ECE 638 Homework 6 Sample Solution

Question 1 (Tianqi Guo)

In [1]:
```
# Problem 1
import numpy as np
from skimage.io import imread
import matplotlib.pyplot as plt
```

In [2]:
```
img_ramp = imread('ramp.tif')
img_wmbw = imread('woman_bw.tif')
def screen(img, T):
    bw = np.zeros_like(img)
    m, n = img.shape
    p, q = T.shape
    for i in range(m):
        for j in range(n):
            bw[i][j] = img[i][j] > T[i%p][j%q]
    return bw
def print_fat(bw):
    if_print = 1-bw
    m, n = bw.shape
    printed = np.zeros((2*m,2*n))
    for i in range(m):
        for j in range(n):
            for jj in range(max(2*j-1,0),min(2*j+3,2*n-1)):
                for ii in range(2*i,2*i+2):
                    printed[ii][jj] = printed[ii][jj] or if_print[i][j]
    return printed
img = np.zeros((6,6))
one_pos = [(1,1),(1,3), (3,3),(3,4), (4,1), (5,3)]
for x,y in one_pos:
    img[x][y] = 1
print('Input bit map')
print(img)
print()
bw_img_test = print_fat(1-img)
print('Digital simulation of fat dot printer')
for row in bw_img_test:
    print(row)
```

```
Input bit map
[[0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 1. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 1. 0.]
 [0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0.]]

Digital simulation of fat dot printer
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
[0. 1. 1. 1. 1. 1. 1. 1. 1. 0. 0. 0.]
[0. 1. 1. 1. 1. 1. 1. 1. 1. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 1. 0.]
[0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 1. 0.]
[0. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0.]
[0. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 0. 0.]
[0. 0. 0. 0. 0. 1. 1. 1. 1. 0. 0. 0.]
```

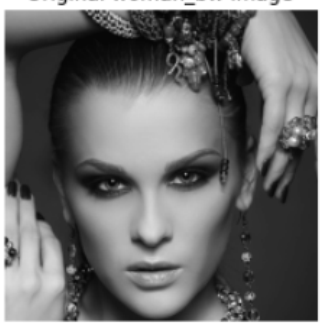
In [3]:

```
# a. Printouts of halftone images
#    from 8x8 clustered dot screen and 8x8 Bayer threshold matrix

i_8x8 = [[62, 57, 48, 36, 37, 49, 58, 63],
         [56, 47, 35, 21, 22, 38, 50, 59],
         [46, 34, 20, 10, 11, 23, 39, 51],
         [33, 19,  9,  3,  0,  4, 12, 24],
         [32, 18,  8,  2,  1,  5, 13, 25],
         [45, 31, 17,  7,  6, 14, 26, 40],
         [55, 44, 30, 16, 15, 27, 41, 52],
         [61, 54, 43, 29, 28, 42, 53, 60]]
T_8x8 = np.asarray((np.asarray(i_8x8)+0.5)/64*255, dtype=np.int32)

T_bayer = [[1, 65, 17, 81, 5, 69, 21, 85],
           [97, 33, 113, 49, 101, 37, 117, 53],
           [25, 89, 9, 73, 29, 93, 13, 77],
           [121, 57, 105, 41, 125, 61, 109, 45],
           [7, 71, 23, 87, 3, 67, 19, 83],
           [103, 39, 119, 55, 99, 35, 115, 51],
           [31, 95, 15, 79, 27, 91, 11, 75],
           [127, 63, 111, 47, 123, 59, 107, 43]]
T_bayer = np.asarray(T_bayer)/128.*255.

bw_ramp_8x8 = screen(img_ramp, T_8x8)
bw_ramp_bayer = screen(img_ramp, T_bayer)
bw_wmbw_8x8 = screen(img_wmbw, T_8x8)
bw_wmbw_bayer = screen(img_wmbw, T_bayer)

bw_ramp_8x8_fat = 1-print_fat(bw_ramp_8x8)
bw_ramp_bayer_fat = 1-print_fat(bw_ramp_bayer)
bw_wmbw_8x8_fat = 1-print_fat(bw_wmbw_8x8)
bw_wmbw_bayer_fat = 1-print_fat(bw_wmbw_bayer)

plt.figure(figsize=[8,20])

plt.subplot(521)
plt.imshow(img_ramp,cmap='gray')
plt.title('Original ramp image')
plt.axis('off')
plt.subplot(522)
plt.imshow(img_wmbw,cmap='gray')
plt.axis('off')
plt.title('Original woman_bw image')
plt.subplot(523)
plt.imshow(bw_ramp_8x8,cmap='gray')
plt.axis('off')
plt.title('8x8 clustered halftoning')
plt.subplot(524)
plt.imshow(bw_wmbw_8x8,cmap='gray')
plt.axis('off')
plt.title('8x8 clustered halftoning')
plt.subplot(525)
plt.imshow(bw_ramp_8x8_fat,cmap='gray')
plt.axis('off')
plt.title('Printouts by fat-dot printer')
plt.subplot(526)
plt.imshow(bw_wmbw_8x8_fat,cmap='gray')
plt.axis('off')
plt.title('Printouts by fat-dot printer')
```

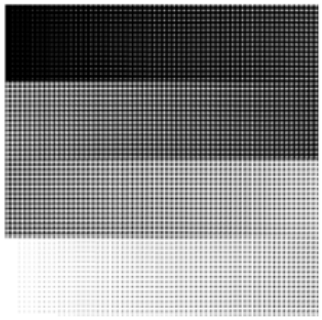
```
plt.subplot(5,2,7)
plt.imshow(bw_ramp_bayer,cmap='gray')
plt.axis('off')
plt.title('Halftoning with Bayer screen')
plt.subplot(5,2,8)
plt.imshow(bw_wmbw_bayer,cmap='gray')
plt.axis('off')
plt.title('Halftoning with Bayer screen')
plt.subplot(5,2,9)
plt.imshow(bw_ramp_bayer_fat,cmap='gray')
plt.axis('off')
plt.title('Printouts by fat-dot printer')
plt.subplot(5,2,10)
plt.imshow(bw_wmbw_bayer_fat,cmap='gray')
plt.axis('off')
plt.title('Printouts by fat-dot printer')

plt.show()
```

Original ramp image

Original woman_bw image

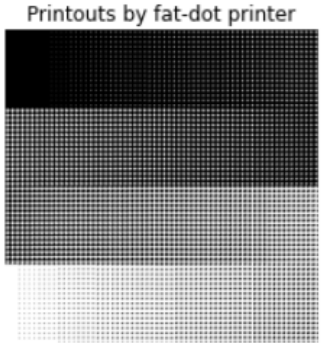


8x8 clustered halftoning

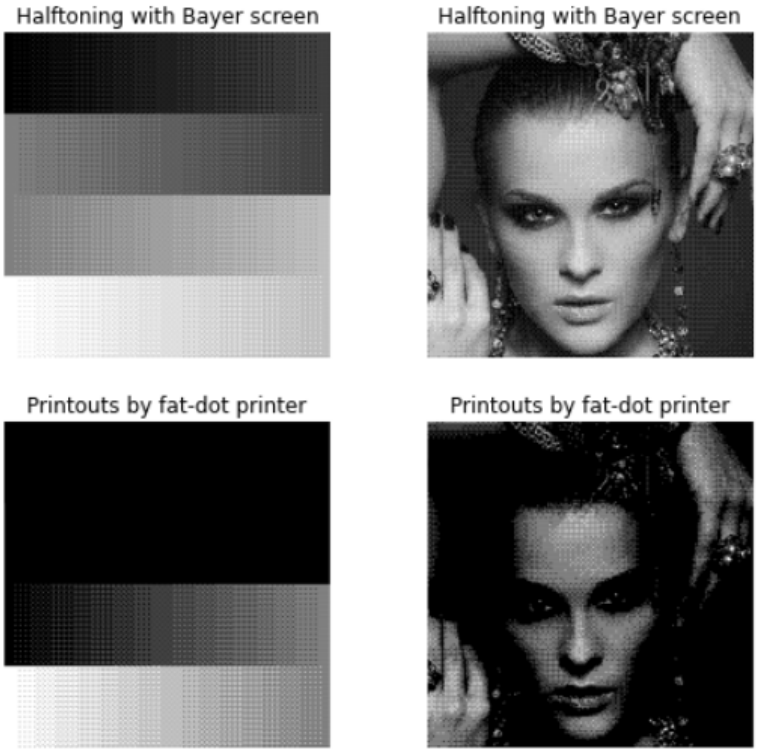


8x8 clustered halftoning

Printouts by fat-dot printer



Printouts by fat-dot printer



In [4]:

```
# Comments:
# 1) Both halftone images from the clustered-dot screen and Bayer screen
#    are darker than the originals since fat dots cause more areas
#    in the printouts to be occupied by ink where absorptance is 1
# 2) The halftone image from the clustered-dot screen is less affected
#    than the image from the Bayer screen.
#    This is because in clustered-dot screen the ink areas are 'clustered'
#    and therefore less highlight area is occupied by the fat dots.
#    On the contrary, in the Bayer screen the dots are 'dispersed',
#    and therefore fat dots can cover more highlight area surrounding each dark dot,
#    causing the printout to be much darker than desired
```

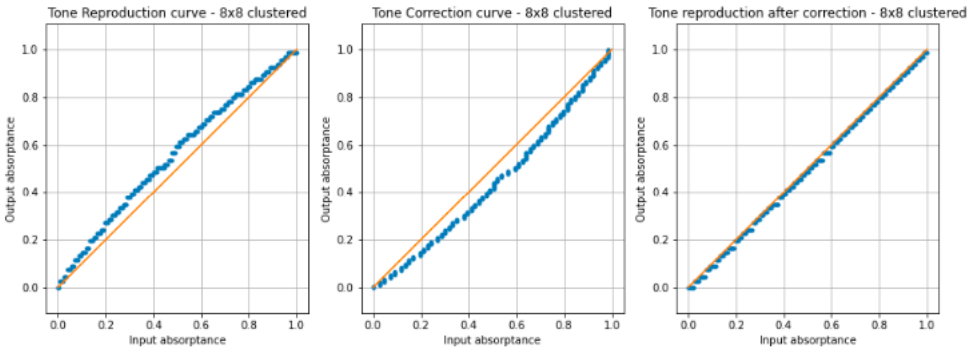
In [5]:

```
# b. Determine the tone-reproduction curves and tone-correction curves
#    Apply the respective tone-correction curve to the continuoustone images,
#    and regenerate the four halftone images.

def tone_reproduction(T):

    input_absorptance = []
    output_absorptance = []
    for gray_value in range(255,-1,-1):
        input_absorptance.append(1.0-gray_value/255.)
        input_img = gray_value * np.ones((32,32))
        output_bw = screen(input_img, T)
        printed_dots = np.asarray(print_fat(output_bw),dtype=np.float64)
        output_absorptance.append(printed_dots.mean())

    return input_absorptance, output_absorptance
```

```
def tone_correction(input_absorptance, output_absorptance,T):
    biased_absorptance = []
    corrected_absorptance = []
    corrected_grayvalue = []
    for gray_value in range(256):
        biased_absorptance_i = 1.0-gray_value/255.
        biased_absorptance.append(biased_absorptance_i)
        corrected_absorptance_i = np.interp(biased_absorptance_i,
                                            input_absorptance,
                                            output_absorptance)
        corrected_grayvalue_i = (1.0-corrected_absorptance_i)*255
        corrected_grayvalue.append(corrected_grayvalue_i)
        input_img = corrected_grayvalue_i * np.ones((32,32))
        output_bw = screen(input_img, T)
        printed_dots = np.asarray(print_fat(output_bw),dtype=np.float64)
        corrected_absorptance.append(printed_dots.mean())
    return biased_absorptance, corrected_absorptance, corrected_grayvalue

def img_mapping(img, corrected_grayvalue):
    img_out = np.zeros_like(img)
    m, n = img.shape
    for i in range(m):
        for j in range(n):
            img_out[i][j] = corrected_grayvalue[img[i][j]]
    return img_out
```

In [6]:

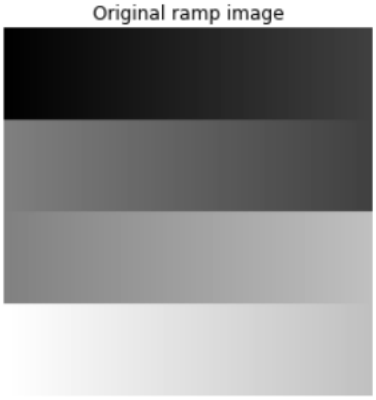
```
input_absorptance, output_absorptance = tone_reproduction(T_8x8)
biased_absorptance, \
corrected_absorptance, \
corrected_grayvalue_8x8 = tone_correction(output_absorptance,
                                          input_absorptance,
                                          T_8x8)
plt.figure(figsize=[15,5])
plt.subplot(131)
plt.plot(input_absorptance, output_absorptance,'.')
plt.plot([0,1],[0,1],'-')
plt.axis('equal')
plt.title('Tone Reproduction curve - 8x8 clustered')
plt.xlabel('Input absorptance')
plt.ylabel('Output absorptance')
plt.grid(b=None, which='major', axis='both')
plt.subplot(132)
plt.plot(output_absorptance, input_absorptance, '.')
plt.plot([0,1],[0,1],'-')
plt.axis('equal')
plt.title('Tone Correction curve - 8x8 clustered')
plt.xlabel('Input absorptance')
plt.ylabel('Output absorptance')
plt.grid(b=None, which='major', axis='both')
plt.subplot(133)
plt.plot(biased_absorptance, corrected_absorptance, '.')
plt.plot([0,1],[0,1],'-')
plt.axis('equal')
plt.title('Tone reproduction after correction - 8x8 clustered')
plt.xlabel('Input absorptance')
plt.ylabel('Output absorptance')
plt.grid(b=None, which='major', axis='both')
plt.show()
```



In [7]:

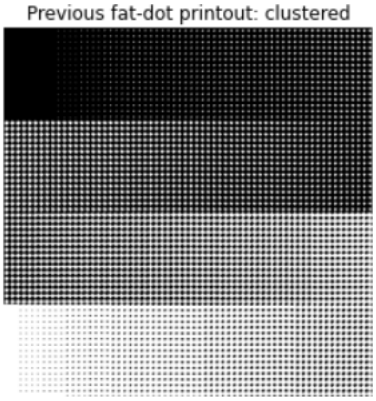
```
img_ramp_corrected_8x8 = img_mapping(img_ramp, corrected_grayvalue_8x8)
bw_ramp_8x8_corrected = screen(img_ramp_corrected_8x8, T_8x8)
bw_ramp_8x8_fat_corrected = 1-print_fat(bw_ramp_8x8_corrected)

img_wmbw_corrected_8x8 = img_mapping(img_wmbw, corrected_grayvalue_8x8)
bw_wmbw_8x8_corrected = screen(img_wmbw_corrected_8x8, T_8x8)
bw_wmbw_8x8_fat_corrected = 1-print_fat(bw_wmbw_8x8_corrected)

plt.figure(figsize=[10,15])
plt.subplot(321)
plt.imshow(img_ramp,cmap='gray')
plt.title('Original ramp image')
plt.axis('off')
plt.subplot(322)
plt.imshow(img_wmbw,cmap='gray')
plt.axis('off')
plt.title('Original woman_bw image')

plt.subplot(323)
plt.imshow(bw_ramp_8x8_fat,cmap='gray')
plt.axis('off')
plt.title('Previous fat-dot printout: clustered')
plt.subplot(324)
plt.imshow(bw_wmbw_8x8_fat,cmap='gray')
plt.axis('off')
plt.title('Previous fat-dot printout: clustered')

plt.subplot(325)
plt.imshow(bw_ramp_8x8_fat_corrected,cmap='gray')
plt.axis('off')
plt.title('Corrected fat-dot printout: clustered')
plt.subplot(326)
plt.imshow(bw_wmbw_8x8_fat_corrected,cmap='gray')
plt.axis('off')
plt.title('Corrected fat-dot printout: clustered')
plt.show()
```

Original ramp image

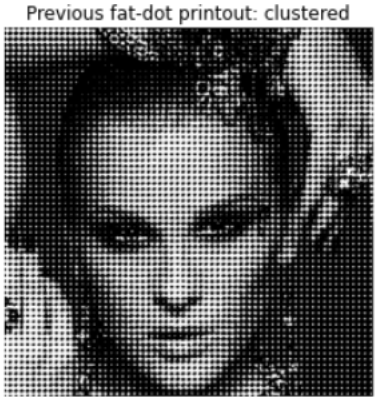


Original woman_bw image

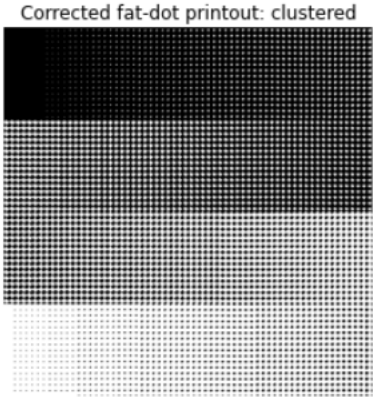
Previous fat-dot printout: clustered



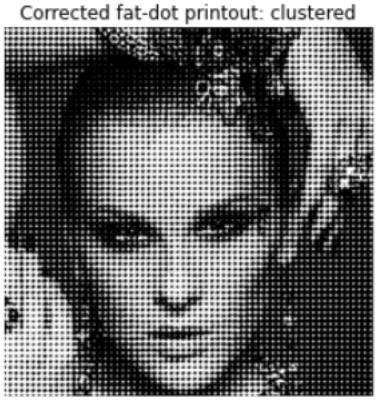
Previous fat-dot printout: clustered



Corrected fat-dot printout: clustered



Corrected fat-dot printout: clustered



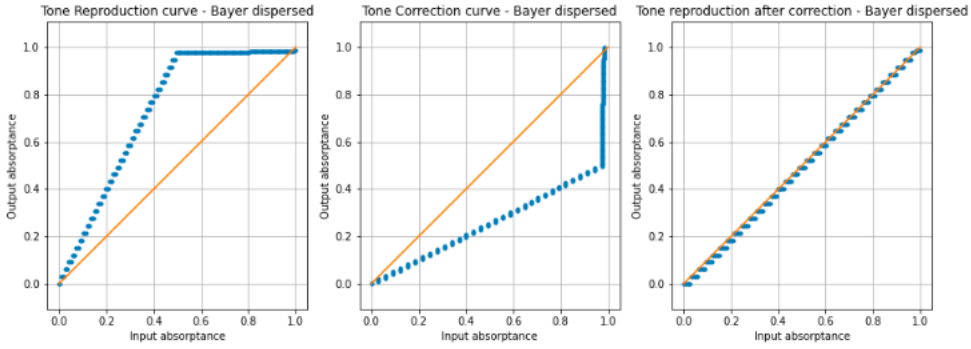
In [8]:

```
# Comments:
# After tone correction,
# both printouts from clustered\dispersed dot screens are better than before,
# and the overall brightness is more similar to the original images
# since the averaged absorptance level is lowered due to reduced ink printed
```

```
# However, the Bayer printouts after tone correction still have artifacts
# especially noticeable in the high brightness region
```

In [9]:

```
input_absorptance, output_absorptance = tone_reproduction(T_bayer)
biased_absorptance, \
corrected_absorptance, \
corrected_grayvalue_bayer = tone_correction(output_absorptance,
                                            input_absorptance,
                                            T_bayer)
plt.figure(figsize=[15,5])
plt.subplot(131)
plt.plot(input_absorptance, output_absorptance,'.')
plt.plot([0,1],[0,1],'-')
plt.axis('equal')
plt.title('Tone Reproduction curve - Bayer dispersed')
plt.xlabel('Input absorptance')
plt.ylabel('Output absorptance')
plt.grid(b=None, which='major', axis='both')

plt.subplot(132)
plt.plot(output_absorptance, input_absorptance, '.')
plt.plot([0,1],[0,1],'-')
plt.axis('equal')
plt.title('Tone Correction curve - Bayer dispersed')
plt.xlabel('Input absorptance')
plt.ylabel('Output absorptance')
plt.grid(b=None, which='major', axis='both')

plt.subplot(133)
plt.plot(biased_absorptance, corrected_absorptance, '.')
plt.plot([0,1],[0,1],'-')
plt.axis('equal')
plt.title('Tone reproduction after correction - Bayer dispersed')
plt.xlabel('Input absorptance')
plt.ylabel('Output absorptance')
plt.grid(b=None, which='major', axis='both')

plt.show()
```

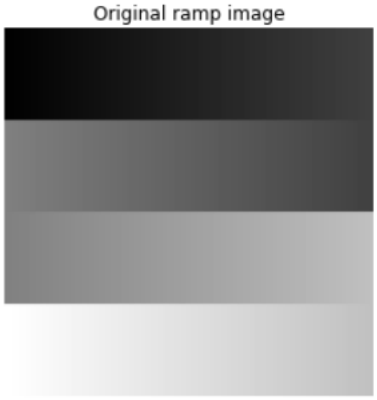


In [10]:

```
img_ramp_corrected_bayer = img_mapping(img_ramp, corrected_grayvalue_bayer)
bw_ramp_bayer_corrected = screen(img_ramp_corrected_bayer, T_bayer)
bw_ramp_bayer_fat_corrected = 1-print_fat(bw_ramp_bayer_corrected)
```

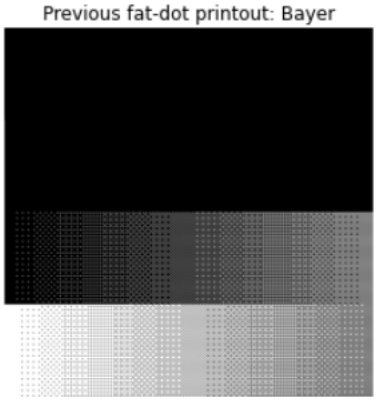
```
img_wmbw_corrected_bayer = img_mapping(img_wmbw, corrected_grayvalue_bayer)
bw_wmbw_bayer_corrected = screen(img_wmbw_corrected_bayer, T_bayer)
bw_wmbw_bayer_fat_corrected = 1-print_fat(bw_wmbw_bayer_corrected)

plt.figure(figsize=[10,15])
plt.subplot(321)
plt.imshow(img_ramp,cmap='gray')
plt.title('Original ramp image')
plt.axis('off')
plt.subplot(322)
plt.imshow(img_wmbw,cmap='gray')
plt.axis('off')
plt.title('Original woman_bw image')

plt.subplot(323)
plt.imshow(bw_ramp_bayer_fat,cmap='gray')
plt.axis('off')
plt.title('Previous fat-dot printout: Bayer')
plt.subplot(324)
plt.imshow(bw_wmbw_bayer_fat,cmap='gray')
plt.axis('off')
plt.title('Previous fat-dot printout: Bayer')

plt.subplot(325)
plt.imshow(bw_ramp_bayer_fat_corrected,cmap='gray')
plt.axis('off')
plt.title('Corrected fat-dot printout: Bayer')
plt.subplot(326)
plt.imshow(bw_wmbw_bayer_fat_corrected,cmap='gray')
plt.axis('off')
plt.title('Corrected fat-dot printout: Bayer')

plt.show()
```

Original ramp image



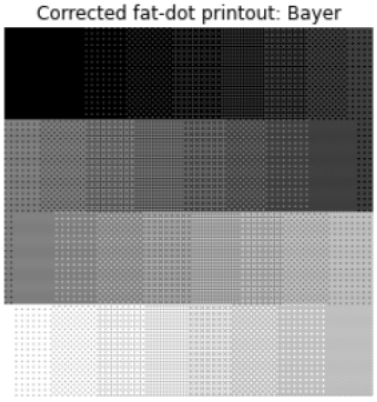
Original woman_bw image

Previous fat-dot printout: Bayer

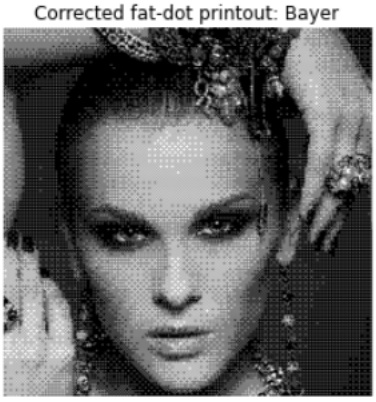


Previous fat-dot printout: Bayer

Corrected fat-dot printout: Bayer



Corrected fat-dot printout: Bayer



Question 2 (Kennedy Monaco)

ECE 638: Principles of Digital Color Imaging Systems — 23 November 2021
Homework 6 — Kennedy Monaco

Problem 2

a. MATLAB Code – Error Diffusion

```
function [img] = errorDiffusion(img)
rows = length(img);
cols = length(img(1,:));
img = img / 255;
for i = 1:rows
    for j=1:cols
        oldpixel = img(i,j);
        img(i,j) = round(oldpixel);
        err = oldpixel - img(i,j);
        if(j < cols)
            img(i,j+1) = img(i,j+1) + err * (7/16);
        end
        if(i < rows && j > 1)
            img(i+1,j-1) = img(i+1,j-1) + err * (3/16);
        end
        if(i < rows)
            img(i+1,j) = img(i+1,j) + err * (5/16);
        end
        if(i < rows && j < cols)
            img(i+1,j+1) = img(i+1,j+1) + err * (1/16);
        end
    end
end
end
```

b. NOTE: the void and cluster algorithm used to generate the images in this section was taken from Qiyue Liang's HW 5 Solution – Thank you Qiyue!
The Error Diffusion Ramp has clear artifacts along the left side of the image which is expected due to the causal structure of the Floyd-Steinberg diffusion matrix. These artifacts have little effect on the woman halftoned image. The ramp and woman images show that those halftoned using void-and-cluster are significantly darker in tone than those halftoned by error diffusion. Since the error diffusion halftoned image of the woman is brighter, it is easier to detect image details.

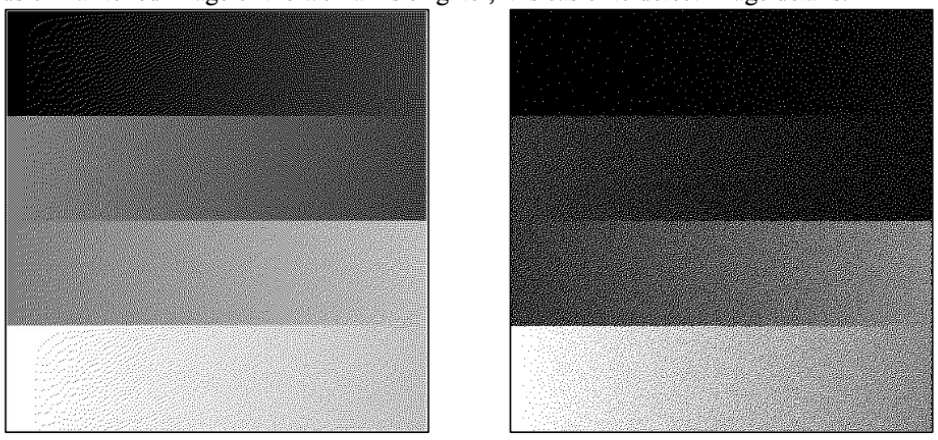


Figure 2.1: *ramp.tif halftoned with Floyd-Steinberg Error Diffusion (left) and Void-and-Cluster (right)*

Figure 2.2: *woman_bw.tif halftoned with Floyd-Steinberg Error Diffusion (left) and Void-and-Cluster (right)*

c. I incorporated Fan Bu's use of images to show the 2D DFTs of the images as I found it was much easier to compare the spectra this way rather than looking at 3D plots. The following images were generated with kappa = 2500. The main difference in the two spectra is that the error diffusion spectrum appears to better suppress the energy surrounding the original spectra of the woman_bw.tif image. The radius of suppression for the error diffusion DFT is larger and more apparent than that of the void-and-cluster spectra.

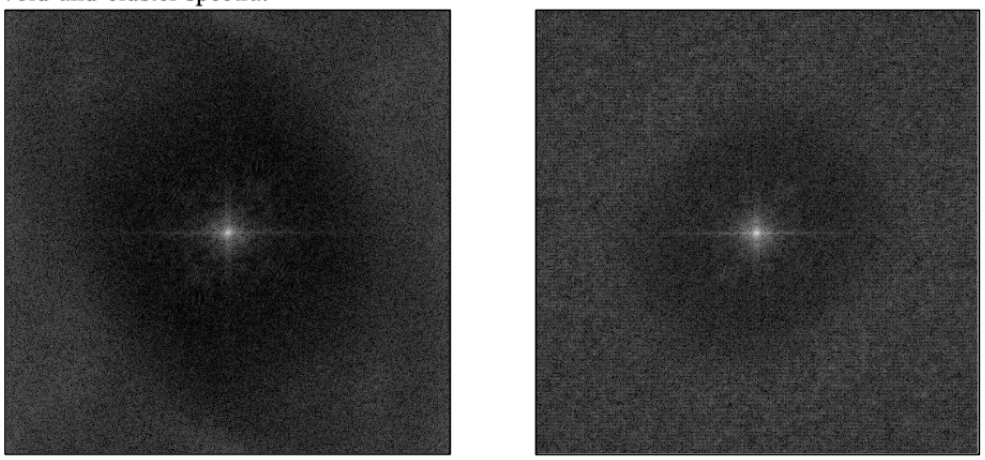


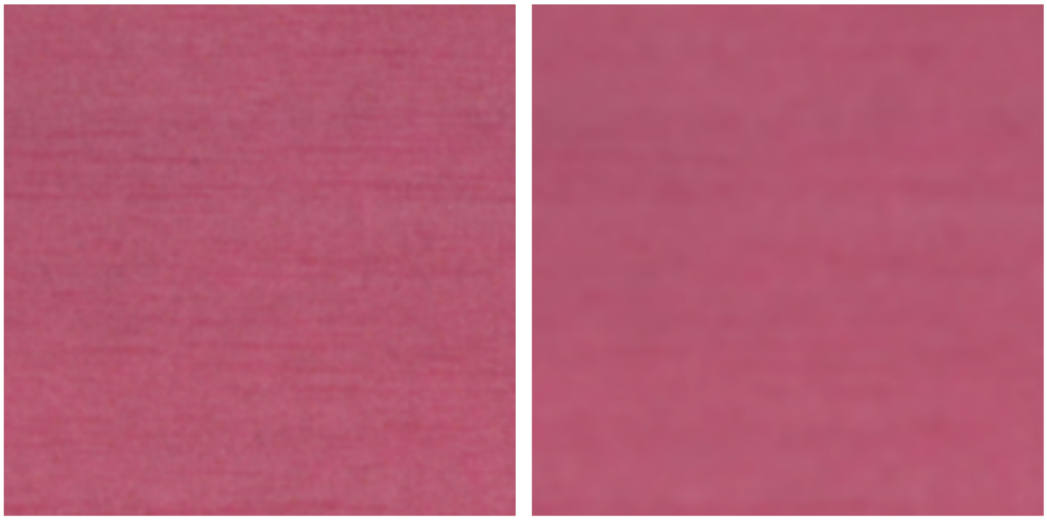
Figure 2.3: *2D DFT of woman_bw.tif halftoned with Floyd-Steinberg Error Diffusion (left) and Void-and-Cluster (right)*

Question 3 (Fan Bu)

Question 3

c

The zoomed-in images of a portion of the test page before and after 2-D Gaussian filtering.



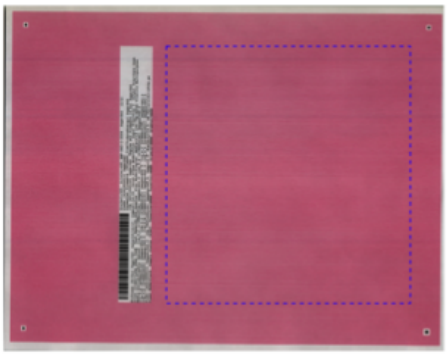
(a) A portion of the test page before filtering.

(b) A portion of the test page after filtering.

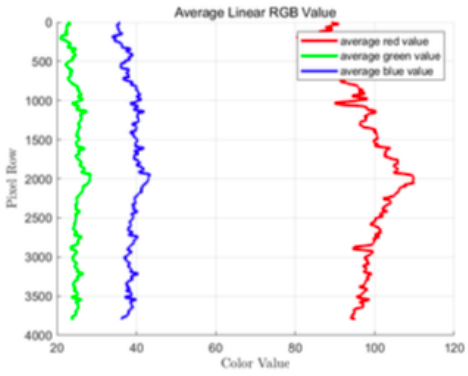
Gaussian filtered image looks smoother.

d

A region is selected as shown below, along with the average RGB values of each row.



(a) Selected region

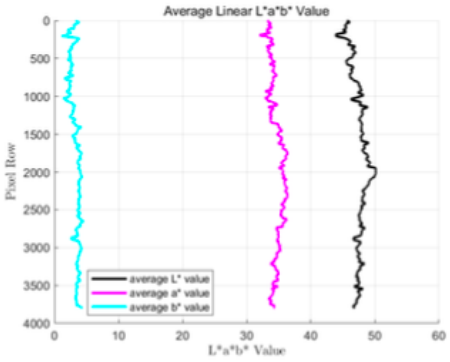


(b) Average Linear RGB Values.

e

A region is selected the same as above question, along with the average $L*a*b*$ values of each row.

(a) Selected region



(b) Average Linear $L*a*b*$ Values.

f

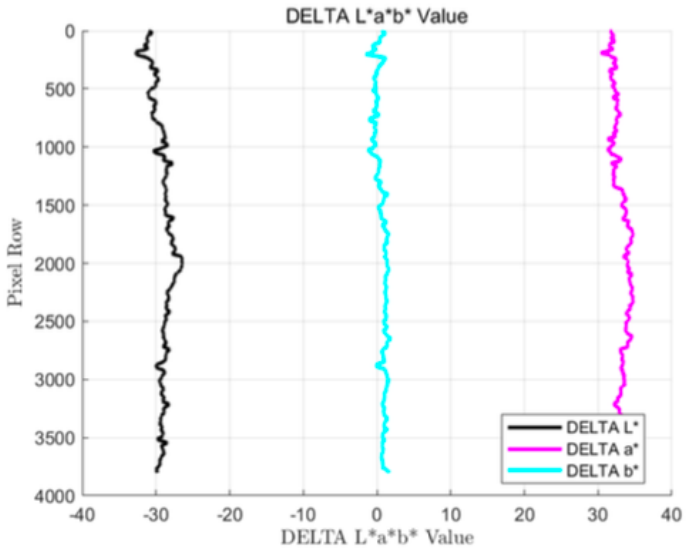
Codes attached to the Code Listing section.

The average RGB value for the white paper background is $R = 137.1379$, $G = 128.1318$, $B = 122.4957$.

The average $L*a*b*$ value for the white paper background is $L* = 76.6149$, $a* = 1.6212$, $b* = 2.6721$.

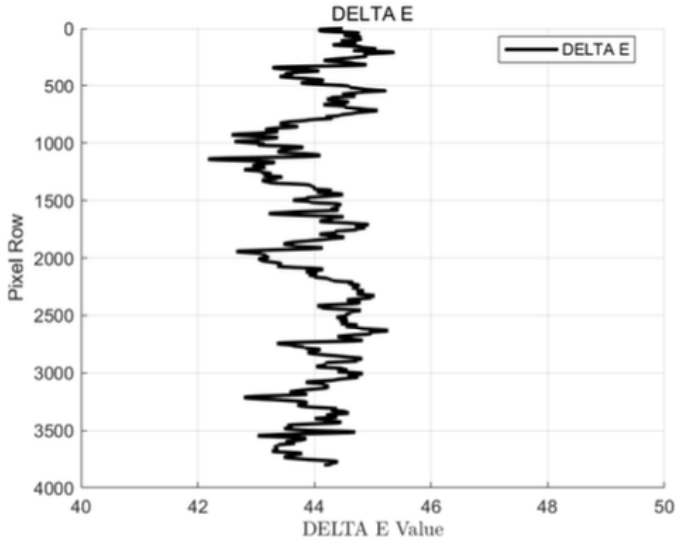
g

The DELTA $L*a*b*$ values are plotted here:



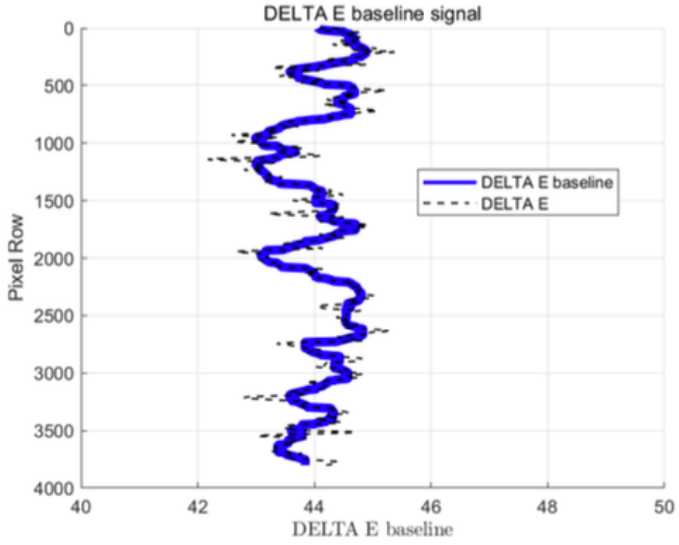
h

The DELTA E values are plotted here:



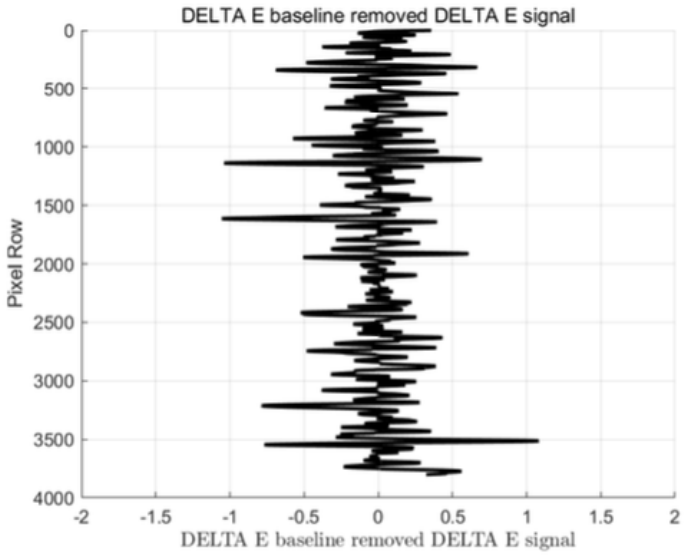
i

Here us a plot of my baseline DELTA E signal using a 100th-order median filter along with the DELTA E signal:



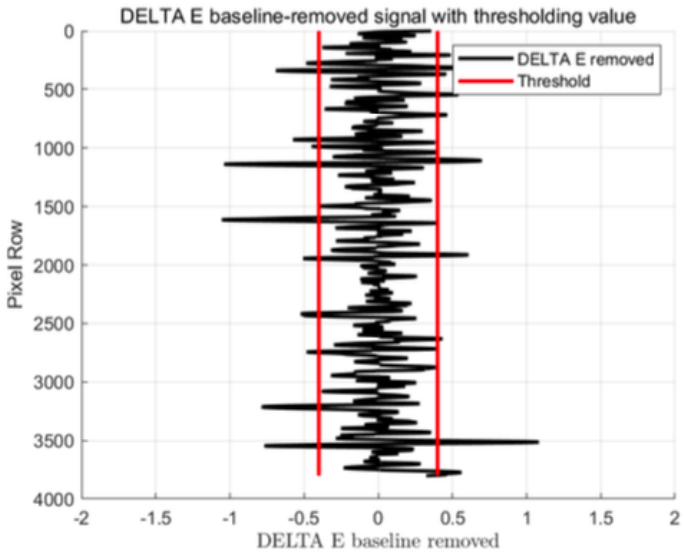
j

Subtract my baseline DELTA E signal from my original DELTA E signal:



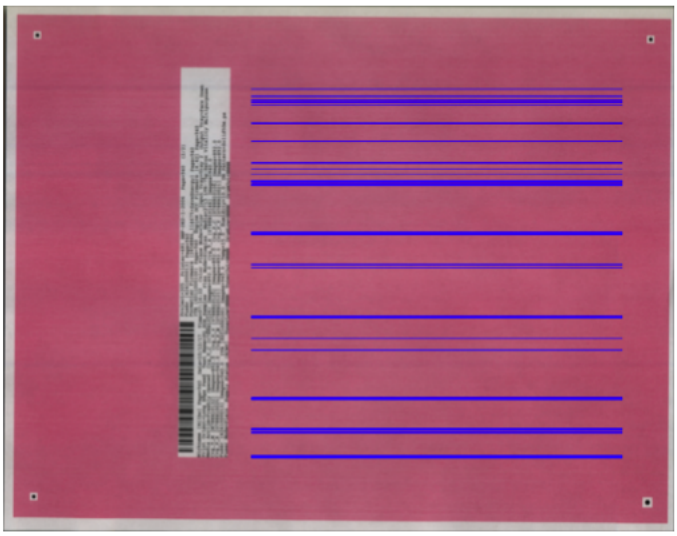
k

An appropriate threshold could be ±0.40, as shown below:



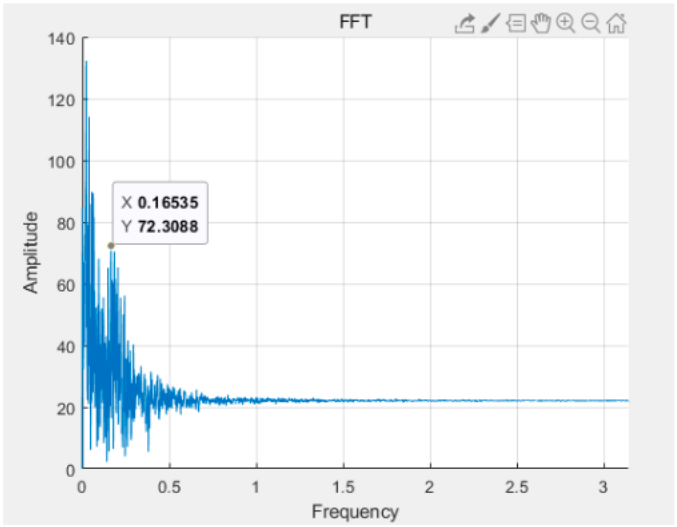
l

The bands are superimposed on the original page image and highlighted in blue:



m

We may FFT the Delta E removed signal and get the following plot:



In the frequency domain, a peak shows up at $f = 0.16535 = 0.05263\pi$. The image height is roughly 3800 pixels, so the estimated period is $\frac{0.05263\pi}{2\pi} \times 3800 \approx 100$ pixels

```
% Code for HW06 Question3, by Fan Bu, Nov. 2021.
clear all; close all; clc;

%% (a) Read image
img_testpage = double(imread('test_page_banding.tif'));

%% (b) Degamma
img_testpage_degamma = (img_testpage./255).^(2.2).*255;

%% (c) Gaussian filter


 sigma = 5;
 img_testpage_gauss = imgaussfilt(img_testpage_degamma,sigma);
 img_testpage_gauss_gamma = (img_testpage_gauss./255).^(1/2.2).*255;
 img_testpage_zoom = img_testpage(2000:2500, 3000:3500, :);
 imwrite(uint8(img_testpage_zoom),'HW06_3_c_origin.png');
 img_testpage_gauss_gamma_zoom = uint8(img_testpage_gauss_gamma(2000:2500, 3000:3500, :));
 imwrite(uint8(img_testpage_gauss_gamma_zoom),'HW06_3_c_gaussian.png');

 %% (d) Average linear rgb value
 img_testpage_gauss_clip = img_testpage_gauss(600:4400, 2400:6000, :);
 [img_h, img_w, img_ch] = size(img_testpage_gauss_clip);
 average_RGB_value = zeros(img_h,3);
 for i = 1:img_h
     row_temp = img_testpage_gauss_clip(i,:,:);
     average_RGB_value(i,:) = mean(row_temp);
 end
 figure(1);
 imshow(uint8(img_testpage));
 hold on;
 rectangle('Position',[2400,600,img_w,img_h],...
          'EdgeColor', 'b','LineWidth',2,'LineStyle','--');
 saveas(figure(1),"HW06_3_d_1.png")

 figure(2);
 hold on; grid on;
 plot(average_RGB_value(:,1), 1:img_h, 'r-','LineWidth', 2);
 plot(average_RGB_value(:,2), 1:img_h, 'g-','LineWidth', 2);
 plot(average_RGB_value(:,3), 1:img_h, 'b-','LineWidth', 2);
 set(gca, 'ydir', 'reverse');
 % legend('average red value', 'average green value', 'average blue value', 'Location', 'best');
 legend('average red value', 'average green value', 'average blue value');
 xlabel('Color Value', 'Interpreter','latex');
 ylabel('Pixel Row', 'Interpreter','latex');
 title('Average Linear RGB Value');
 saveas(figure(2),"HW06_3_d_2.png")

 %% (e) Transform to lab via xyz
 wh_pt = whitepoint('d50');
 T_d50  = [0.4360,0.3851,0.1431;
           0.2224,0.7169,0.0606;
```

```
          0.0139,0.0971,0.7139];
average_XYZ_value = (T_d50*(average_RGB_value./255)')';
average_lab_value = zeros(img_h,3);
for i = 1:img_h
    average_lab_value(i,:) = XYZ_2_Lab(average_XYZ_value(i,:),wh_pt);
end
figure(3);
hold on; grid on;
plot(average_lab_value(:,1), 1:img_h, 'k-','LineWidth', 2);
plot(average_lab_value(:,2), 1:img_h, 'm-','LineWidth', 2);
plot(average_lab_value(:,3), 1:img_h, 'c-','LineWidth', 2);
set(gca, 'ydir', 'reverse');
legend('average L* value', 'average a* value', 'average b* value', 'Location', 'best');
xlabel('L*a*b* Value', 'Interpreter','latex');
ylabel('Pixel Row', 'Interpreter','latex');
title('Average Linear L*a*b* Value');
saveas(figure(3),"HW06_3_e.png")

%% (f) concatenate the white background to a 1bynby3
background_R_value = 0; background_G_value = 0; background_B_value = 0;
background_clip_1 = img_testpage_gauss(40:5000, 40:80, :);
background_clip_2 = img_testpage_gauss(30:70, 150:6400, :);
background_clip_3 = img_testpage_gauss(5040:5090, 150:6400, :);
[img_h_1, img_w_1, img_ch_1] = size(background_clip_1);
[img_h_2, img_w_2, img_ch_2] = size(background_clip_2);
[img_h_3, img_w_3, img_ch_3] = size(background_clip_3);
cnt = 0;
for i = 1:img_h_1
    for j = 1:img_w_1
        background_R_value = background_R_value+background_clip_1(i,j,1);
        background_G_value = background_G_value+background_clip_1(i,j,2);
        background_B_value = background_B_value+background_clip_1(i,j,3);
        cnt = cnt + 1;
    end
end
for i = 1:img_h_2
    for j = 1:img_w_2
        background_R_value = background_R_value+background_clip_2(i,j,1);
        background_G_value = background_G_value+background_clip_2(i,j,2);
        background_B_value = background_B_value+background_clip_2(i,j,3);
```

```
        cnt = cnt + 1;
    end
end
for i = 1:img_h_3
    for j = 1:img_w_3
        background_R_value = background_R_value+background_clip_3(i,j,1);
        background_G_value = background_G_value+background_clip_3(i,j,2);
        background_B_value = background_B_value+background_clip_3(i,j,3);
        cnt = cnt + 1;
    end
end
background_R_value = background_R_value/cnt;
background_G_value = background_G_value/cnt;
background_B_value = background_B_value/cnt;
background_RGB = [background_R_value; background_G_value; background_B_value]
background_XYZ = T_d50*(background_RGB./255);
background_Lab = XYZ_2_Lab(background_XYZ,wh_pt)

%% (g) plot 1d DELTA Lab
Delta_Lab = average_lab_value-(repmat(background_Lab,1,img_h))';
figure(4);
hold on; grid on;
plot(Delta_Lab(:,1), 1:img_h, 'k-','LineWidth', 2);
plot(Delta_Lab(:,2), 1:img_h, 'm-','LineWidth', 2);
plot(Delta_Lab(:,3), 1:img_h, 'c-','LineWidth', 2);
set(gca, 'ydir', 'reverse');
legend('DELTA L*', 'DELTA a*', 'DELTA b*', 'Location', 'best');
xlabel('DELTA L*a*b* Value', 'Interpreter','latex');
ylabel('Pixel Row', 'Interpreter','latex');
title('DELTA L*a*b* Value');
saveas(figure(4),"HW06_3_g.png")

%% (h) DELTA E
Delta_E = sqrt(Delta_Lab(:,1).^2+Delta_Lab(:,2).^2+Delta_Lab(:,3).^2);
figure(5);
hold on; grid on;
plot(Delta_E, 1:img_h, 'k-','LineWidth', 2);
xlim([40,50]);
set(gca, 'ydir', 'reverse');
legend('DELTA E', 'Location', 'best');
```

```
title('DELTA E')
xlabel('DELTA E Value', 'Interpreter','latex');
ylabel('Pixel Row')
saveas(figure(5),"HW06_3_h.png")

%% (i) deltaE smooth
Delta_E_median_filtered = medfilt1(Delta_E,100);
figure(6);
hold on; grid on;
plot(Delta_E_median_filtered(2:end), 2:img_h, 'b-','LineWidth', 5);
plot(Delta_E(2:end), 2:img_h, 'k--','LineWidth', 1);
xlim([40,50]);
set(gca, 'ydir', 'reverse');
legend('DELTA E baseline', 'DELTA E', 'Location', 'best');
title('DELTA E baseline signal')
xlabel('DELTA E baseline', 'Interpreter','latex');
ylabel('Pixel Row');
saveas(figure(6),"HW06_3_i.png");

%% (j) baseline removed DELTA E signal
Delta_E_removed = Delta_E - Delta_E_median_filtered;
figure(7);
hold on; grid on;
plot(Delta_E_removed(2:end), 2:img_h, 'k-','LineWidth', 2);
set(gca, 'ydir', 'reverse');
xlim([-2,2]);
title('DELTA E baseline removed DELTA E signal')
xlabel('DELTA E baseline removed DELTA E signal', 'Interpreter','latex');
ylabel('Pixel Row');
saveas(figure(7),"HW06_3_j.png");

%% (k) DELTA E baseline-removed signal with thresholding value
threshold = 0.4;
thresholds_pos = repmat(threshold,img_h-1,1);
thresholds_neg = repmat(-threshold,img_h-1,1);
figure(8);
hold on; grid on;
plot(Delta_E_removed(2:end), 2:img_h, 'k-','LineWidth', 2);
plot(thresholds_pos, 2:img_h,'r-','LineWidth', 2);
plot(thresholds_neg, 2:img_h,'r-','LineWidth', 2);
```

```
set(gca, 'ydir', 'reverse');
xlim([-2,2]);
legend('DELTA E removed', 'Threshold');
title('DELTA E baseline-removed signal with thresholding value')
xlabel('DELTA E baseline removed', 'Interpreter','latex');
ylabel('Pixel Row');
saveas(figure(8),"HW06_3_k.png");

%% (l) superimpose bandlocation
img_band = img_testpage_gauss;
for i = 2:img_h
    if Delta_E_removed(i) > threshold || Delta_E_removed(i) < -threshold
        img_band(i+600,2400:6000,1) = zeros(img_w,1);
        img_band(i+600,2400:6000,2) = zeros(img_w,1);
        img_band(i+600,2400:6000,3) = 255*ones(img_w,1);
    end
end
% gamma
img_band_gamma = (img_band./255).^(1/2.2).*255;
imwrite(uint8(img_band_gamma),'HW06_3_l.png');

%% (m) Estimate the period of the band
Delta_E_removed_fft = fft(abs(Delta_E_removed)-mean(abs(Delta_E_removed)));
Delta_E_removed_fft = abs(Delta_E_removed_fft);
[~,max_idx] = max(Delta_E_removed_fft);
frequencies = 0:2*pi/(img_h-1):2*pi;
max_f = frequencies(max_idx);
max_period = 2*pi/max_f;
figure(9);
hold on; grid on;
plot(frequencies,Delta_E_removed_fft);
xlim([0,pi]);
title('FFT');
xlabel('Frequency');
ylabel('Amplitude');
saveas(figure(9),"HW06_3_m.png");


function lab_values = XYZ_2_Lab(XYZ_values,wh_pt)
lab_values = zeros(3,1);
```

```
x = XYZ_values(1)/wh_pt(1);
y = XYZ_values(2)/wh_pt(2);
z = XYZ_values(3)/wh_pt(3);

if (y <= 0.008856)&&(y >= 0)
    y = 7.787*y+16/116;
else
    if ((y <= 1)&&(y >= 0.008856))
        y = y^(1/3);
    end
end
if (x <= 0.008856)&&(x >= 0)
    x = 7.787*x+16/116;
else
    if ((x <= 1)&&(x >= 0.008856))
        x = x^(1/3);
    end
end
if (z <= 0.008856)&&(z >= 0)
    z = 7.787*z+16/116;
else
    if ((z <= 1)&&(z >= 0.008856))
        z = z^(1/3);
    end
end

lab_values(1) = 116*y-16;
lab_values(2) = 500*(x-y);
lab_values(3) = 200*(y-z);
end
```