Introduction to Digital Image Processing **HW #2 Solutions**

(Gonzalez 3rd edition)

1. Problem 3.7 (10%)

Let $n = MN$ be the total number of pixels and let n_{r_j} be the number of pixels in the input image with intensity value r_j. Then, the histogram equalization transformation is

$$s_k = T(r_k) = \sum_{j=0}^{k} n_{r_j}/n = \frac{1}{n}\sum_{j=0}^{k} n_{r_j}.$$

Because every pixel (and no others) with value r_k is mapped to value s_k, it follows that $n_{s_k} = n_{r_k}$. A second pass of histogram equalization would produce values v_k according to the transformation

$$v_k = T(s_k) = \frac{1}{n}\sum_{j=0}^{k} n_{s_j}.$$

But, $n_{s_j} = n_{r_j}$, so

$$v_k = T(s_k) = \frac{1}{n}\sum_{j=0}^{k} n_{r_j} = s_k$$

which shows that a second pass of histogram equalization would yield the same result as the first pass. We have assumed negligible round-off errors.

2. Problem 3.9 (10%)

We are interested in just one example in order to satisfy the statement of the problem. Consider the probability density function in Fig. P3.9(a). A plot of

the transformation $T(r)$ in Eq. (3.3-4) using this particular density function is shown in Fig. P3.9(b). Because $p_r(r)$ is a probability density function we know from the discussion in Section 3.3.1 that the transformation $T(r)$ satisfies conditions (a) and (b) stated in that section. However, we see from Fig. P3.9(b) that the inverse transformation from s back to r is not single valued, as there are an infinite number of possible mappings from $s = (L-1)/2$ back to r. It is important to note that the reason the inverse transformation function turned out not to be single valued is the gap in $p_r(r)$ in the interval $[L/4, 3L/4]$.

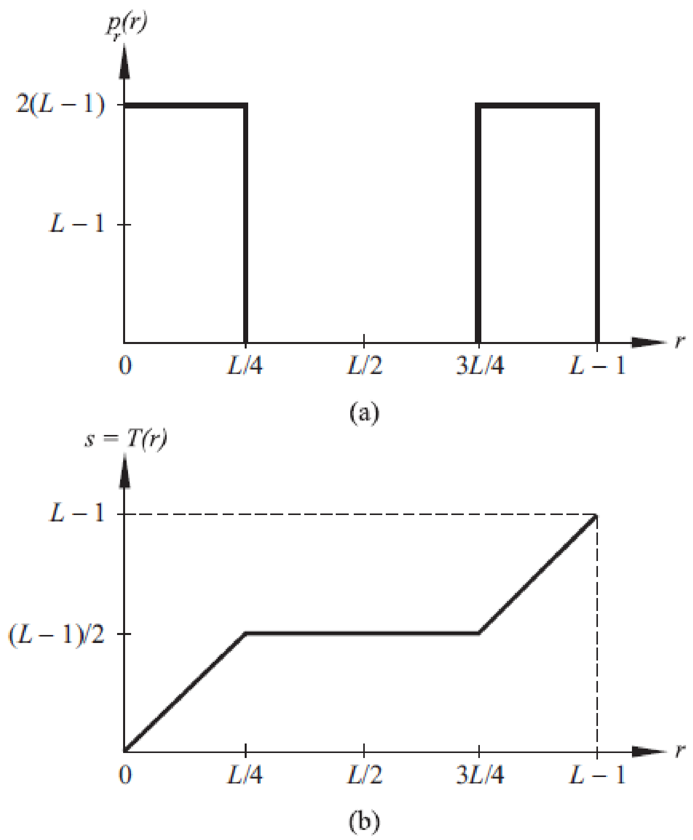


Figure P3.9.

3. Problem 3.11 (10%)

First, we obtain the histogram equalization transformation:

$$s = T(r) = \int_0^r p_r(w)dw = \int_0^r (-2w+2)dw = -r^2 + 2r.$$

Next we find

$$v = G(z) = \int_0^z p_z(w)\,dw = \int_0^z 2w\,dw = z^2.$$

Finally,

$$z = G^{-1}(v) = \pm\sqrt{v}.$$

But only positive intensity levels are allowed, so $z = \sqrt{v}$. Then, we replace v with s, which in turn is $-r^2 + 2r$, and we have

$$z = \sqrt{-r^2 + 2r}.$$

4. Problem 3.20 (10%)

(a) The most extreme case is when the mask is positioned on the center pixel of a 3-pixel gap, along a thin segment, in which case a 3×3 mask would encompass a completely blank field. Since this is known to be the largest gap, the next (odd) mask size up is guaranteed to encompass some of the pixels in the segment. Thus, the smallest mask that will do the job is a 5×5 averaging mask.

(b) The smallest average value produced by the mask is when it encompasses only two pixels of the segment. This average value is a gray-scale value, not binary, like the rest of the segment pixels. Denote the smallest average value by $A_{\min}$, and the binary values of pixels in the thin segment by B. Clearly, $A_{\min}$ is less than B. Then, setting the binarizing threshold slightly smaller than $A_{\min}$ will create one binary pixel of value B in the center of the mask.

5. Problem 3.24 (10%)

The Laplacian operator is defined as

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

for the unrotated coordinates, and as

$$\nabla^2 f = \frac{\partial^2 f}{\partial x'^2} + \frac{\partial^2 f}{\partial y'^2}.$$

for rotated coordinates. It is given that

$$x = x'\cos\theta - y'\sin\theta \quad \text{and} \quad y = x'\sin\theta + y'\cos\theta$$

where θ is the angle of rotation. We want to show that the right sides of the first two equations are equal. We start with

$$\begin{aligned}\frac{\partial f}{\partial x'} &= \frac{\partial f}{\partial x}\frac{\partial x}{\partial x'} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial x'} \\ &= \frac{\partial f}{\partial x}\cos\theta + \frac{\partial f}{\partial y}\sin\theta.\end{aligned}$$

Taking the partial derivative of this expression again with respect to x' yields

$$\frac{\partial^2 f}{\partial x'^2} = \frac{\partial^2 f}{\partial x^2}\cos^2\theta + \frac{\partial}{\partial x}\left(\frac{\partial f}{\partial y}\right)\sin\theta\cos\theta + \frac{\partial}{\partial y}\left(\frac{\partial f}{\partial x}\right)\cos\theta\sin\theta + \frac{\partial^2 f}{\partial y^2}\sin^2\theta.$$

Next, we compute

$$\begin{aligned}\frac{\partial f}{\partial y'} &= \frac{\partial f}{\partial x}\frac{\partial x}{\partial y'} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial y'} \\ &= -\frac{\partial f}{\partial x}\sin\theta + \frac{\partial f}{\partial y}\cos\theta.\end{aligned}$$

Taking the derivative of this expression again with respect to y' gives

$$\frac{\partial^2 f}{\partial y'^2}=\frac{\partial^2 f}{\partial x^2}\sin^2\theta-\frac{\partial}{\partial x}\left(\frac{\partial f}{\partial y}\right)\cos\theta\sin\theta-\frac{\partial}{\partial y}\left(\frac{\partial f}{\partial x}\right)\sin\theta\cos\theta+\frac{\partial^2 f}{\partial y^2}\cos^2\theta.$$

Adding the two expressions for the second derivatives yields

$$\frac{\partial^2 f}{\partial x'^2}+\frac{\partial^2 f}{\partial y'^2}=\frac{\partial^2 f}{\partial x^2}+\frac{\partial^2 f}{\partial y^2}$$

which proves that the Laplacian operator is independent of rotation.

6. Problem 3.25 (10%)

The Laplacian mask with a -4 in the center performs an operation proportional to differentiation in the horizontal and vertical directions. Consider for a moment a 3×3 "Laplacian" mask with a -2 in the center and 1s above and below the center. All other elements are 0. This mask will perform differentiation in only one direction, and will ignore intensity transitions in the orthogonal direction. An image processed with such a mask will exhibit sharpening in only one direction. A Laplacian mask with a -4 in the center and 1s in the vertical and horizontal directions will obviously produce an image with sharpening in both directions and in general will appear sharper than with the previous mask. Similarly, and mask with a -8 in the center and 1s in the horizontal, vertical, and diagonal directions will detect the same intensity changes as the mask with the -4 in the center but, in addition, it will also be able to detect changes along the diagonals, thus generally producing sharper-looking results.

7. Problem 3.27 (10%)

With reference to Eqs. (3.6-8) and (3.6-9), and using $k=1$ we can write the following equation for unsharp masking:

$$\begin{aligned} g(x,y) &= f(x,y)+f(x,y)-\bar{f}(x,y) \\ &= 2f(x,y)-\bar{f}(x,y) \end{aligned}$$

Convolving $f(x,y)$ with the mask in Fig. P3.27(a) produces $f(x,y)$. Convolving $f(x,y)$ with the mask in Fig. 3.32(a) produces $\bar{f}(x,y)$. Then, because these operations are linear, we can use superposition, and we see from the preceding equation that using two masks of the form in Fig. P3.27(a) and the mask in Fig. 3.32(a) produces the composite mask in Fig. P3.27(b). Convolving this mask with $f(x,y)$ produces $g(x,y)$, the unsharp result.

$$\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} \qquad \frac{1}{9} \times \begin{array}{|c|c|c|}\hline -1 & -1 & -1 \\ \hline -1 & 17 & -1 \\ \hline -1 & -1 & -1 \\ \hline \end{array}$$

(a) (b)

Figure P3.27

8. **Histogram Equalization** (10%)

(a) Write a computer program for computing the histogram of an image.

(b) Implement the histogram equalization technique discussed in Section 3.3.1.

(c) Download Fig. 3.8(a) from the course web site and perform histogram equalization on it. As a minimum, your answer should include the original image, a plot of its histogram, a plot of the histogram-equalization transformation function, the enhanced image, and a plot of its histogram. Use this information to explain why the resulting image was enhanced as it was.

Sol:

(a)

(Matlab code)

```
clc;
clear;
src = imread('Fig0308(a)(fractured_spine).tif');
[rows, colums, channels] = size(src);
histogram = zeros(256);
fprintf('%d', src(5, 5));
for row = 1:rows
      for colum = 1:colums
          num = src(row, colum);
          histogram(num + 1) = histogram(num + 1) + 1;
      end
end
subplot(2, 2, 1);
plot(histogram);
subplot(2, 2, 2);
imshow(src);
```

(Python code)

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
src = cv2.imread('Fig0308(a)(fractured_spine).tif', 0)
rows, cols = src.shape[:2]
histogram = np.zeros(256)
print(src[4, 4])
for row in range(rows):
    for col in range(cols):
        num = src[row, col]
        histogram[num] = histogram[num] + 1
plt.subplot(2, 2, 1)
plt.plot(histogram)
plt.subplot(2, 2, 2)
plt.imshow(src, cmap='gray')
plt.show()
```

(b)

(Matlab code)

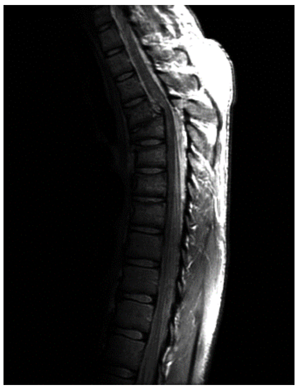
```
clc;
clear;
src = imread('Fig0308(a)(fractured_spine).tif');
[rows, colums] = size(src);
subplot(2, 3, 1);
imshow(src);

histogram = zeros(256);
for row = 1:rows
    for colum = 1:colums
        num = src(row, colum);
        histogram(num + 1) = histogram(num + 1) + 1;
    end
end
subplot(2, 3, 2);
plot(histogram);

pdf_src = histogram / (rows * colums);
transform = zeros(256);
num = 0;
for count = 1:256
    num = num + pdf_src(count);
```

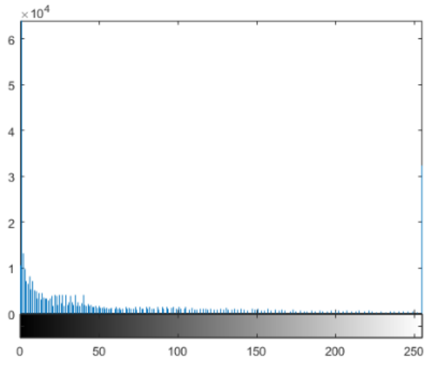
```
        transform(count) = round(num * 256);
end
subplot(2, 3, 3);
plot(transform);
finish = src;
histogram = zeros(256);
for row = 1:rows
        for colum = 1:colums
              index = src(row, colum) + 1;
              finish(row, colum) = transform(index);
              histogram(transform(index)) = histogram(transform(index)) + 1;
        end
end
subplot(2, 3, 4);
imshow(finish);
subplot(2, 3, 5);
plot(histogram);
```

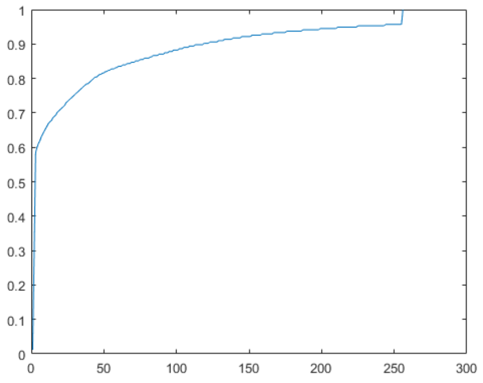
(Python code)

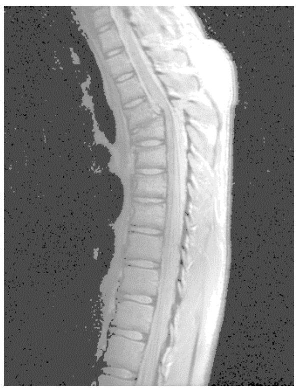
```
import cv2
import numpy as np
from matplotlib import pyplot as plt

src = cv2.imread('Fig0308(a)(fractured_spine).tif', cv2.IMREAD_GRAYSCALE)
rows, cols = src.shape[:2]
plt.subplot(2, 3, 1)
plt.imshow(src, cmap='gray')
histogram = np.zeros(256, dtype=np.uint32)
for row in range(rows):
        for col in range(cols):
                num = src[row, col]
                histogram[num] = histogram[num] + 1
plt.subplot(2, 3, 2)
plt.plot(histogram)
pdf_src = histogram / (rows * cols)
transform = np.zeros(256, dtype=np.uint8)
num = 0.0
for count in range(256):
        num = num + pdf_src[count]
        transform[count] = int(round(num * 255))
plt.subplot(2, 3, 3)
plt.plot(transform)
finish = np.zeros((rows, cols), dtype=np.uint8)
histogram = np.zeros(256, dtype=np.uint32)
for row in range(rows):
        for col in range(cols):
                index = src[row, col]
                finish[row, col] = transform[index]
                histogram[transform[index]] = histogram[transform[index]] + 1
plt.subplot(2, 3, 4)
plt.imshow(finish, cmap='gray')
plt.subplot(2, 3, 5)
plt.plot(histogram)
plt.show()
```

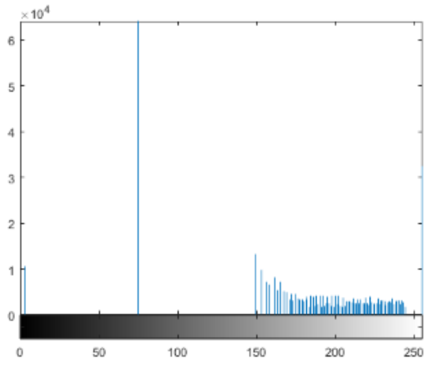
(c)


original image


plot of its histogram


plot of the histogram-equalization transformation function


enhanced image


plot of its histogram

直方圖等化是使用機率分佈， 將原先的亮度分佈重新的等化到新的亮度值，增加圖像的局部對比度，尤其是當圖像的有用數據的對比度相當接近的時候。透過直方圖等化，能更清楚看見圖像的細節。

9. **Enhancement Using the Laplacian** (10%)

(a) Write a program to perform spatial filtering of an image (see Section 3.4 regarding implementation). You can fix the size of the spatial mask at 3 x 3, but the coefficients need to be variables that can be input into your program.

(b) Use the programs developed in (a) to implement the Laplacian enhancement technique described in connection with Eq. (3.6-7).

(c) Duplicate the results in Fig. 3.38. You can download the original image from the course web site.

Sol:

(a)

(Matlab code)

```
clear all;
clc;
mask = zeros(3, 3);
for i = 1:3
    for j = 1:3
        fprintf('mask%d%d', i, j);
        mask(i, j) = input(':');
    end
end
src = imread('Fig0338(a)(blurry_moon).tif');
finish = imfilter(src, mask);
subplot(1, 2, 1);
imshow(src);
subplot(1, 2, 2);
imshow(finish);
```

(Python code)

```
import cv2
import numpy as np
mask = np.zeros((3, 3), dtype=np.float32)
for i in range(3):
    for j in range(3):
        print("mask[{}][{}]:".format(i, j), end="")
        mask[i, j] = float(input())
print("mask:")
print(mask)

src = cv2.imread('D:/pythonProject/Fig0338(a)(blurry_moon).tif')
finish = cv2.filter2D(src, -1, mask)
```

```
cv2.imwrite('output_image.png', finish)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

(b)

(Matlab code)

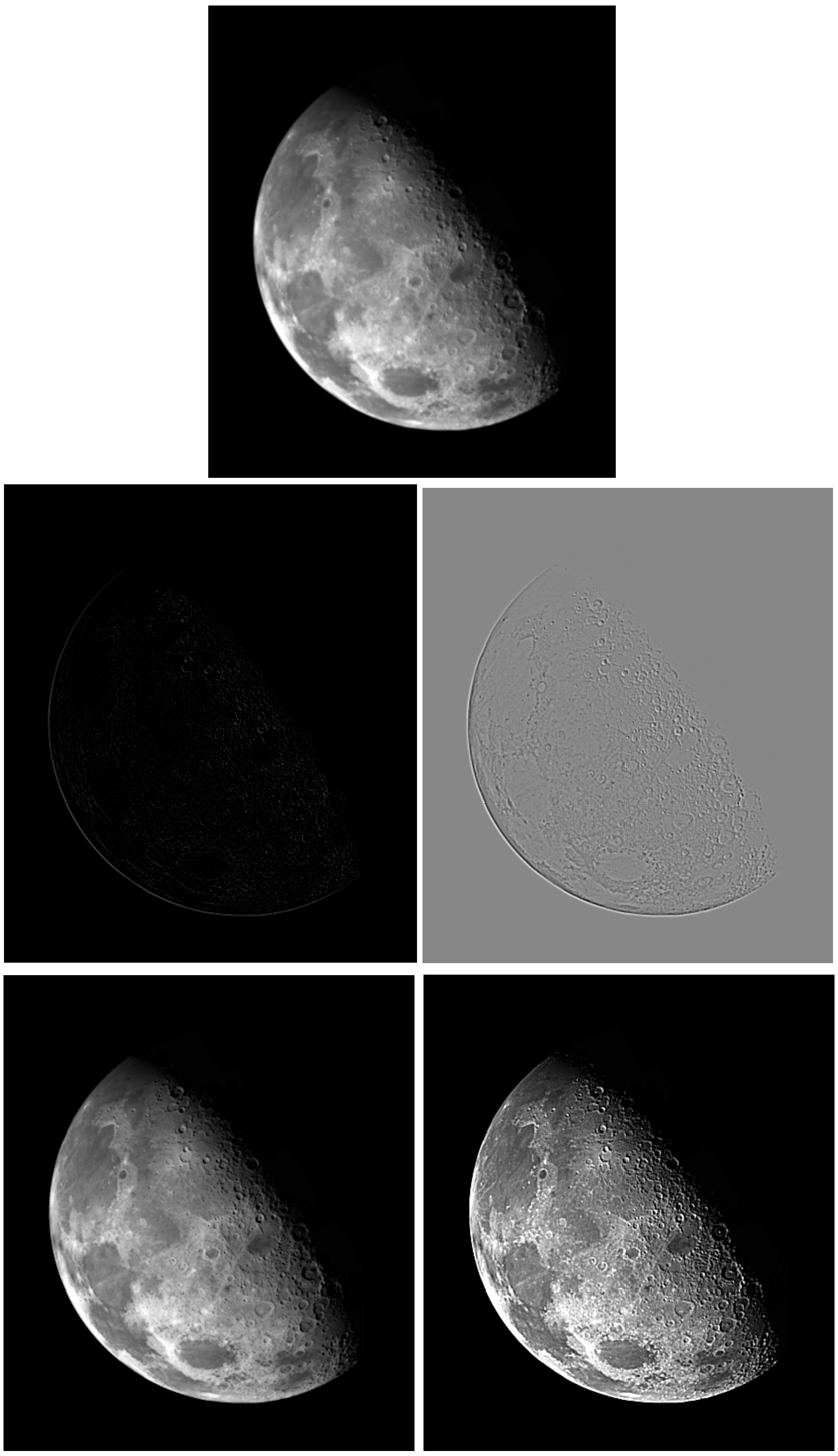
```
clear all;
clc;
mask = zeros(3, 3);
for i = 1:3
    for j = 1:3
        fprintf('mask%d%d', i, j);
        mask(i, j) = input(':');
    end
end
src = imread('Fig0338(a)(blurry_moon).tif');
finish = imfilter(src, mask);
enh = src + finish;
subplot(1, 2, 1);
imshow(enh);
subplot(1, 2, 2);
imshow(finish);
```

(Python code)

```
import cv2
import numpy as np
#(b)
mask = np.zeros((3, 3), dtype=np.float32)

for i in range(3):
    for j in range(3):
        print("mask[{}][{}]:".format(i, j), end="")
        mask[i, j] = float(input())

print("mask:")
print(mask)
src = cv2.imread('Fig0338(a)(blurry_moon).tif')
finish = cv2.filter2D(src, -1, mask)
enh = src + finish
```

```
cv2.imwrite('Enhanced Image.png', enh)
cv2.imwrite('Filtered Image.png', finish)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

(c)



10. **Unsharp Masking** (10%)

(a) Use the program developed in 9. (a) to implement high-boost filtering, as given in Eq. (3.6-9). The averaging part of the process should be done using the mask in Fig. 3.32(a).

(b) Download Fig. 3.40(a) from the course web site and enhance it using the program you developed in (a). Your objective is to approximate the result in Fig. 3.40(e).

Sol:

(a)

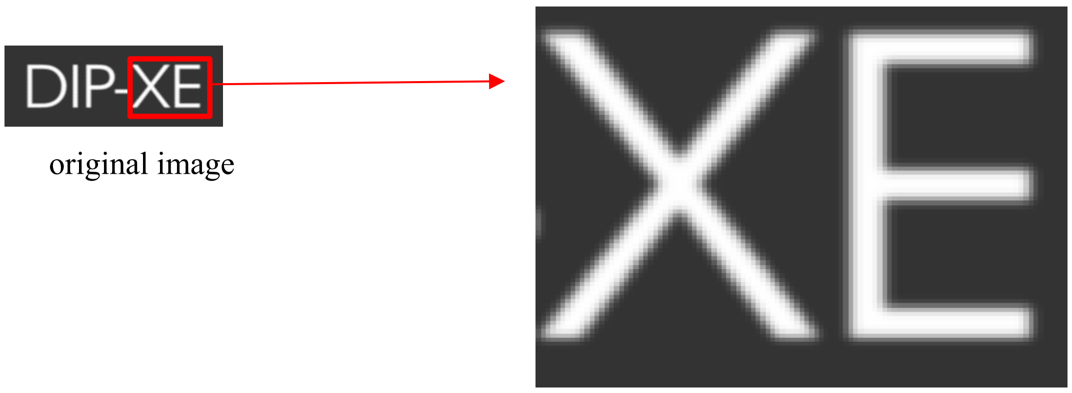
(Matlab code)

```
clc;clear all;close all;
I = imread('Fig0340(a)(dipxe_text).tif');
k = 4.5;
for i=1:9
        eval(['p' num2str(i) ' = input("input your desired factor' num2str(i) ' = ");']);
        eval(['if isempty(p' num2str(i) ') p' num2str(i) ' = 1/9; end']);
end
mask = [p1 p2 p3;p4 p5 p6;p7 p8 p9];
I2 = imfilter(I,mask,'conv');
unsharp_mask = I - I2;
highboost_image = I + k*unsharp_mask;
figure,imshow(I);
figure,imshow(highboost_image);
```

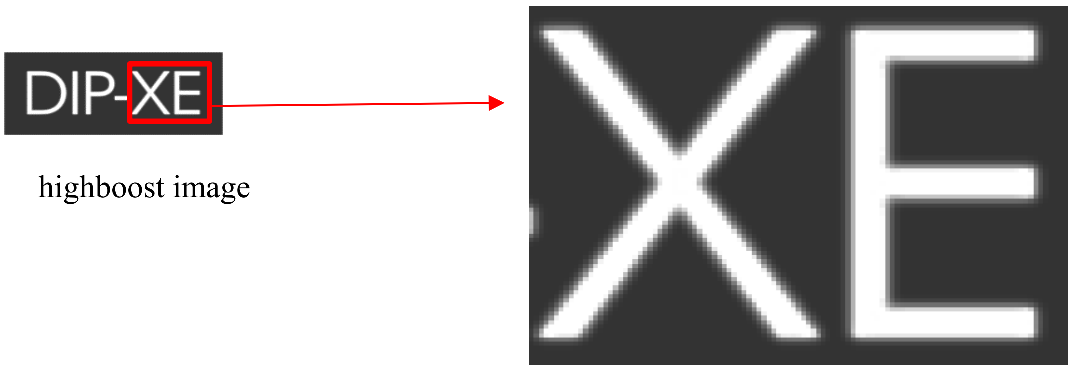
(Python code)

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

k = 4.5
mask = np.zeros((3, 3))
for i in range(1, 10):
        p = input("input your desired factor{} = ".format(i))
        if p == "":
                p = 1/9
        else:
                p = float(p)
        row = int((i - 1) / 3)
        col = int((i - 1) % 3)
        mask[row, col] = p
I = cv2.imread("Fig0340(a)(dipxe_text).tif", cv2.IMREAD_GRAYSCALE)
I2 = cv2.filter2D(I, -1, mask)
unsharp_mask = I - I2
highboost_image = I + k * unsharp_mask
plt.imshow(I, cmap="gray")
plt.title("Original Image")
plt.show()
plt.imshow(highboost_image, cmap="gray")
plt.title("High-boost Filtered Image")
plt.show()
```

(b)



original image



highboost image