CSE489 Machine Vision

Project #01: Image Printing Based on Halftoning

Student Name: Menna Allah Muahmmed Farouk

ID: 15P3050

Due date 18-11-2019

Due handed in 18-11-2019

- Abstract

The project is based on the concept of halftoning which replace each pixel with 3x3 matrix represent the color of this pixel, which is called dot representation, this technique has been used in printing process at the past.

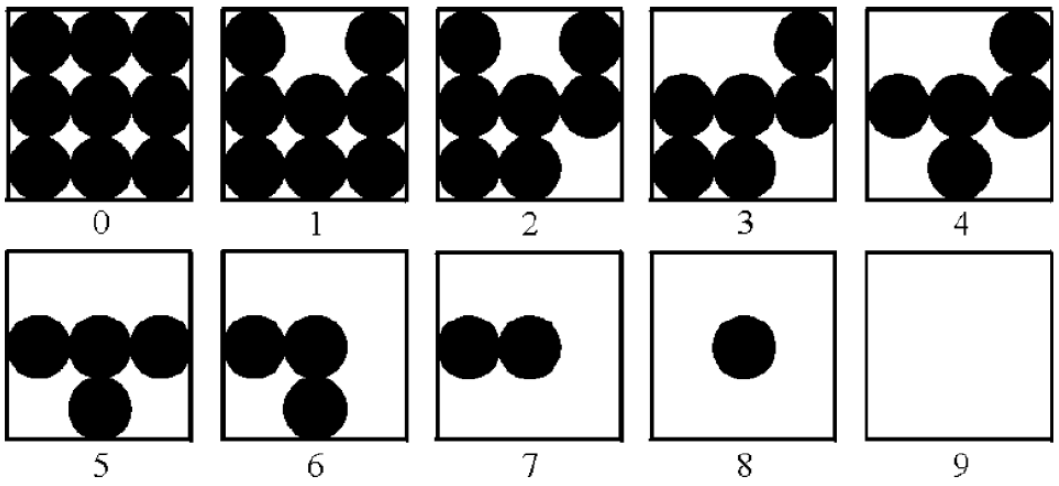
A couple of functions were implemented to reach the purpose of this project; scale_image function, gray_halftone function, colored_ halftone function which their inputs is the desired image, and to measure the resolution of a give image a ' find_resolution(dim1,dim2,dpi) ' function was used + to convert from RGB to grayscale image a ' gray_conversion('image') was used to create only 10 graylevels

- Technical discussion:

In order to produce the required output was to map each pixel of the input image to its 3x3 halftone patterns and save this output as a new array. Since the project used only 10 halftone patterns, the mapping was be carried out using if conditions that classify each pixel in one of 10 ranges by the usage of dots representation to correspond the value of each pixel with a 3x3 matrix

Therefore, before mapping the halftone patterns, the input image was read using ‘Image.Open or mpimg.imread’ in PIL library , matplotlib library in Python and then change the RGB image to grayscale using the function ‘im.convert('L')’. The 2 functions 'plt.imshow()', ‘pltshow()’ were used to show the final output image.

Creating an array triple the size of the original image. Using the function ' scale_image(image) to reduce the size of the original image by 3 times, to get the size of the original image matrix we use 'np.shape(I1), and the multiply it by 3 creating a new array of zeros then saving the result of the mapping in it.



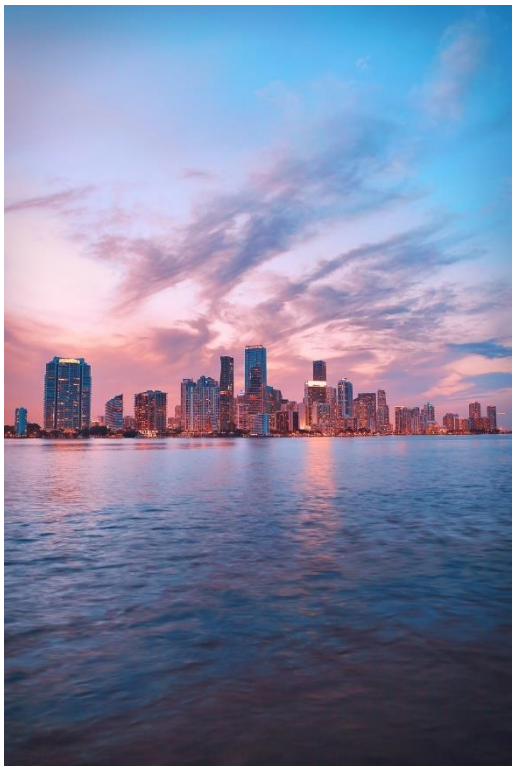
- Discussion of results:

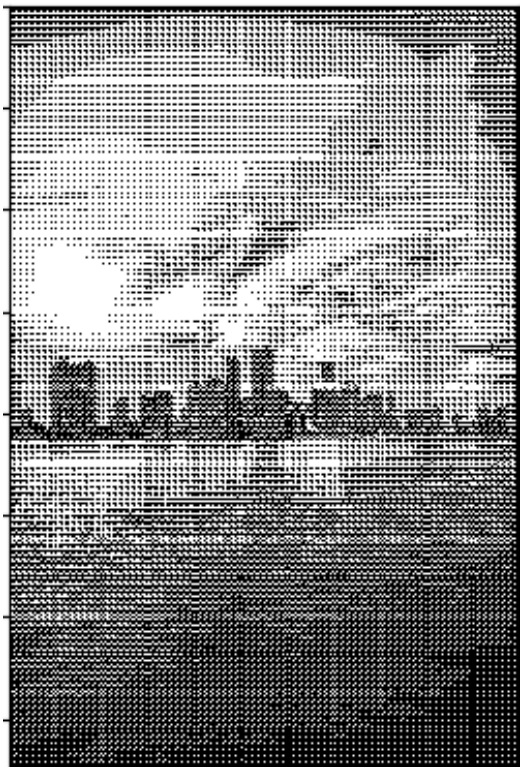
The images used were of different intensities and we used the halftoning technique on both gray and colored RGB images,

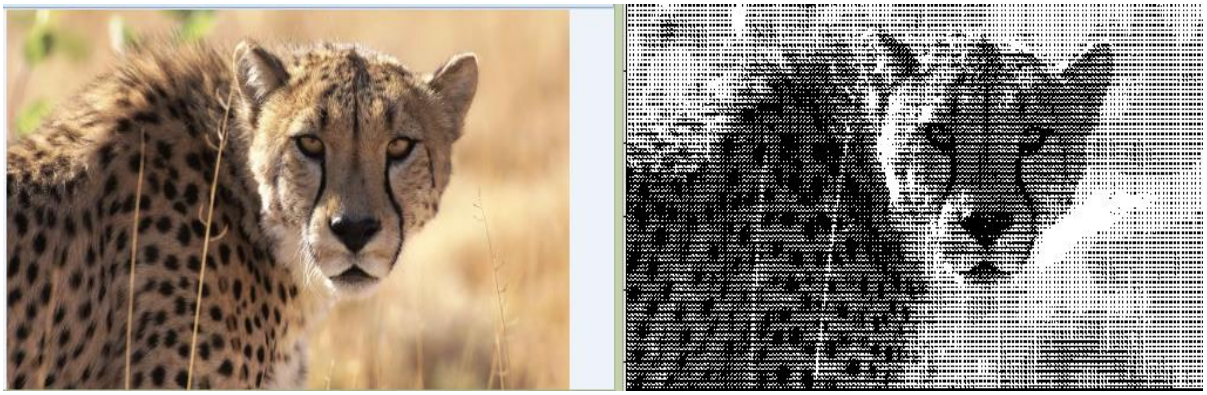
The results are affected by scaling and halftoning which cause the images to have less contrast and resolution

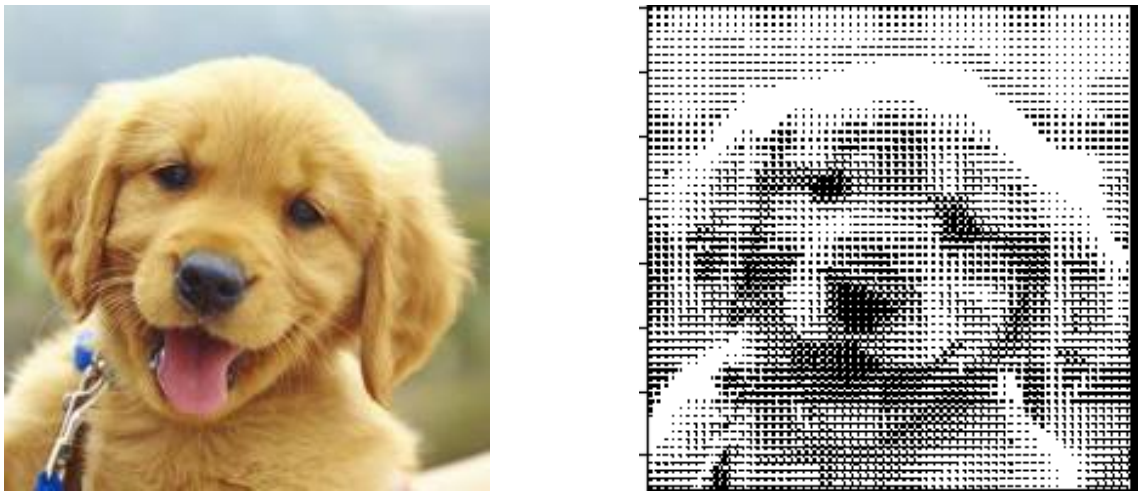
- Results:

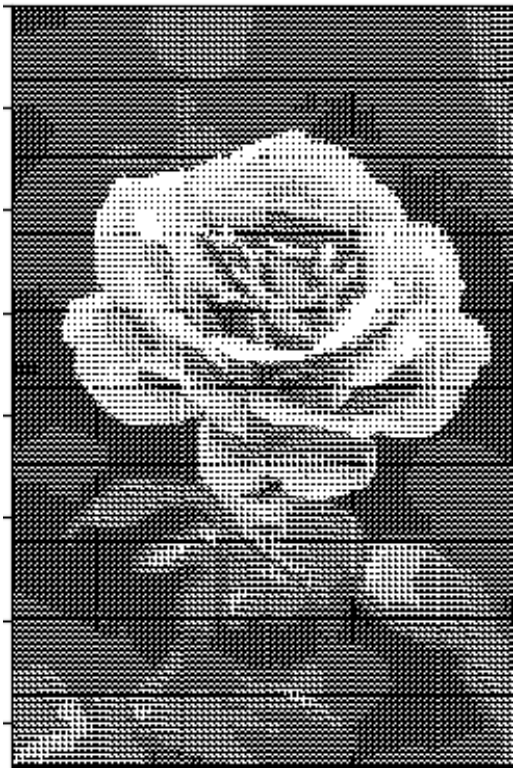
→Gray halftone

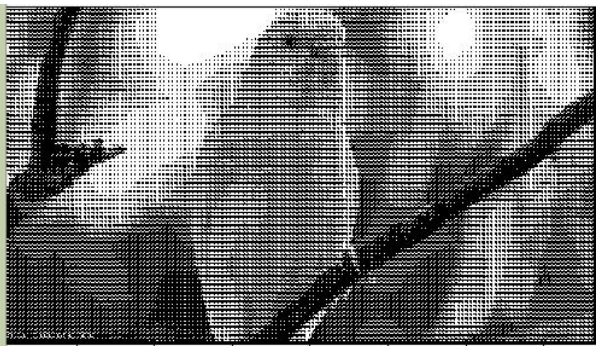




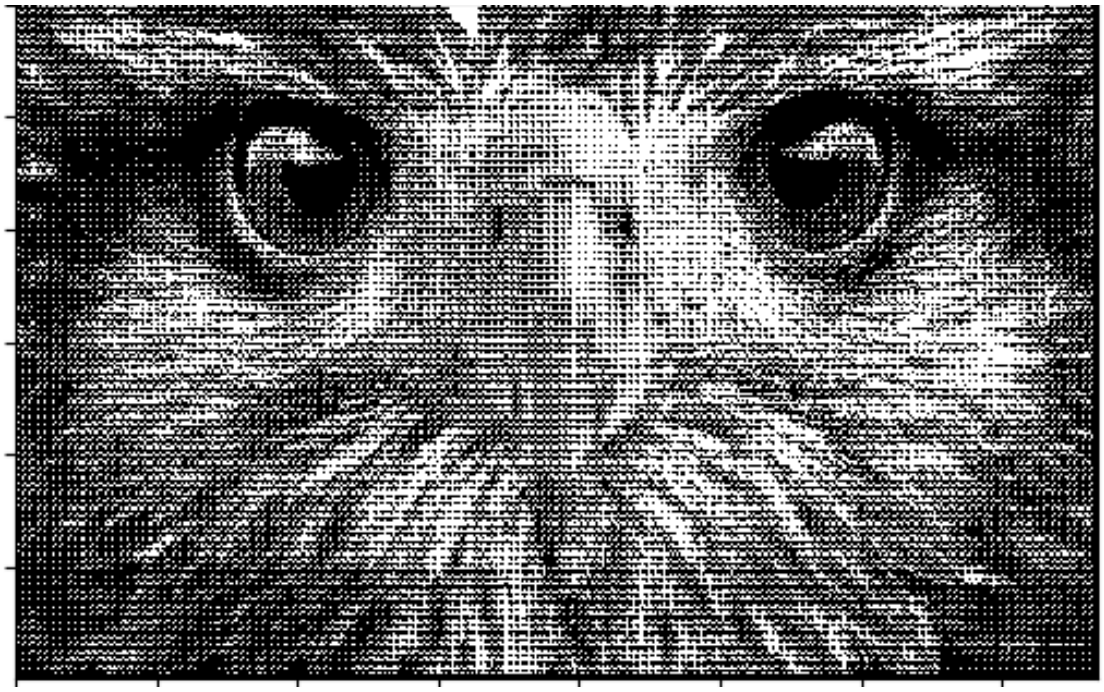


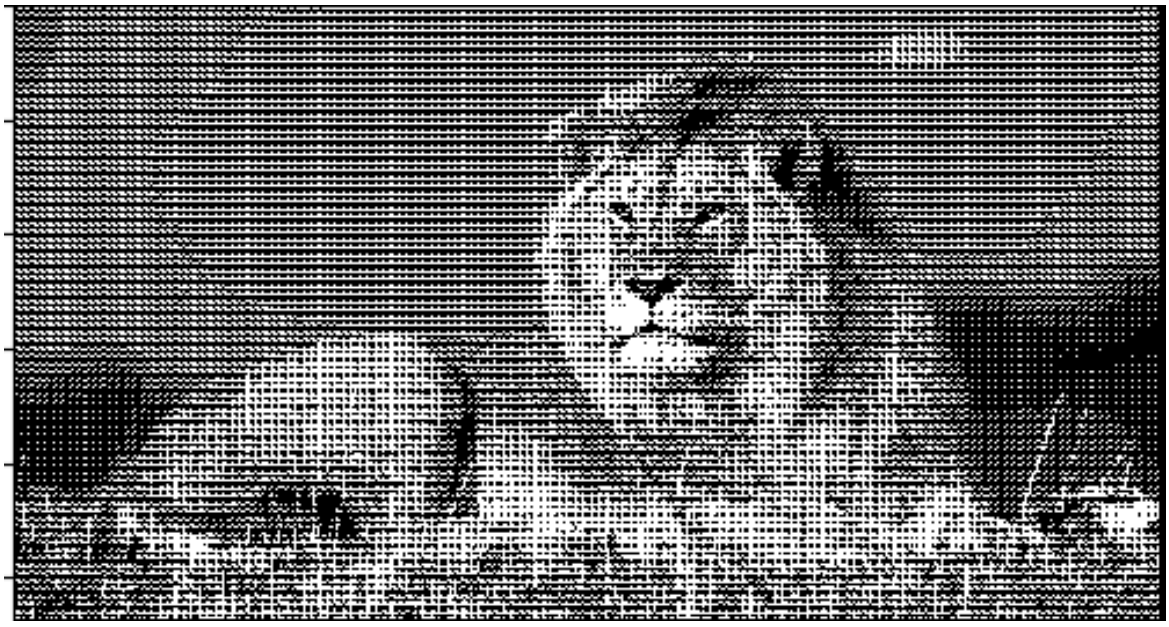







WWW.CRESTOCK.COM

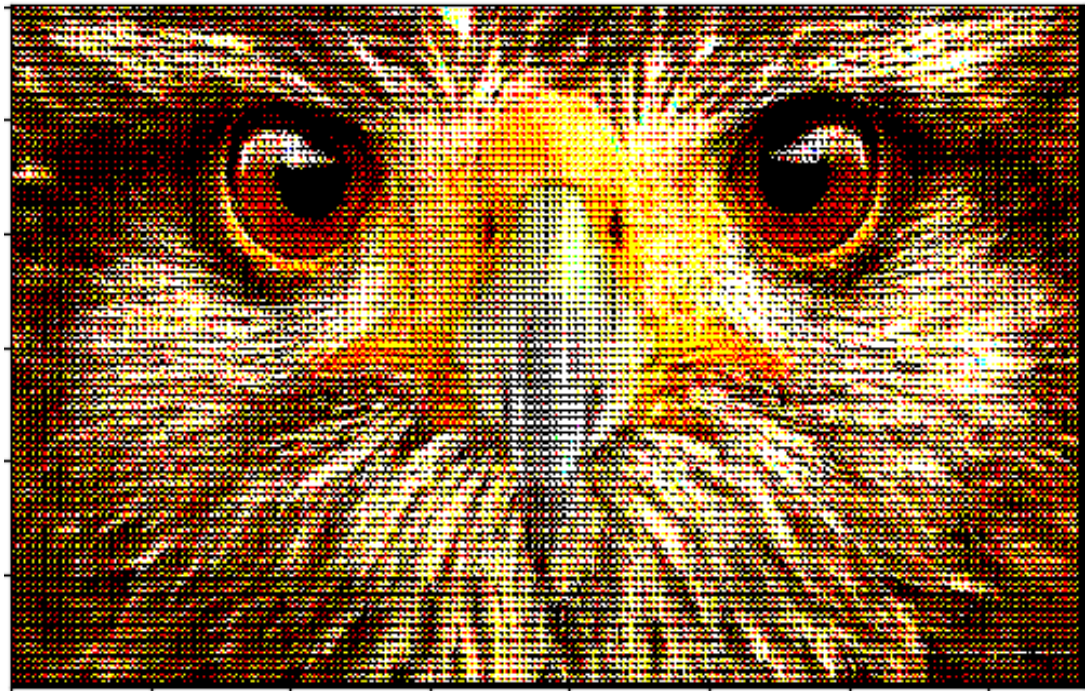


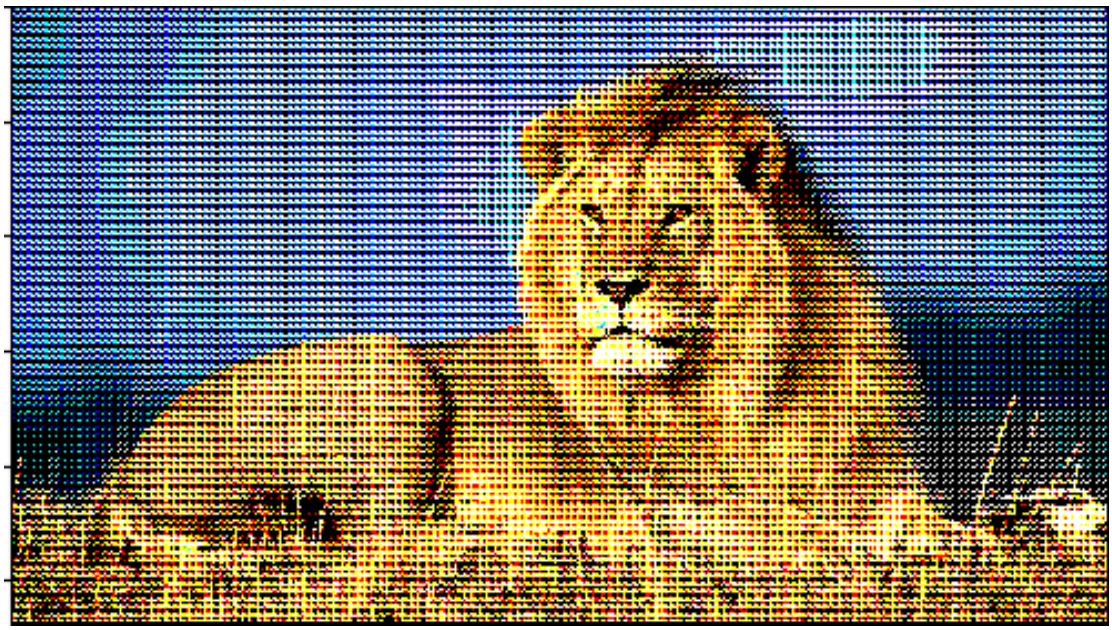


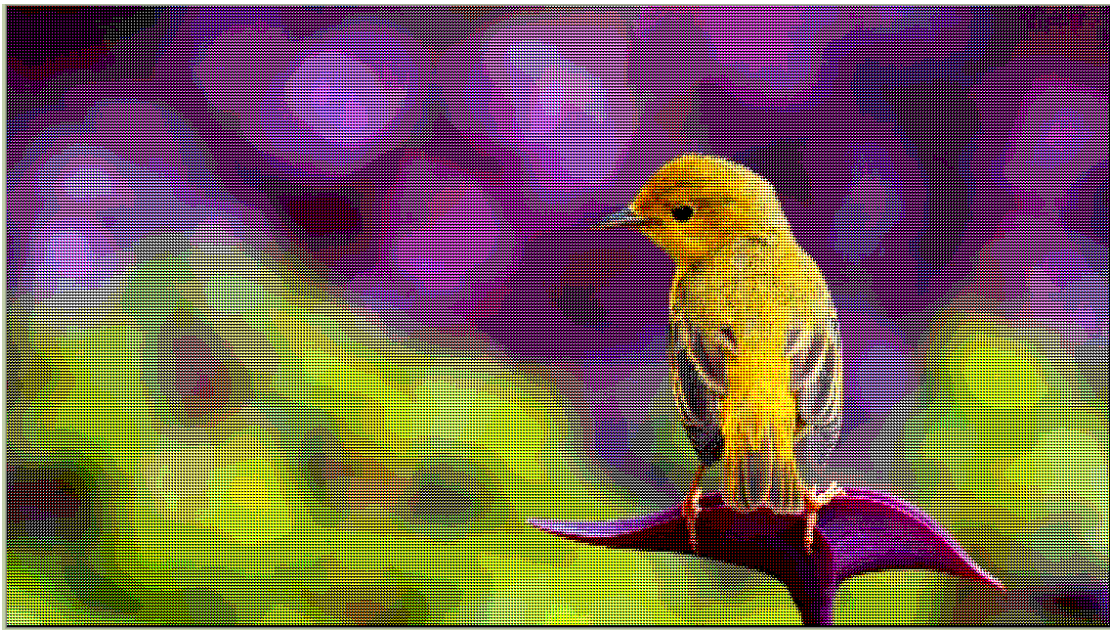
- Results:

→Colored halftone


WWW.CRESTOCK.COM







Question (1):

a-

```
def gray_conversion(image):
    im = Image.open(image)
    I1 = im.convert('L')  # 'L' for gray scale mode
    I2 = np.asarray(I1)
    H, W = np.shape(I2)
    gray_matrix = np.zeros([H, W])

    for i in range(H):
        for k in range(W):
            if 0 <= I2[i][k] < 25:
                gray_matrix[i][k] = 0
            elif 25 <= I2[i][k] < 50:
                gray_matrix[i][k] = 1
            elif 50 <= I2[i][k] < 75:
                gray_matrix[i][k] = 2
            elif 75 <= I2[i][k] < 100:
                gray_matrix[i][k] = 3
            elif 100 <= I2[i][k] < 125:
                gray_matrix[i][k] = 4
            elif 125 <= I2[i][k] < 150:
                gray_matrix[i][k] = 5
            elif 150 <= I2[i][k] < 175:
                gray_matrix[i][k] = 6
            elif 175 <= I2[i][k] < 200:
                gray_matrix[i][k] = 7
            elif 200 <= I2[i][k] < 225:
                gray_matrix[i][k] = 8
            elif 225 <= I2[i][k] < 255:
                gray_matrix[i][k] = 9

    print(gray_matrix)
    plt.imshow(gray_matrix, cmap=plt.get_cmap('gray'))
```

b- Resolution function

```python
def find_the_resolution(height, width, dpi):
    H = height * dpi
    W = width * dpi
    new_H = 3 * round(H / 3)
    new_W = 3 * round(W / 3)
    return new_H, new_W
```

c- Scale function

```python
def scale_image(image):
    im = Image.open(image)
    # I1 = im.convert('L')  # 'L' for gray scale mode
    I2 = np.asarray(im)
    H, W, S = np.shape(I2)
    h = math.floor(H / p)
    w = math.floor(W / p)
    after_img = np.zeros([h, w, S])
    for m in range(0, S):
        for i in range(1, h - 2):
            for k in range(1, w - 2):
                j = round(np.mean(I2[(i * p) - p:(i * p), (k * p) - p:(k * p), m]))
                after_img[i][k][m] = j
    plt.imshow(np.uint8(after_img))
    plt.show()
    return after_img
```

d- halftone function

```
def halftone(image):
    scale = scale1_image(image)
    H, W = np.shape(scale)
    new_H = 3 * H
    new_W = 3 * W
    halftone_matrix = np.zeros([new_H, new_W])
    grayscale_level0 = np.array([(0, 0, 0), (0, 0, 0), (0, 0, 0)])
    grayscale_level1 = np.array([(0, 1, 0), (0, 0, 0), (0, 0, 0)])
    grayscale_level2 = np.array([(0, 1, 0), (0, 0, 0), (0, 0, 1)])
    grayscale_level3 = np.array([(1, 1, 0), (0, 0, 0), (0, 0, 1)])
    grayscale_level4 = np.array([(1, 1, 0), (0, 0, 0), (1, 0, 1)])
    grayscale_level5 = np.array([(1, 1, 1), (0, 0, 0), (1, 0, 1)])
    grayscale_level6 = np.array([(1, 1, 1), (0, 0, 1), (1, 0, 1)])
    grayscale_level7 = np.array([(1, 1, 1), (0, 0, 1), (1, 1, 1)])
    grayscale_level8 = np.array([(1, 1, 1), (1, 0, 1), (1, 1, 1)])
    grayscale_level9 = np.array([(1, 1, 1), (1, 1, 1), (1, 1, 1)])

    for i in range(1, H):
        for k in range(1, W):
            if 0 <= scale[i][k] < 25:
                halftone_matrix[(i * 3) - 3:(i * 3), (k * 3) - 3:(k * 3)] = grayscale_level0
            elif 25 <= scale[i][k] < 50:
                halftone_matrix[(i * 3) - 3:(i * 3), (k * 3) - 3:(k * 3)] = grayscale_level1
            elif 50 <= scale[i][k] < 75:
                halftone_matrix[(i * 3) - 3:(i * 3), (k * 3) - 3:(k * 3)] = grayscale_level2
            elif 75 <= scale[i][k] < 100:
                halftone_matrix[(i * 3) - 3:(i * 3), (k * 3) - 3:(k * 3)] = grayscale_level3
            elif 100 <= scale[i][k] < 125:
                halftone_matrix[(i * 3) - 3:(i * 3), (k * 3) - 3:(k * 3)] = grayscale_level4
            elif 125 <= scale[i][k] < 150:
                halftone_matrix[(i * 3) - 3:(i * 3), (k * 3) - 3:(k * 3)] = grayscale_level5
```

```
            elif 150 <= scale[i][k] < 175:
                halftone_matrix[(i * 3) - 3:(i * 3), (k * 3) - 3:(k * 3)] = grayscale_level6
            elif 175 <= scale[i][k] < 200:
                halftone_matrix[(i * 3) - 3:(i * 3), (k * 3) - 3:(k * 3)] = grayscale_level7
            elif 200 <= scale[i][k] < 225:
                halftone_matrix[(i * 3) - 3:(i * 3), (k * 3) - 3:(k * 3)] = grayscale_level8
            elif 225 <= scale[i][k] < 255:
                halftone_matrix[(i * 3) - 3:(i * 3), (k * 3) - 3:(k * 3)] = grayscale_level9
    plt.imshow(halftone_matrix, cmap=plt.get_cmap('gray'))
    plt.show()
```

- Question (2):

```
def colored_halftone(image):
    I2 = scale_image(image)
    H, W, S = np.shape(I2)
    I1 = np.asarray(I2)
    new_H = H * 3
    new_W = W * 3
    colored_halftone_matrix = np.zeros([new_H, new_W, S], dtype=np.uint8)
    level0 = np.array([(0, 0, 0), (0, 0, 0), (0, 0, 0)])
    level1 = np.array([(0, 255, 0), (0, 0, 0), (0, 0, 0)])
    level2 = np.array([(0, 255, 0), (0, 0, 0), (0, 0, 255)])
    level3 = np.array([(255, 255, 0), (0, 0, 0), (0, 0, 255)])
    level4 = np.array([(255, 255, 0), (0, 0, 0), (255, 0, 255)])
    level5 = np.array([(255, 255, 255), (0, 0, 0), (255, 0, 255)])
    level6 = np.array([(255, 255, 255), (0, 0, 255), (255, 0, 255)])
    level7 = np.array([(255, 255, 255), (0, 0, 255), (255, 255, 255)])
    level8 = np.array([(255, 255, 255), (255, 0, 255), (255, 255, 255)])
    level9 = np.array([(255, 255, 255), (255, 255, 255), (255, 255, 255)])

    for j in range(0, S):
        for i in range(1, H - 2):
            for k in range(1, W - 2):
                if 0 <= I1[i][k][j] < 25.5:
                    colored_halftone_matrix[(i * 3) - 3:(i * 3), (k * 3) - 3:(k * 3), j] = level0
                elif 25.5 <= I1[i][k][j] < 51:
                    colored_halftone_matrix[(i * 3) - 3:(i * 3), (k * 3) - 3:(k * 3), j] = level1
                elif 51 <= I1[i][k][j] < 76.5:
                    colored_halftone_matrix[(i * 3) - 3:(i * 3), (k * 3) - 3:(k * 3), j] = level2
                elif 76.5 <= I1[i][k][j] < 102:
                    colored_halftone_matrix[(i * 3) - 3:(i * 3), (k * 3) - 3:(k * 3), j] = level3
                elif 102 <= I1[i][k][j] < 127.5:
                    colored_halftone_matrix[(i * 3) - 3:(i * 3), (k * 3) - 3:(k * 3), j] = level4
                elif 127.5 <= I1[i][k][j] < 153:
                    colored_halftone_matrix[(i * 3) - 3:(i * 3), (k * 3) - 3:(k * 3), j] = level5
```

```
                elif 153 <= I1[i][k][j] < 178.5:
                    colored_halftone_matrix[(i * 3) - 3:(i * 3), (k * 3) - 3:(k * 3), j] = level6
                elif 178.5 <= I1[i][k][j] < 204:
                    colored_halftone_matrix[(i * 3) - 3:(i * 3), (k * 3) - 3:(k * 3), j] = level7
                elif 204 <= I1[i][k][j] < 229.5:
                    colored_halftone_matrix[(i * 3) - 3:(i * 3), (k * 3) - 3:(k * 3), j] = level8
                elif 229.5 <= I1[i][k][j] < 255:
                    colored_halftone_matrix[(i * 3) - 3:(i * 3), (k * 3) - 3:(k * 3), j] = level9
    plt.imshow(colored_halftone_matrix)
    plt.show()
```