ECE 438 - Laboratory 10b
Image Processing (Week 2)

Last updated on April 24, 2022

Date:
Section:

Name	Signature	Time spent outside lab
Student Name #1 [---%]		
Student Name #2 [---%]		

	Below expectations	Lacks in some respect	Meets all expectations
Completeness of the report			
Organization of the report			
Quality of figures: *Correctly labeled with title, x-axis, y-axis, and name(s)*			
Understanding of color images and color spaces (40 pts): *Python figures with color components, code (`ycbcr2rgb`), filtered images, questions*			
Understanding of halftoning (60 pts): *Original and binary images with MSE, error images, table of MSE's for filered and nonfiltered images, questions*			

In [153]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [154]:
```
# make sure the plot is displayed in this notebook
%matplotlib inline
# specify the size of the plot
plt.rcParams['figure.figsize'] = (16, 10)

# for auto-reloading extenrnal modules
%load_ext autoreload
%autoreload 2
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

Exercise 2.3: Color

1. Load the image file `girl.tif`. Check the size of array for this image by using the command `print(image.shape)`, where `image` is the image matrix. Also, print the data type of this matrix.

Notice that this is a three dimensional array of type `uint8`. It contains three gray scale image planes corresponding to the red, green, and blue components for each pixel. Since each color pixel is represented by three bytes, this is commonly known as a 24-bit image.

In [155]:
```
girl = plt.imread("girl.tif")
print(girl.shape)
```

```
(256, 384, 3)
```

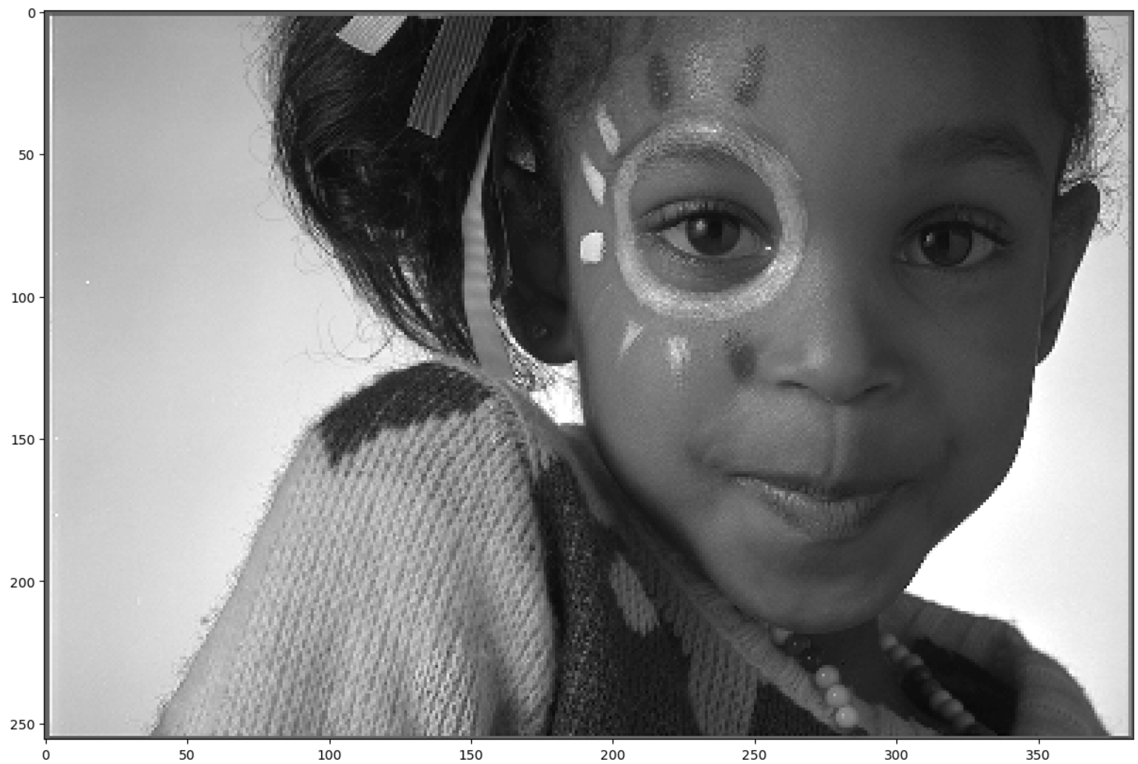
2. Display the image. Note that `cmap`, `vmin`, `vmax` arguments are not needed.

In [156]:
```
plt.imshow(girl)
plt.show()
```

3. Extract each of the color components, then plot each color component.

Note that while the original is a color image, each color component separately is a monochrome image, so plotting each color component requires `cmap` , `vmin` , `vmax`

In [157]:
```
plt.imshow(girl[:,:,0], cmap='gray', vmin=0, vmax=255)
plt.show()
```



In [158]:
```
plt.imshow(girl[:,:,1], cmap='gray', vmin=0, vmax=255)
plt.show()
```

In [159]:
```
plt.imshow(girl[:,:,2], cmap='gray', vmin=0, vmax=255)
plt.show()
```

4. Load the files `ycbcr.npy` using [`np.load()` (https://numpy.org/doc/stable/reference/generated/numpy.load.html)](https://numpy.org/doc/stable/reference/generated/numpy.load.html), and print its type and data shape `dtype` .

This file contains a NumPy array for a color image in YC_bC_r format. The array contains three gray scale image planes that correspond to the luminance (Y) and two chrominance (C_bC_r) components.

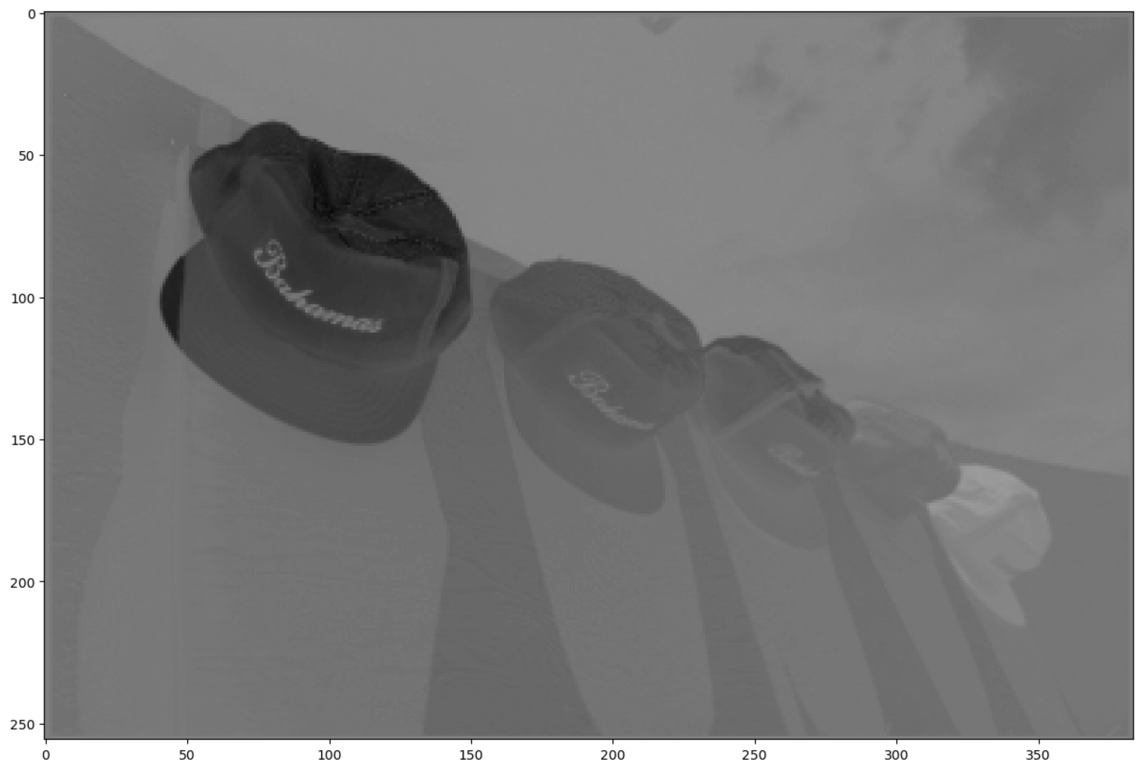
In [160]:
```
ycbcr = np.load("ycbcr.npy")
print(ycbcr.dtype)
print(ycbcr.shape)
B = ycbcr.astype(float)
```

```
uint8
(256, 384, 3)
```

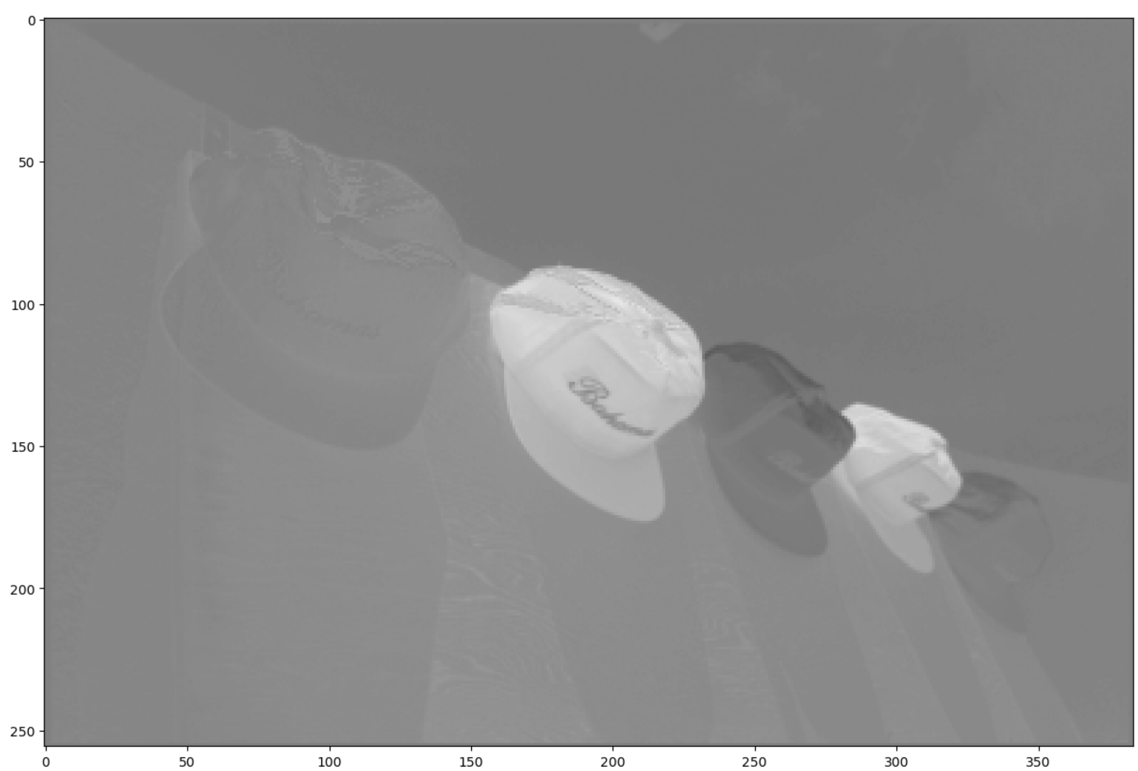
5. Plot each of the components.

In [161]:
```
plt.imshow(ycbcr[:,:,0], cmap='gray', vmin=0, vmax=255)
plt.show()
```

In [162]:
```
plt.imshow(ycbcr[:,:,1], cmap='gray', vmin=0, vmax=255)
plt.show()
```



In [163]:
```
plt.imshow(ycbcr[:,:,2], cmap='gray', vmin=0, vmax=255)
plt.show()
```



6. Complete the function below that will perform the transformation of equation (2). It should accept a 3-D YC_bC_r image array as input, and return a 3-D RGB image array.

- Make sure `ycbcr` is in `double` or `float` before any processing.
- After conversion, to make sure the values of `rgb` are in $[0, 255]$, use [`np.clip()` (https://numpy.org/doc/stable/reference/generated/numpy.clip.html)](https://numpy.org/doc/stable/reference/generated/numpy.clip.html).

In [164]:
```
def ycbcr2rgb(ycbcr):
    """
    Parameters
    ---
    ycbcr: image in YCbCr

    Returns
    ---
    rgb: image RGB
    """

    a = np.zeros(ycbcr.shape)
    rgb = np.zeros(ycbcr.shape)
    a[:,:,0] = ycbcr[:,:,0] + 1.4025*(ycbcr[:,:,2] - 128)
    a[:,:,1] = ycbcr[:,:,0] - 0.3443*(ycbcr[:,:,1] - 128) - 0.7144*(ycbcr[:
    a[:,:,2] = ycbcr[:,:,0] + 1.7730*(ycbcr[:,:,1] - 128)
    np.clip(a, 0, 255, out=rgb)
    return rgb
```

7. Now, convert the ycbcr array to an RGB representation and display the color image.

- Before displaying the image, make sure its data type is `np.uint8` .

In [165]:
```
rgb = ycbcr2rgb(B)
print(rgb.dtype)

plt.imshow(rgb.astype(np.uint8))
plt.show()
```

```
float64
```

8. Load the file `h.npy` . This is a 5×5 Gaussian filter with $\sigma^2 = 2.0$. (See the first week of the experiment for more details on this type of filter.)

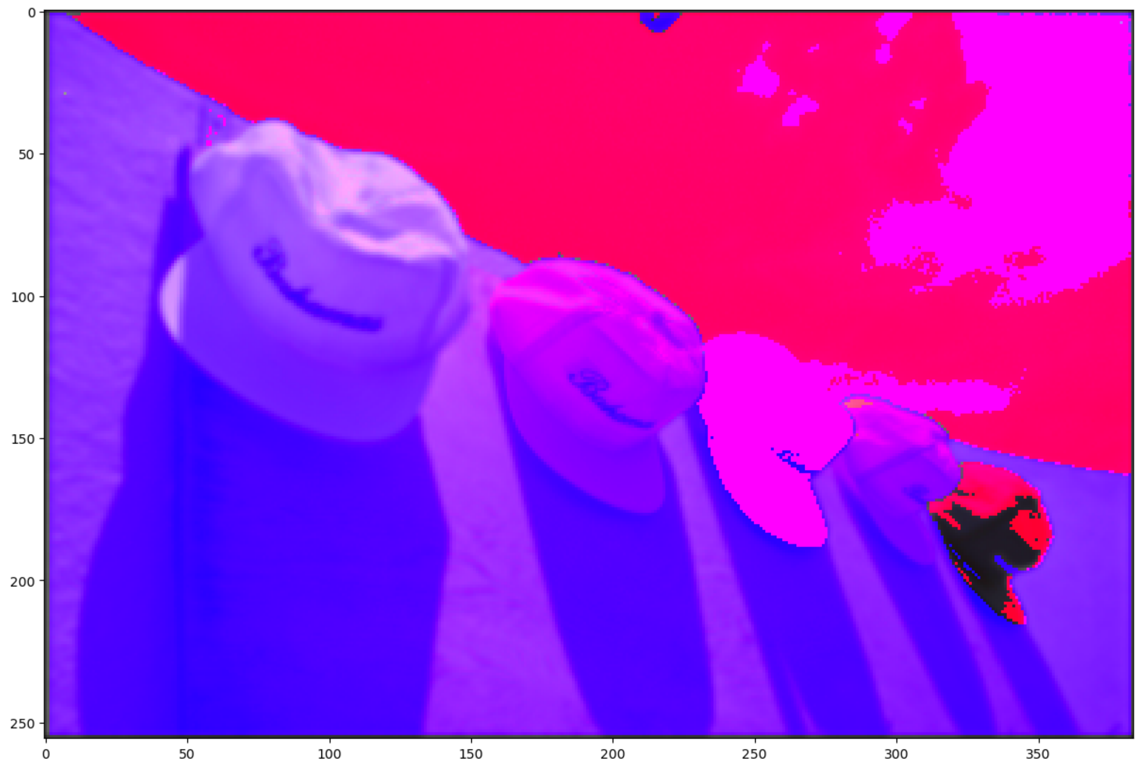
In [166]:
```
h = np.load("h.npy")
```

9. Alter the `ycbcr` array by filtering only the luminance component, `ycbcr[:,:,0]` , using the Gaussian filter (use `convolve2d()` function from last lab). Convert the result to RGB, and display it.

- Instead of altering the original `ycbcr` , you can create a copy by `ycbcr1 = ycbcr.copy()` .

In [167]:
```
def convolve2d(image, kernel):
    """
    Parameters
    ---
    image: the input image
    kernel: the filter

    Returns
    ---
    filtered: the filtered image
    """
    H,W = np.shape(image)
    s = len(kernel)
    filtered = np.zeros((H,W))
    new = np.zeros((H+s-1,W+s-1))

    for i in range(H):
        for j in range(W):
            new[i+s//2,j+s//2] = image[i,j]

    h = np.flip(kernel)

    for k in range(H):
        for l in range(W):
            filtered[k,l] = np.sum((new[k:k+s,l:l+s]*h))
    return filtered
```

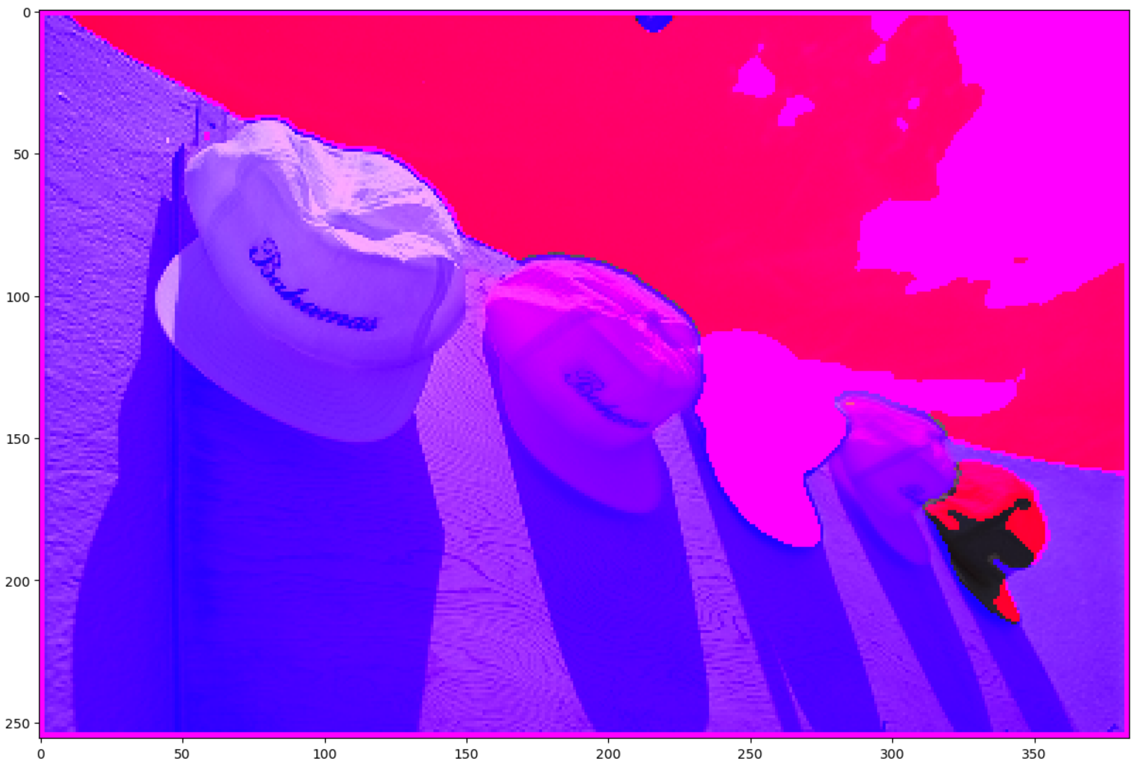
In [168]:
```
ycbcr1 = ycbcr.copy()

ycbcr1[:,:,0] = convolve2d(ycbcr1[:,:,0], h)

rgb1 = ycbcr2rgb(ycbcr1)

plt.imshow(rgb1.astype(np.uint8))
plt.show()
```



10. Now alter `ycbcr` by filtering both chrominance components, `ycbcr[:,:,1]` and `ycbcr[:,:,2]` , using the Gaussian filter. Convert this result to RGB, and display it.

- Again, instead of altering the original `ycbcr` , you can create a copy by `ycbcr2 = ycbcr.copy()` .

In [169]:
```
ycbcr2 = ycbcr.copy()

ycbcr2[:,:,1] = convolve2d(ycbcr2[:,:,1], h)
ycbcr2[:,:,2] = convolve2d(ycbcr2[:,:,2], h)
rgb2 = ycbcr2rgb(ycbcr2)

plt.imshow(rgb2.astype(np.uint8))
plt.show()
```



Exercise 3.2: Halftoning - Simple Thresholding

1. Load the grayscale image file `house.tif` and display it.

In [170]:
```
house = plt.imread("house.tif")

plt.imshow(house, cmap='gray', vmin=0, vmax=255)
plt.show()
```

2. Try the simple thresholding technique based on equation (3), using $T = 108$, and display the result.

- In Python, an easy way to threshold an image X is to use the command `Y = 255 * (X > T)`.

In [171]:
```
Y = 255*(house>108)

plt.imshow(Y, cmap='gray', vmin=0, vmax=255)
plt.show()
```

3. Create an “absolute error” image by subtracting the binary from the original image, and then taking the absolute value. The degree to which the original image is present in the error image is a measure of signal dependence of the quantization error. Display the error image.

In [172]:
```
Z = np.abs(house - Y)

plt.imshow(Z, cmap='gray', vmin=0, vmax=255)
plt.show()
```

4. Compute the mean square error (MSE), which is defined by

$$\text{MSE} = \frac{1}{MN} \sum_{i,j} \{f[i,j] - b[i,j]\}^2 \tag{9}$$

where MN is the total number of pixels in each image, f is the original image and b is the binarized image.

In [173]:
```
H,W = np.shape(house)
M = H*W
H1,W1 = np.shape(Y)
N = H1*W1

MSE = np.sum((house-Y)**2)/(M*N)
print(MSE)
```

```
0.084497663916813
```

Exercise 3.4: Halftoning - Ordered Dithering

1. Based on this index matrix and equation (6), create the corresponding threshold matrix and print it.

In [174]:
```
I = np.array([[12, 8, 10, 6], [4, 16, 2, 14], [9, 5, 11, 7], [1, 13, 3, 15]
H,W = np.shape(I)
T = 255 * (I - 0.5)/(H*W)
print(T)
```

```
[[183.28125 119.53125 151.40625  87.65625]
 [ 55.78125 247.03125  23.90625 215.15625]
 [135.46875  71.71875 167.34375 103.59375]
 [  7.96875 199.21875  39.84375 231.09375]]
```

2. Apply the ordered dithering and display the halftoned image.

In [175]:
```
H,W = np.shape(house)
num_y = H//4
num_x = W//4


m = np.tile(T, (num_y, num_x))

Y = 255*(house>m)

plt.imshow(Y, cmap='gray', vmin=0, vmax=255)
plt.show()
```

3. Compute the error image and display it.

In [176]:
```
Z = np.abs(house - Y)

plt.imshow(Z, cmap='gray', vmin=0, vmax=255)
plt.show()
```

4. Compute the MSE of the error image.

In [177]:
```
H,W = np.shape(house)
M = H*W
H1,W1 = np.shape(Y)
N = H1*W1

MSE = np.sum((house-Y)**2)/(M*N)
print(MSE)
```

```
0.1327400335835086
```

Exercise 3.6: Halftoning - Error Diffusion

1. Use the algorithm to create the halftoned image and display it.

In [181]:
```
house0 = plt.imread("house.tif")
house1 = house0.astype(np.double).copy()
H,W = np.shape(house1)
output = np.zeros((H,W))
e = np.zeros(house1.shape)

for i in range(1,H-1):
    for j in range(1,W-1):
        output[i,j] = 255*(house1[i,j]>108)
        e[i,j] = house1[i,j] - output[i,j]
        output[i,j+1] += e[i,j]*7/16
        output[i+1,j-1] += e[i,j]*3/16
        output[i+1,j] += e[i,j]*5/16
        output[i+1,j+1] += e[i,j]*1/16


plt.imshow(output.astype(np.uint8), cmap='gray', vmin=0, vmax=255)
plt.show()
plt.imshow(e.astype(np.uint8), cmap='gray', vmin=0, vmax=255)
plt.show()
```

2. Compute the error image and display it.

In [ ]:

```

```

3. Compute the MSE of the error image.

In [179]:

```
# insert your code here
```

4. By comparing three MSE values, is the MSE consistent with the visual quality?

insert your answer here

5. By looking at the error images, determine which method appears to be the least signal dependent? Does the signal dependence seem to be correlated with the visual quality?

insert your answer here

Exercise 3.7: Halftoning - Filtered Halftone

1. The human visual system naturally lowpass filters halftone images. To analyze this phenomenon, filter each of the halftone images with the Gaussian lowpass filter `h` that you loaded from `h.npy` , and measure the MSE of the filtered versions.

In [180]:
```
# insert your code here
```

2. Use the following template to make a table.

Halftone Method	Filtered	Not filtered
Simple Thresholding		
Ordered Dithering		
Error Diffusion		

3. Compare the MSE's of the filtered versions with the nonfiltered versions for each method. What is the implication of these observations with respect to how we perceive halftone images.

insert your answer here