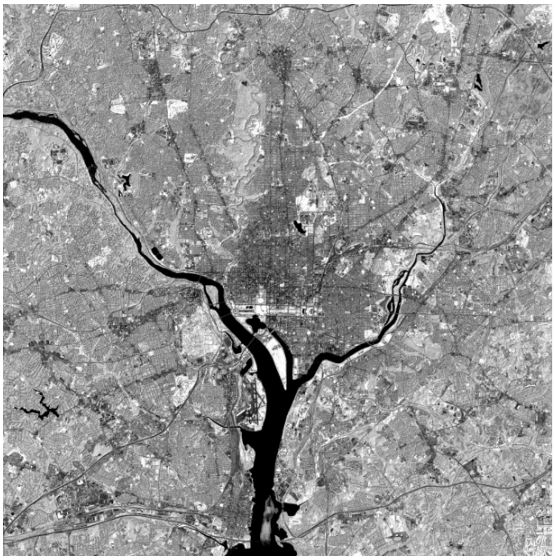
(Python code)

```
import cv2
import numpy as np
img = cv2.imread('Fig0110(4)(WashingtonDC Band4).tif', cv2.IMREAD_GRAYSCALE)
```

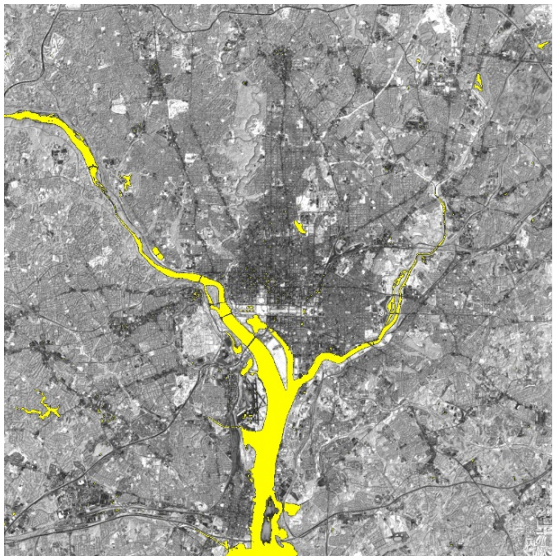
```
imgsize = img.shape

M = input('1st parameter of range: ')
N = input('2nd parameter of range: ')
if M == '':
    M_input = 0
else:
    M_input = int(M)
if N == '':
    N_input = 30
else:
    N_input = int(N)

img2 = cv2.cvtColor(img, cv2.COLOR_GRAY2RGB)

for i in range(imgsize[0]):
    for j in range(imgsize[1]):
        if M_input <= img[i, j] <= N_input:
            img2[i, j, 0] = 0           #B
            img2[i, j, 1] = 255         #G
            img2[i, j, 2] = 255         #R
cv2.imshow('Original Image', img)
cv2.imshow('Modified Image', img2)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

(b)

The two gray-scale bands that I input into the Python program are 0 and 25, and if you press Enter directly, they will be 0 and 30, the river will also appears yellow and the rest of the pixels are the same shades of gray as in the input image.



original image



pseudo-color image

9. Color Image Enhancement by Histogram Processing (10%)

(a) Download the dark-stream color picture in Fig. 6.35 from the course web site. Convert the image to RGB. Histogram-equalize the R, G, and B images separately using the histogram-equalization program from Homework 2 and convert the image back to tif format.

(b) Form an average histogram from the three histograms in (a) and use it as the basis to obtain a single histogram equalization intensity transformation function. Apply this function to the R, G, and B components individually, and convert the results to tif. Compare and explain the differences in the tif images in (a) and (b).

Ans:

(a)(b)

(Matlab code)

```
clc;
clear all;
img = imread('Fig0635(bottom_left_stream).tif');
figure(1),imshow(img);
title('Original image');
S = size(img);
%(a)
imgA(:,:,1) = histeq(img(:,:,1));
imgA(:,:,2) = histeq(img(:,:,2));
imgA(:,:,3) = histeq(img(:,:,3));
figure(2),imshow(imgA);
title('(a)');
%(b)
imgB = [img(:,:,1) img(:,:,2) img(:,:,3)];
imgB = histeq(imgB);
for i=1:S(1)
    for j=1:S(2)
        img(i,j,1) = imgB(i, j);
        img(i,j,2) = imgB(i, j + S(2));
        img(i,j,3) = imgB(i, j + S(2) + S(2));
    end
end
figure(3),imshow(img);
title('(b)');
```

(Python code)

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

img = cv2.imread('Fig0635(bottom_left_stream).tif')
```

```
plt.figure(1)
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.title('Original image')
imgA = img.copy()
imgA[:, :, 0] = cv2.equalizeHist(img[:, :, 0])
imgA[:, :, 1] = cv2.equalizeHist(img[:, :, 1])
imgA[:, :, 2] = cv2.equalizeHist(img[:, :, 2])
plt.figure(2)
plt.imshow(cv2.cvtColor(imgA, cv2.COLOR_BGR2RGB))
plt.title('(a)')
imgB = np.concatenate((img[:, :, 0], img[:, :, 1], img[:, :, 2]), axis=1)
imgB = cv2.equalizeHist(imgB)
S = img.shape
for i in range(S[0]):
    for j in range(S[1]):
        img[i, j, 0] = imgB[i, j]
        img[i, j, 1] = imgB[i, j + S[1]]
        img[i, j, 2] = imgB[i, j + S[1] + S[1]]
plt.figure(3)
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.title('(b)')
plt.show()
```

(a)

(b)

(a)小題是使用RGB各自等化的結果，因此顏色強度會平均分布，(b)小題則是平均後等化的結果，因此顏色較不鮮明，在視覺比較上，(a)比(b)還要來的好。

10. **Color Image Segmentation** (10%)

Download Fig. 6.28(b) from the course web site and duplicate Example 6.15, but segment instead the darkest regions in the image.

Ans:

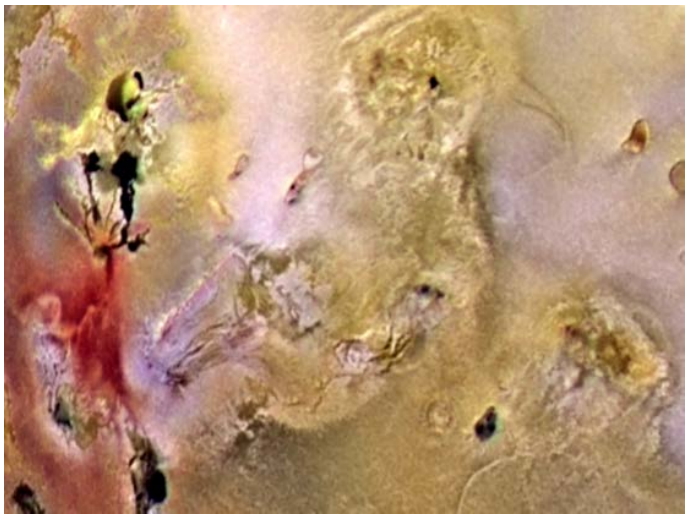
(Matlab code)

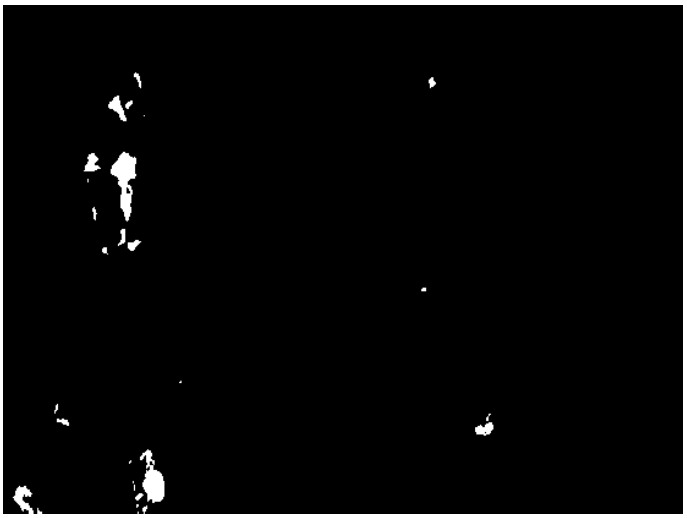
```
clc; clear all;
ori_Img = imread('Fig0628(b)(jupiter-Io-closeup).tif');
[w,h,c] = size(ori_Img);
seg_Img = zeros(w,h);
test_img = ori_Img;
for i =1:w
    for j=1:h
        if   ori_Img(i,j,1) < 50 && ori_Img(i,j,2)<50&& ori_Img(i,j,3)<50
            seg_Img(i,j)=1;
            test_Img(i,j,:)=[0, 0, 255];
        end
    end
end
figure,imshow(ori_Img);
figure,imshow(seg_Img);
figure,imshow(test_Img);
```

(Python code)

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

```
ori_img = cv2.imread('Fig0628(b)(jupiter-Io-closeup).tif')
w, h, c = ori_img.shape
seg_img = np.zeros((w, h), dtype=np.uint8)
test_img = ori_img.copy()
for i in range(w):
    for j in range(h):
        if ori_img[i, j, 0] < 50 and ori_img[i, j, 1] < 50 and ori_img[i, j, 2] < 50:
            seg_img[i, j] = 1
            test_img[i, j] = [255, 0, 0]    # Change color to blue
plt.figure()
plt.imshow(cv2.cvtColor(ori_img, cv2.COLOR_BGR2RGB))
plt.figure()
plt.imshow(seg_img, cmap='gray')
plt.figure()
plt.imshow(cv2.cvtColor(test_img, cv2.COLOR_BGR2RGB))
plt.show()
```





Can be clearly seen in the figure below, the darkest area is segmented.